Package 'mcmcr'

July 13, 2020

Title Manipulate MCMC Samples

Version 0.3.0

Description Functions and classes to store, manipulate and
summarise Monte Carlo Markov Chain (MCMC) samples. For more
information see Brooks et al. (2011) <isbn:978-1-4200-7941-8>.

License MIT + file LICENSE

URL https://github.com/poissonconsulting/mcmcr

BugReports https://github.com/poissonconsulting/mcmcr/issues

Depends R (>= 3.5)

Imports abind,
chk,
coda,
utils,
stats,
term,
nlist,
purrr,
universals,
extras,
lifecycle

Suggests covr,
graphics,
testthat,
rlang,
tibble

RdMacros lifecycle

Encoding UTF-8

Language en-US

LazyData true

RoxygenNote 7.1.1.9000

Roxygen list(markdown = TRUE)

R topics documented:

`as.mcarray` *Coerce to an mcarray object*

Description

Coerces MCMC objects to an mcarray object.

Usage

```
as.mcarray(x, ...)

## S3 method for class 'list'
as.mcmcr(x, ...)
```

Arguments

`x`	object to coerce.
`...`	Unused.

Methods (by class)

- `list`: Convert a list of uniquely named objects that can be coerced to [mcmcarray-object]s to an mcmcr object

Examples

```
as.mcarray(mcmcr_example$beta)
```

`as.mcmc.list.mcarray` *Turn an object into a tidy tibble*

Description

Turn an object into a tidy tibble

Usage

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

Arguments

`x`	An object to be converted into a tidy `tibble::tibble()`.
`...`	Additional arguments to tidying method.

Value

A `tibble::tibble()` with information about model components.

Methods

No methods found in currently loaded packages.

`as.mcmc.list.mcmc` *Turn an object into a tidy tibble*

Description

Turn an object into a tidy tibble

Usage

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

Arguments

`x`	An object to be converted into a tidy `tibble::tibble()`.
`...`	Additional arguments to tidying method.

Value

A `tibble::tibble()` with information about model components.

Methods

No methods found in currently loaded packages.

`as.mcmc.list.mcmc.list` *Turn an object into a tidy tibble*

Description

Turn an object into a tidy tibble

Usage

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

Arguments

`x`	An object to be converted into a tidy `tibble::tibble()`.
`...`	Additional arguments to tidying method.

Value

A `tibble::tibble()` with information about model components.

Methods

No methods found in currently loaded packages.

`as.mcmc.list.mcmcarray` *Turn an object into a tidy tibble*

Description

Turn an object into a tidy tibble

Usage

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

Arguments

`x`	An object to be converted into a tidy `tibble::tibble()`.
`...`	Additional arguments to tidying method.

Value

A `tibble::tibble()` with information about model components.

Methods

No methods found in currently loaded packages.

`as.mcmc.list.mcmcr` *Turn an object into a tidy tibble*

Description

Turn an object into a tidy tibble

Usage

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

Arguments

`x`	An object to be converted into a tidy `tibble::tibble()`.
`...`	Additional arguments to tidying method.

Value

A `tibble::tibble()` with information about model components.

Methods

No methods found in currently loaded packages.

`as.mcmc.mcarray` *Markov Chain Monte Carlo Objects*

Description

The function `mcmc` is used to create a Markov Chain Monte Carlo object. The input data are taken to be a vector, or a matrix with one column per variable.

If the optional arguments `start`, `end`, and `thin` are omitted then the chain is assumed to start with iteration 1 and have thinning interval 1. If `data` represents a chain that starts at a later iteration, the first iteration in the chain should be given as the `start` argument. Likewise, if `data` represents a chain that has already been thinned, the thinning interval should be given as the `thin` argument.

An mcmc object may be summarized by the `summary` function and visualized with the `plot` function.

MCMC objects resemble time series (`ts`) objects and have methods for the generic functions `time`, `start`, `end`, `frequency` and `window`.

Usage

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

Arguments

`x`	An object that may be coerced to an mcmc object
`...`	Further arguments to be passed to specific methods

Author(s)

Martyn Plummer

See Also

`mcmc.list`, `mcmcUpgrade`, `thin`, `window.mcmc`, `summary.mcmc`, `plot.mcmc`.

`as.mcmc.mcmc` *Markov Chain Monte Carlo Objects*

Description

The function `mcmc` is used to create a Markov Chain Monte Carlo object. The input data are taken to be a vector, or a matrix with one column per variable.

If the optional arguments `start`, `end`, and `thin` are omitted then the chain is assumed to start with iteration 1 and have thinning interval 1. If `data` represents a chain that starts at a later iteration, the first iteration in the chain should be given as the `start` argument. Likewise, if `data` represents a chain that has already been thinned, the thinning interval should be given as the `thin` argument.

An mcmc object may be summarized by the `summary` function and visualized with the `plot` function.

MCMC objects resemble time series (`ts`) objects and have methods for the generic functions `time`, `start`, `end`, `frequency` and `window`.

Usage

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

Arguments

`x`	An object that may be coerced to an mcmc object
`...`	Further arguments to be passed to specific methods

Author(s)

Martyn Plummer

See Also

`mcmc.list`, `mcmcUpgrade`, `thin`, `window.mcmc`, `summary.mcmc`, `plot.mcmc`.

as.mcmc.mcmcarray *Markov Chain Monte Carlo Objects*

Description

The function `mcmc` is used to create a Markov Chain Monte Carlo object. The input data are taken to be a vector, or a matrix with one column per variable.

If the optional arguments `start`, `end`, and `thin` are omitted then the chain is assumed to start with iteration 1 and have thinning interval 1. If `data` represents a chain that starts at a later iteration, the first iteration in the chain should be given as the `start` argument. Likewise, if `data` represents a chain that has already been thinned, the thinning interval should be given as the `thin` argument.

An mcmc object may be summarized by the `summary` function and visualized with the `plot` function.

MCMC objects resemble time series (`ts`) objects and have methods for the generic functions `time`, `start`, `end`, `frequency` and `window`.

Usage

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

Arguments

`x`	An object that may be coerced to an mcmc object
`...`	Further arguments to be passed to specific methods

Author(s)

Martyn Plummer

See Also

`mcmc.list`, `mcmcUpgrade`, `thin`, `window.mcmc`, `summary.mcmc`, `plot.mcmc`.

as.mcmc.mcmcr *Markov Chain Monte Carlo Objects*

Description

The function `mcmc` is used to create a Markov Chain Monte Carlo object. The input data are taken to be a vector, or a matrix with one column per variable.

If the optional arguments `start`, `end`, and `thin` are omitted then the chain is assumed to start with iteration 1 and have thinning interval 1. If `data` represents a chain that starts at a later iteration, the first iteration in the chain should be given as the `start` argument. Likewise, if `data` represents a chain that has already been thinned, the thinning interval should be given as the `thin` argument.

An mcmc object may be summarized by the `summary` function and visualized with the `plot` function.

MCMC objects resemble time series (`ts`) objects and have methods for the generic functions `time`, `start`, `end`, `frequency` and `window`.

Usage

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

Arguments

`x`	An object that may be coerced to an mcmc object
`...`	Further arguments to be passed to specific methods

Author(s)

Martyn Plummer

See Also

`mcmc.list`, `mcmcUpgrade`, `thin`, `window.mcmc`, `summary.mcmc`, `plot.mcmc`.

`as.mcmcarray` *Coerce to an mcmcarray object*

Description

Coerces MCMC objects to an `mcmcarray-object()`.

Usage

```
as.mcmcarray(x, ...)
```

Arguments

`x`	object to coerce.
`...`	Unused.

Examples

```
as.mcmcarray(as.mcarray(mcmcr_example$beta))
```

as.mcmcr — *Convert to an mcmcr Object*

Description

Converts an MCMC object to an `mcmcr-object()`.

Usage

```
as.mcmcr(x, ...)

## S3 method for class 'mcarray'
as.mcmcr(x, name = "par", ...)

## S3 method for class 'mcmcarray'
as.mcmcr(x, name = "par", ...)

## S3 method for class 'nlist'
as.mcmcr(x, ...)

## S3 method for class 'nlists'
as.mcmcr(x, ...)

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

## S3 method for class 'mcmc.list'
as.mcmcr(x, ...)

## S3 method for class 'mcmcrs'
as.mcmcr(x, ...)
```

Arguments

`x`	An MCMC object.
`...`	Unused.
`name`	A string specifying the parameter name.

Value

An mcmcr object.

Methods (by class)

- `mcarray`: Convert an mcarray object to an mcmcr object
- `mcmcarray`: Convert an `mcmcarray-object()` to an mcmcr object
- `nlist`: Convert an `nlist::nlist-object()` to an mcmcr object
- `nlists`: Convert an `nlist::nlists-object()` to an mcmcr object
- `mcmc`: Convert an `coda::mcmc()` object to an mcmcr object
- `mcmc.list`: Convert an `coda::mcmc.list()` object to an mcmcr object
- `mcmcrs`: Convert tan `mcmcrs-object()` to an mcmcr object

Examples

```
mcmc.list <- coda::as.mcmc.list(mcmcr::mcmcr_example)
as.mcmcr(mcmc.list)
```

`as.mcmcrs` *Convert to an mcmcrs object*

Description

Converts an MCMC object to an `mcmcrs-object()`.

Usage

```
as.mcmcrs(x, ...)

## S3 method for class 'list'
as.mcmcrs(x, ...)

## S3 method for class 'mcmcr'
as.mcmcrs(x, name = "mcmcr1", ...)
```

Arguments

`x`	An MCMC object.
`...`	Unused.
`name`	A string specifying the element name.

Value

An mcmcrs object.

Methods (by class)

- `list`: Convert a list of [mcmcr-object]s to an mcmcrs object
- `mcmcr`: Convert an `mcmcr-object()` to an mcmcrs object

Examples

```
as.mcmcrs(mcmcr::mcmcr_example)
```

`as_nlist.mcmc` *Coerce to nlist*

Description

Coerce an R object to an `nlist_object()`.

Usage

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

Arguments

`x`	An object.
`...`	Unused.

Value

An nlist object.

Methods (by class)

- `numeric`: Coerce named numeric vector to nlist
- `list`: Coerce list to nlist
- `data.frame`: Coerce data.frame to nlist

Examples

```
as_nlist(list(x = 1:4))
as_nlist(c(`a[2]` = 3, `a[1]` = 2))
```

`as_nlist.mcmc.list` *Coerce to nlist*

Description

Coerce an R object to an `nlist_object()`.

Usage

```
## S3 method for class 'mcmc.list'
as_nlist(x, ...)
```

Arguments

`x`	An object.
`...`	Unused.

Value

An nlist object.

Methods (by class)

- `numeric`: Coerce named numeric vector to nlist
- `list`: Coerce list to nlist
- `data.frame`: Coerce data.frame to nlist

Examples

```
as_nlist(list(x = 1:4))
as_nlist(c(`a[2]` = 3, `a[1]` = 2))
```

`as_nlist.mcmcr` *Coerce to nlist*

Description

Coerce an R object to an `nlist_object()`.

Usage

```
## S3 method for class 'mcmcr'
as_nlist(x, ...)
```

Arguments

`x`	An object.
`...`	Unused.

Value

An nlist object.

Methods (by class)

- `numeric`: Coerce named numeric vector to nlist
- `list`: Coerce list to nlist
- `data.frame`: Coerce data.frame to nlist

Examples

```
as_nlist(list(x = 1:4))
as_nlist(c(`a[2]` = 3, `a[1]` = 2))
```

`as_nlists.mcmc` *Coerce to nlists*

Description

Coerce an R object to an `nlists_object()`.

Usage

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

Arguments

`x`	An object.
`...`	Unused.

Value

An nlists object.

Methods (by class)

- `list`: Coerce list to nlists
- `nlist`: Coerce nlist to nlists

Examples

```
as_nlists(list(nlist(x = c(1, 5)), nlist(x = c(2, 3)), nlist(x = c(3, 2))))
```

`as_nlists.mcmc.list` *Coerce to nlists*

Description

Coerce an R object to an `nlists_object()`.

Usage

```
## S3 method for class 'mcmc.list'
as_nlists(x, ...)
```

Arguments

`x`	An object.
`...`	Unused.

Value

An nlists object.

Methods (by class)

- `list`: Coerce list to nlists
- `nlist`: Coerce nlist to nlists

Examples

```
as_nlists(list(nlist(x = c(1, 5)), nlist(x = c(2, 3)), nlist(x = c(3, 2))))
```

`as_nlists.mcmcr` *Coerce to nlists*

Description

Coerce an R object to an `nlists_object()`.

Usage

```
## S3 method for class 'mcmcr'
as_nlists(x, ...)
```

Arguments

`x`	An object.
`...`	Unused.

Value

An nlists object.

Methods (by class)

- `list`: Coerce list to nlists
- `nlist`: Coerce nlist to nlists

Examples

```
as_nlists(list(nlist(x = c(1, 5)), nlist(x = c(2, 3)), nlist(x = c(3, 2))))
```

`bind_chains.mcarray` *Bind by Chains.*

Description

Binds two MCMC objects (with the same parameters and iterations) by chains.

Usage

```
## S3 method for class 'mcarray'
bind_chains(x, x2, ...)
```

Arguments

`x`	An object.
`x2`	A second object.
`...`	Other arguments passed to methods.

Value

The combined object.

See Also

Other MCMC manipulations: `collapse_chains()`, `estimates()`, `split_chains()`

`bind_chains.mcmc` *Bind by Chains.*

Description

Binds two MCMC objects (with the same parameters and iterations) by chains.

Usage

```
## S3 method for class 'mcmc'
bind_chains(x, x2, ...)
```

Arguments

`x`	An object.
`x2`	A second object.
`...`	Other arguments passed to methods.

Value

The combined object.

See Also

Other MCMC manipulations: `collapse_chains()`, `estimates()`, `split_chains()`

`bind_chains.mcmc.list` *Bind by Chains.*

Description

Binds two MCMC objects (with the same parameters and iterations) by chains.

Usage

```
## S3 method for class 'mcmc.list'
bind_chains(x, x2, ...)
```

Arguments

`x`	An object.
`x2`	A second object.
`...`	Other arguments passed to methods.

Value

The combined object.

See Also

Other MCMC manipulations: `collapse_chains()`, `estimates()`, `split_chains()`

`bind_chains.mcmcarray` *Bind by Chains.*

Description

Binds two MCMC objects (with the same parameters and iterations) by chains.

Usage

```
## S3 method for class 'mcmcarray'
bind_chains(x, x2, ...)
```

Arguments

`x`	An object.
`x2`	A second object.
`...`	Other arguments passed to methods.

Value

The combined object.

See Also

Other MCMC manipulations: `collapse_chains()`, `estimates()`, `split_chains()`

`bind_chains.mcmcr` *Bind by Chains.*

Description

Binds two MCMC objects (with the same parameters and iterations) by chains.

Usage

```
## S3 method for class 'mcmcr'
bind_chains(x, x2, ...)
```

Arguments

`x`	An object.
`x2`	A second object.
`...`	Other arguments passed to methods.

Value

The combined object.

See Also

Other MCMC manipulations: `collapse_chains()`, `estimates()`, `split_chains()`

`bind_dimensions` *Combine two MCMC objects by dimensions*

Description

Combines multiple MCMC objects (with the same parameters, chains and iterations) by parameter dimensions.

Usage

```
bind_dimensions(x, x2, along = NULL, ...)
```

Arguments

`x`	An MCMC object.
`x2`	a second MCMC object.
`along`	A count (or NULL) indicating the parameter dimension to bind along.
`...`	Unused.

See Also

`bind_dimensions_n()`

Examples

```
bind_dimensions(mcmcr_example, mcmcr_example)
```

bind_dimensions_n *Combine multiple MCMC objects by parameter dimensions*

Description

Combines multiple MCMC objects (with the same parameters, chains and iterations) by parameter dimensions.

Usage

```
bind_dimensions_n(...)
```

Arguments

`...`	one or more MCMC objects

See Also

`bind_dimensions()`

Examples

```
bind_dimensions_n(mcmcr_example, mcmcr_example, mcmcr_example)
```

bind_iterations *Combine two MCMC objects by iterations*

Description

Combines two MCMC objects (with the same parameters and chains) by iterations.

Usage

```
bind_iterations(x, x2, ...)
```

Arguments

`x`	an MCMC object
`x2`	a second MCMC object
`...`	unused

Examples

```
bind_iterations(mcmcr_example, mcmcr_example)
```

`bind_parameters` *Combine two MCMC object by parameters*

Description

Combines two MCMC objects (with the same chains and iterations) by their parameters.

Usage

```
bind_parameters(x, x2, ...)
```

Arguments

`x`	an MCMC object
`x2`	a second MCMC object
`...`	unused

Examples

```
bind_parameters(
  subset(mcmcr_example, pars = "sigma"),
  subset(mcmcr_example, pars = "beta")
)
```

`check_mcmcarray` **Soft-deprecated** *Check mcmcarray*

Description

Soft-deprecated Check mcmcarray

Usage

```
check_mcmcarray(x, x_name = substitute(x), error = TRUE)
```

Arguments

`x`	The object to check.
`x_name`	A string of the name of the object.
`error`	A flag indicating whether to throw an informative error or immediately generate an informative message if the check fails.

Value

An invisible copy of x (it if doesn't throw an error).

Examples

```
check_mcmcarray(mcmcr::mcmcr_example$beta)
```

`check_mcmcr` **Soft-deprecated** *Check mcmcr*

Description

Soft-deprecated Check mcmcr

Usage

```
check_mcmcr(x, sorted = FALSE, x_name = substitute(x), error = TRUE)
```

Arguments

`x`	The object to check.
`sorted`	A flag specifying whether the parameters must be sorted.
`x_name`	A string of the name of the object.
`error`	A flag indicating whether to throw an informative error or immediately generate an informative message if the check fails.

Value

An invisible copy of x (it if doesn't throw an error).

Examples

```
check_mcmcr(mcmcr::mcmcr_example)
```

`chk_mcmcr` *Check MCMC Objects*

Description

Checks class and structure of MCMC objects.

`chk_mcmcarray` checks if `mcmcarray-object()` object using

`is.array(x) && is.numeric(x)`

`chk_mcmcr` checks if an `mcmcr-object()`.

`chk_mcmcrs` checks if an `mcmcrs-object()`.

Usage

```
chk_mcmcarray(x, x_name = NULL)

chk_mcmcr(x, x_name = NULL)

chk_mcmcrs(x, x_name = NULL)
```

Arguments

`x`	The object to check.
`x_name`	A string of the name of object x or NULL.

Details

To just check class use `chk::chk_s3_class()`.

Value

`NULL`, invisibly. Called for the side effect of throwing an error if the condition is not met.

Functions

- `chk_mcmcarray`: Check mcmcarray Object
- `chk_mcmcr`: Check mcmcr Object
- `chk_mcmcrs`: Check mcmcrs Object

See Also

`vld_mcmcr()`

Examples

```
# chk_mcmcarray
try(chk_mcmcarray(1))

# chk_mcmcr
chk_mcmcr(as.mcmcr(list(x = 1)))
try(chk_mcmcr(1))

# chk_mcmcrs
chk_mcmcrs(as.mcmcrs(as.mcmcr(list(x = 1))))
try(chk_mcmcrs(1))
```

`coef` *Term Coefficients*

Description

Gets coefficients for all the terms in an MCMC object.

Usage

```
## S3 method for class 'mcmc'
coef(object, conf_level = 0.95, estimate = median, ...)
```

Arguments

`object`	The MCMC object to get the coefficients for
`conf_level`	A number specifying the confidence level. By default 0.95.
`estimate`	The function to use to calculate the estimate.
`...`	Unused

Value

An data frame of the coefficients with the columns indicating the `term`, `estimate`, standard deviation (`sd`), `zscore`, `lower` and `upper` credible intervals and `pvalue`.

Methods (by class)

- `mcmc`: Get coefficients for terms in mcmc object

See Also

stats::[coef][stats::coef]

Examples

```
coef(mcmcr_example)
```

`collapse_chains.default` *Collapse Chains*

Description

Collapses an MCMC object's chains into a single chain.

Usage

```
## Default S3 method:
collapse_chains(x, ...)
```

Arguments

`x`	An object.
`...`	Other arguments passed to methods.

Value

The modified object with one chain.

See Also

Other MCMC manipulations: `bind_chains()`, `estimates()`, `split_chains()`

`collapse_chains.mcmc.list` *Collapse Chains*

Description

Collapses an MCMC object's chains into a single chain.

Usage

```
## S3 method for class 'mcmc.list'
collapse_chains(x, ...)
```

Arguments

`x`	An object.
`...`	Other arguments passed to methods.

Value

The modified object with one chain.

See Also

Other MCMC manipulations: `bind_chains()`, `estimates()`, `split_chains()`

`collapse_chains.mcmcr` *Collapse Chains*

Description

Collapses an MCMC object's chains into a single chain.

Usage

```
## S3 method for class 'mcmcr'
collapse_chains(x, ...)
```

Arguments

`x`	An object.
`...`	Other arguments passed to methods.

Value

The modified object with one chain.

See Also

Other MCMC manipulations: `bind_chains()`, `estimates()`, `split_chains()`

`combine_dimensions` *Combine Samples by Dimensions*

Description

Combines MCMC object samples by dimensions along `along` using `fun`.

Usage

```
combine_dimensions(x, fun = mean, along = NULL, ...)
```

Arguments

`x`	An MCMC object
`fun`	The function to use when combining dimensions
`along`	A positive integer (or NULL) indicating the parameter dimension(s) to bind along.
`...`	Unused

Value

The MCMC object with reduced dimensions.

Examples

```
combine_dimensions(mcmcr_example$alpha)
```

`combine_samples` *Combine MCMC Samples of Two Objects*

Description

Combines samples of two MCMC objects (with the same parameters, chains and iterations) using a function.

Usage

```
combine_samples(x, x2, fun = mean, ...)
```

Arguments

`x`	An MCMC object.
`x2`	A second MCMC object.
`fun`	The function to use to combine the samples. The function must return a scalar.
`...`	Unused.

Value

The combined samples as an MCMC object with the same parameters, chains and iterations as the original objects.

Examples

```
combine_samples(mcmcr_example, mcmcr_example, fun = sum)
```

`combine_samples_n` *Combine MCMC Samples of multiple objects*

Description

Combines samples of multiple MCMC objects (with the same parameters, chains and iterations) using a function.

Usage

```
combine_samples_n(x, ..., fun = mean)
```

Arguments

`x`	An MCMC object (or a list of mcmc objects).
`...`	Additional MCMC objects.
`fun`	A function.

Examples

```
combine_samples_n(mcmcr_example, mcmcr_example, mcmcr_example, fun = sum)
```

`complete_terms.mcmc` *Complete Terms*

Description

Adds any absent elements to an mcmc object.

Usage

```
## S3 method for class 'mcmc'
complete_terms(x, silent = FALSE, ...)
```

Arguments

`x`	An mcmc object.
`silent`	A flag specifying whether to suppress warning messages.
`...`	Unused

Details

The terms are repaired before being completed. Missing or invalid or inconsistent terms are dropped with a warning.

Value

The repaired and complete mcmc object.

Examples

```
mcmc <- coda::as.mcmc(subset(mcmcr::mcmcr_example, chain = 1L))
mcmc <- mcmc[, -c(1, 5, 7)]
term::complete_terms(mcmc)
```

`converged.default` *Converged*

Description

Tests whether an object has converged.

Usage

```
## Default S3 method:
converged(
  x,
  rhat = 1.1,
  esr = 0.33,
  by = "all",
  as_df = FALSE,
  na_rm = FALSE,
  ...
)
```

Arguments

`x`	An object.
`rhat`	The maximum rhat value.
`esr`	The minimum effective sampling rate.
`by`	A string indicating whether to determine by "term", "parameter" or "all".
`as_df`	A flag indicating whether to return the values as a data frame versus a named list.
`na_rm`	A flag specifying whether to ignore missing values.
`...`	Other arguments passed to methods.

Value

A logical scalar indicating whether the object has converged.

See Also

Other convergence: `converged_pars()`, `converged_terms()`, `esr_pars()`, `esr_terms()`, `esr()`, `rhat_pars()`, `rhat_terms()`, `rhat()`

Examples

```
converged(mcmcr_example)
```

`converged.mcmcrs` *Converged*

Description

Tests whether an object has converged.

Usage

```
## S3 method for class 'mcmcrs'
converged(
  x,
  rhat = 1.1,
  esr = 0.33,
  by = "all",
  as_df = FALSE,
  bound = FALSE,
  na_rm = FALSE,
  ...
)
```

Arguments

`x`	An object.
`rhat`	The maximum rhat value.
`esr`	The minimum effective sampling rate.
`by`	A string indicating whether to determine by "term", "parameter" or "all".
`as_df`	A flag indicating whether to return the values as a data frame versus a named list.
`bound`	flag specifying whether to bind mcmcrs objects by their chains before calculating rhat.
`na_rm`	A flag specifying whether to ignore missing values.
`...`	Other arguments passed to methods.

Value

A logical scalar indicating whether the object has converged.

See Also

Other convergence: `converged_pars()`, `converged_terms()`, `esr_pars()`, `esr_terms()`, `esr()`, `rhat_pars()`, `rhat_terms()`, `rhat()`

Examples

```
converged(mcmcrs(mcmcr_example, mcmcr_example))
converged(mcmcrs(mcmcr_example, mcmcr_example), bound = TRUE)
```

---

esr.mcarray *Effective Sampling Rate*

Description

Calculates the effective sampling rate (`esr`).

Usage

```
## S3 method for class 'mcarray'
esr(x, by = "all", as_df = FALSE, na_rm = FALSE, ...)
```

Arguments

`x`	An object.
`by`	A string indicating whether to determine by "term", "parameter" or "all".
`as_df`	A flag indicating whether to return the values as a data frame versus a named list.
`na_rm`	A flag specifying whether to ignore missing values.
`...`	Other arguments passed to methods.

Details

By default

$$\frac{1}{1 + 2\sum_{k=1}^{\infty} \rho_k(\theta)}$$

from Brooks et al. (2011) where the infinite sum is truncated at lag k when $\rho_{k+1}(\theta) < 0$.

Value

A number between 0 and 1 indicating the esr value.

References

Brooks, S., Gelman, A., Jones, G.L., and Meng, X.-L. (Editors). 2011. Handbook for Markov Chain Monte Carlo. Taylor & Francis, Boca Raton.

See Also

Other convergence: `converged_pars()`, `converged_terms()`, `converged()`, `esr_pars()`, `esr_terms()`, `rhat_pars()`, `rhat_terms()`, `rhat()`

esr.mcmc *Effective Sampling Rate*

Description

Calculates the effective sampling rate (`esr`).

Usage

```
## S3 method for class 'mcmc'
esr(x, by = "all", as_df = FALSE, na_rm = FALSE, ...)
```

Arguments

`x`	An object.
`by`	A string indicating whether to determine by "term", "parameter" or "all".
`as_df`	A flag indicating whether to return the values as a data frame versus a named list.
`na_rm`	A flag specifying whether to ignore missing values.
`...`	Other arguments passed to methods.

Details

By default

$$\frac{1}{1 + 2\sum_{k=1}^{\infty} \rho_k(\theta)}$$

from Brooks et al. (2011) where the infinite sum is truncated at lag k when $\rho_{k+1}(\theta) < 0$.

Value

A number between 0 and 1 indicating the esr value.

References

Brooks, S., Gelman, A., Jones, G.L., and Meng, X.-L. (Editors). 2011. Handbook for Markov Chain Monte Carlo. Taylor & Francis, Boca Raton.

See Also

Other convergence: `converged_pars()`, `converged_terms()`, `converged()`, `esr_pars()`, `esr_terms()`, `rhat_pars()`, `rhat_terms()`, `rhat()`

`esr.mcmc.list` *Effective Sampling Rate*

Description

Calculates the effective sampling rate (`esr`).

Usage

```
## S3 method for class 'mcmc.list'
esr(x, by = "all", as_df = FALSE, na_rm = FALSE, ...)
```

Arguments

`x`	An object.
`by`	A string indicating whether to determine by "term", "parameter" or "all".
`as_df`	A flag indicating whether to return the values as a data frame versus a named list.
`na_rm`	A flag specifying whether to ignore missing values.
`...`	Other arguments passed to methods.

Details

By default

$$\frac{1}{1 + 2\sum_{k=1}^{\infty} \rho_k(\theta)}$$

from Brooks et al. (2011) where the infinite sum is truncated at lag k when $\rho_{k+1}(\theta) < 0$.

Value

A number between 0 and 1 indicating the esr value.

References

Brooks, S., Gelman, A., Jones, G.L., and Meng, X.-L. (Editors). 2011. Handbook for Markov Chain Monte Carlo. Taylor & Francis, Boca Raton.

See Also

Other convergence: `converged_pars()`, `converged_terms()`, `converged()`, `esr_pars()`, `esr_terms()`, `rhat_pars()`, `rhat_terms()`, `rhat()`

`esr.mcmcarray` *Effective Sampling Rate*

Description

Calculates the effective sampling rate (`esr`).

Usage

```
## S3 method for class 'mcmcarray'
esr(x, by = "all", as_df = FALSE, na_rm = FALSE, ...)
```

Arguments

`x`	An object.
`by`	A string indicating whether to determine by "term", "parameter" or "all".
`as_df`	A flag indicating whether to return the values as a data frame versus a named list.
`na_rm`	A flag specifying whether to ignore missing values.
`...`	Other arguments passed to methods.

Details

By default

$$\frac{1}{1 + 2\sum_{k=1}^{\infty} \rho_k(\theta)}$$

from Brooks et al. (2011) where the infinite sum is truncated at lag k when $\rho_{k+1}(\theta) < 0$.

Value

A number between 0 and 1 indicating the esr value.

References

Brooks, S., Gelman, A., Jones, G.L., and Meng, X.-L. (Editors). 2011. Handbook for Markov Chain Monte Carlo. Taylor & Francis, Boca Raton.

See Also

Other convergence: `converged_pars()`, `converged_terms()`, `converged()`, `esr_pars()`, `esr_terms()`, `rhat_pars()`, `rhat_terms()`, `rhat()`

esr.mcmcr *Effective Sampling Rate*

Description

Calculates the effective sampling rate (esr).

Usage

```
## S3 method for class 'mcmcr'
esr(x, by = "all", as_df = FALSE, na_rm = FALSE, ...)
```

Arguments

x	An object.
by	A string indicating whether to determine by "term", "parameter" or "all".
as_df	A flag indicating whether to return the values as a data frame versus a named list.
na_rm	A flag specifying whether to ignore missing values.
...	Other arguments passed to methods.

Details

By default

$$\frac{1}{1 + 2\sum_{k=1}^{\infty} \rho_k(\theta)}$$

from Brooks et al. (2011) where the infinite sum is truncated at lag k when $\rho_{k+1}(\theta) < 0$.

Value

A number between 0 and 1 indicating the esr value.

References

Brooks, S., Gelman, A., Jones, G.L., and Meng, X.-L. (Editors). 2011. Handbook for Markov Chain Monte Carlo. Taylor & Francis, Boca Raton.

See Also

Other convergence: converged_pars(), converged_terms(), converged(), esr_pars(), esr_terms(), rhat_pars(), rhat_terms(), rhat()

Examples

```
esr(mcmcr_example)
```

`esr.mcmcrs` *Effective Sampling Rate*

Description

Calculates the effective sampling rate (`esr`).

Usage

```
## S3 method for class 'mcmcrs'
esr(x, by = "all", as_df = FALSE, na_rm = FALSE, ...)
```

Arguments

`x`	An object.
`by`	A string indicating whether to determine by "term", "parameter" or "all".
`as_df`	A flag indicating whether to return the values as a data frame versus a named list.
`na_rm`	A flag specifying whether to ignore missing values.
`...`	Other arguments passed to methods.

Details

By default

$$\frac{1}{1 + 2\sum_{k=1}^{\infty} \rho_k(\theta)}$$

from Brooks et al. (2011) where the infinite sum is truncated at lag k when $\rho_{k+1}(\theta) < 0$.

Value

A number between 0 and 1 indicating the esr value.

References

Brooks, S., Gelman, A., Jones, G.L., and Meng, X.-L. (Editors). 2011. Handbook for Markov Chain Monte Carlo. Taylor & Francis, Boca Raton.

See Also

Other convergence: `converged_pars()`, `converged_terms()`, `converged()`, `esr_pars()`, `esr_terms()`, `rhat_pars()`, `rhat_terms()`, `rhat()`

Examples

```
esr(mcmcrs(mcmcr_example, mcmcr_example))
```

ess *P-Value Effective Sample Size*

Description

Calculates the effective sample size based on `esr()`.

Usage

```
ess(x, by = "all", as_df = FALSE)
```

Arguments

`x`	An MCMC object.
`by`	A string indicating whether to determine by "term", "parameter" or "all".
`as_df`	A flag indicating whether to return the results as a data frame or list.

Examples

```
ess(mcmcr_example)
```

estimates.mcarray *Estimates*

Description

Calculates the estimates for an MCMC object.

Usage

```
## S3 method for class 'mcarray'
estimates(x, fun = median, as_df = FALSE, ...)
```

Arguments

`x`	An object.
`fun`	A function that given a numeric vector returns a numeric scalar.
`as_df`	A flag indicating whether to return the values as a data frame versus a named list.
`...`	Other arguments passed to methods.

Value

A named list or data frame.

See Also

Other MCMC manipulations: `bind_chains()`, `collapse_chains()`, `split_chains()`

Examples

```
library(nlist)

estimates(nlist(x = 1:9))
estimates(nlist(y = 3:5, zz = matrix(1:9, 3)))
estimates(nlists(nlist(x = 1:3), nlist(x = 2:4)), fun = mean)
```

---

estimates.mcmc *Estimates*

Description

Calculates the estimates for an MCMC object.

Usage

```
## S3 method for class 'mcmc'
estimates(x, fun = median, as_df = FALSE, ...)
```

Arguments

`x`	An object.
`fun`	A function that given a numeric vector returns a numeric scalar.
`as_df`	A flag indicating whether to return the values as a data frame versus a named list.
`...`	Other arguments passed to methods.

Value

A named list or data frame.

See Also

Other MCMC manipulations: `bind_chains()`, `collapse_chains()`, `split_chains()`

Examples

```
library(nlist)

estimates(nlist(x = 1:9))
estimates(nlist(y = 3:5, zz = matrix(1:9, 3)))
estimates(nlists(nlist(x = 1:3), nlist(x = 2:4)), fun = mean)
```

`estimates.mcmc.list` *Estimates*

Description

Calculates the estimates for an MCMC object.

Usage

```
## S3 method for class 'mcmc.list'
estimates(x, fun = median, as_df = FALSE, ...)
```

Arguments

`x`	An object.
`fun`	A function that given a numeric vector returns a numeric scalar.
`as_df`	A flag indicating whether to return the values as a data frame versus a named list.
`...`	Other arguments passed to methods.

Value

A named list or data frame.

See Also

Other MCMC manipulations: `bind_chains()`, `collapse_chains()`, `split_chains()`

Examples

```
library(nlist)

estimates(nlist(x = 1:9))
estimates(nlist(y = 3:5, zz = matrix(1:9, 3)))
estimates(nlists(nlist(x = 1:3), nlist(x = 2:4)), fun = mean)
```

`estimates.mcmcarray` *Estimates*

Description

Calculates the estimates for an MCMC object.

Usage

```
## S3 method for class 'mcmcarray'
estimates(x, fun = median, as_df = FALSE, ...)
```

Arguments

`x`	An object.
`fun`	A function that given a numeric vector returns a numeric scalar.
`as_df`	A flag indicating whether to return the values as a data frame versus a named list.
`...`	Unused.

Value

A named list or data frame.

See Also

Other MCMC manipulations: `bind_chains()`, `collapse_chains()`, `split_chains()`

Examples

```
library(nlist)

estimates(nlist(x = 1:9))
estimates(nlist(y = 3:5, zz = matrix(1:9, 3)))
estimates(nlists(nlist(x = 1:3), nlist(x = 2:4)), fun = mean)
```

`estimates.mcmcr` *Estimates*

Description

Calculates the estimates for an MCMC object.

Usage

```
## S3 method for class 'mcmcr'
estimates(x, fun = median, as_df = FALSE, ...)
```

Arguments

`x`	An object.
`fun`	A function that given a numeric vector returns a numeric scalar.
`as_df`	A flag indicating whether to return the values as a data frame versus a named list.
`...`	Other arguments passed to methods.

Value

A named list or data frame.

See Also

Other MCMC manipulations: `bind_chains()`, `collapse_chains()`, `split_chains()`

Examples

```
estimates(mcmcr_example)
```

`fill_all.mcarray` *Fill All Values*

Description

Fills all of an object's (missing and non-missing) values while preserving the object's dimensionality and class.

Usage

```
## S3 method for class 'mcarray'
fill_all(x, value = 0, nas = TRUE, ...)
```

Arguments

`x`	An object.
`value`	A scalar of the value to replace values with.
`nas`	A flag specifying whether to also fill missing values.
`...`	Other arguments passed to methods.

Value

The modified object.

Methods (by class)

- `logical`: Fill All for logical Objects
- `integer`: Fill All for integer Objects
- `numeric`: Fill All for numeric Objects
- `character`: Fill All for character Objects

See Also

Other fill: `fill_na`()

Examples

```
# logical
fill_all(c(TRUE, NA, FALSE))
fill_all(c(TRUE, NA, FALSE, nas = FALSE))
fill_all(c(TRUE, NA, FALSE, value = NA))

# integer
fill_all(matrix(1:4, nrow = 2), value = -1)

# numeric
fill_all(c(1, 4, NA), value = TRUE)
fill_all(c(1, 4, NA), value = TRUE, nas = FALSE)
```

```
# character
fill_all(c("some", "words"), value = TRUE)
```

`fill_all.mcmcarray` *Fill All Values*

Description

Fills all of an object's (missing and non-missing) values while preserving the object's dimensionality and class.

Usage

```
## S3 method for class 'mcmcarray'
fill_all(x, value = 0, nas = TRUE, ...)
```

Arguments

`x`	An object.
`value`	A scalar of the value to replace values with.
`nas`	A flag specifying whether to also fill missing values.
`...`	Other arguments passed to methods.

Value

The modified object.

Methods (by class)

- `logical`: Fill All for logical Objects
- `integer`: Fill All for integer Objects
- `numeric`: Fill All for numeric Objects
- `character`: Fill All for character Objects

See Also

Other fill: `fill_na()`

Examples

```
# logical
fill_all(c(TRUE, NA, FALSE))
fill_all(c(TRUE, NA, FALSE, nas = FALSE))
fill_all(c(TRUE, NA, FALSE, value = NA))

# integer
fill_all(matrix(1:4, nrow = 2), value = -1)

# numeric
fill_all(c(1, 4, NA), value = TRUE)
fill_all(c(1, 4, NA), value = TRUE, nas = FALSE)

# character
fill_all(c("some", "words"), value = TRUE)
```

`fill_all.mcmcr` *Fill All Values*

Description

Fills all of an object's (missing and non-missing) values while preserving the object's dimensionality and class.

Usage

```
## S3 method for class 'mcmcr'
fill_all(x, value = 0, nas = TRUE, ...)
```

Arguments

`x`	An object.
`value`	A scalar of the value to replace values with.
`nas`	A flag specifying whether to also fill missing values.
`...`	Other arguments passed to methods.

Value

The modified object.

Methods (by class)

- `logical`: Fill All for logical Objects
- `integer`: Fill All for integer Objects
- `numeric`: Fill All for numeric Objects
- `character`: Fill All for character Objects

See Also

Other fill: `fill_na()`

Examples

```
# logical
fill_all(c(TRUE, NA, FALSE))
fill_all(c(TRUE, NA, FALSE, nas = FALSE))
fill_all(c(TRUE, NA, FALSE, value = NA))

# integer
fill_all(matrix(1:4, nrow = 2), value = -1)

# numeric
```

```
fill_all(c(1, 4, NA), value = TRUE)
fill_all(c(1, 4, NA), value = TRUE, nas = FALSE)

# character
fill_all(c("some", "words"), value = TRUE)
```

`is.mcarray` *Is mcarray Object*

Description

Tests whether an object is an mcarray.

Usage

```
is.mcarray(x)
```

Arguments

`x`	The object to test.

Value

A flag indicating whether the test was positive.

Examples

```
is.mcarray(mcmcr_example)
```

`is.mcmcarray` *Is mcmcarray Object*

Description

Tests whether an object is an `mcmcarray-object()`.

Usage

```
is.mcmcarray(x)
```

Arguments

`x`	The object to test.

Value

A flag indicating whether the test was positive.

Examples

```
is.mcmcarray(mcmcr_example$beta)
```

`is.mcmcr` *Is mcmcr Object*

Description

Tests whether an object is an `mcmcr-object()`.

Usage

```
is.mcmcr(x)
```

Arguments

`x`	The object to test.

Value

A flag indicating whether the test was positive.

Examples

```
is.mcmcr(mcmcr_example)
```

`is.mcmcrs` *Is mcmcrs Object*

Description

Tests whether an object is an `mcmcrs-object()`.

Usage

```
is.mcmcrs(x)
```

Arguments

`x`	The object to test.

Value

A flag indicating whether the test was positive.

Examples

```
is.mcmcrs(mcmcrs(mcmcr_example))
```

`mcmcarray-object` *mcmcarray*

Description

An `mcmcarray` object is an an array where the first dimension is the chains, the second dimension is the iterations and the subsequent dimensions represent the dimensionality of the parameter. The name `mcmcarray` reflects the fact that the MCMC dimensions, ie the chains and iterations, precede the parameter dimensions.

Examples

```
mcmcr_example$beta
```

`mcmcr-object` *mcmcr*

Description

An `mcmcr` object stores multiple uniquely named `mcmcarray-object()` objects with the same number of chains and iterations.

Details

`mcmcr` objects allow a set of dimensionality preserving parameters to be manipulated and queried as a whole.

Examples

```
mcmcr_example
```

`mcmcrs` *Create mcmcrs*

Description

Creates an `mcmcrs-object()` from multiple `link{mcmcr-object}`s.

Usage

```
mcmcrs(...)
```

Arguments

`...`	Objects of class mcmcr.

Value

An object of class mcmcrs

Examples

```
mcmcrs(mcmcr_example, mcmcr_example)
```

`mcmcrs-object` *mcmcrs*

Description

An `mcmcrs` object stores multiple `mcmcr-object()`s with the same parameters and the same number of chains and iterations.

Details

`mcmcrs` objects allow the results of multiple analyses using the same model to be manipulated and queried as a whole.

Examples

```
mcmcrs(mcmcr_example, mcmcr_example)
```

`mcmcr_example` *An Example mcmcr Object*

Description

An example `mcmcr-object()` derived from coda::[line][coda::line].

Usage

```
mcmcr_example
```

Format

An object of class `mcmcr` of length 3.

Examples

```
mcmcr_example
```

`mcmc_aperm` *MCMC Object Transposition*

Description

Transpose an MCMC object by permuting its parameter dimensions.

Usage

```
mcmc_aperm(x, perm, ...)
```

Arguments

`x`	The MCMC object to transpose.
`perm`	A integer vector of the new order for the parameter dimensions. Missing parameter dimensions are added on the end. If perm = NULL (the default) the parameter dimensions are reversed.
`...`	Unused

Value

The modified MCMC object

`mcmc_map` *MCMC Map*

Description

Adjust the sample values of an MCMC object using a function.

Usage

```
mcmc_map(.x, .f, .by = 1:npdims(.x), ...)
```

Arguments

`.x`	An MCMC object
`.f`	The function to use
`.by`	A positive integer vector of the dimensions to apply the function over.
`...`	Additional arguments passed to .f.

Value

The updated MCMC object.

Examples

```
mcmc_map(mcmcr_example$beta, exp)
```

`nchains.mcarray` *Number of Chains*

Description

Gets the number of chains of an MCMC object.

Usage

```
## S3 method for class 'mcarray'
nchains(x, ...)
```

Arguments

`x`	An object.
`...`	Other arguments passed to methods.

Value

An integer scalar of the number of chains.

See Also

Other MCMC dimensions: `niters()`, `npars()`, `nsams()`, `nsims()`, `nterms()`

Examples

```
library(nlist)

nchains(nlist(x = 1:2))
nlists <- nlists(nlist(x = c(2, 9)), nlist(x = c(1, 7)))
nchains(nlists)
nchains(split_chains(nlists))
```

`nchains.mcmc` *Number of Chains*

Description

Gets the number of chains of an MCMC object.

Usage

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

Arguments

`x`	An object.
`...`	Other arguments passed to methods.

Value

An integer scalar of the number of chains.

See Also

Other MCMC dimensions: `niters()`, `npars()`, `nsams()`, `nsims()`, `nterms()`

Examples

```
library(nlist)

nchains(nlist(x = 1:2))
nlists <- nlists(nlist(x = c(2, 9)), nlist(x = c(1, 7)))
nchains(nlists)
nchains(split_chains(nlists))
```

`nchains.mcmc.list` *Number of Chains*

Description

Gets the number of chains of an MCMC object.

Usage

```
## S3 method for class 'mcmc.list'
nchains(x, ...)
```

Arguments

`x`	An object.
`...`	Other arguments passed to methods.

Value

An integer scalar of the number of chains.

See Also

Other MCMC dimensions: `niters()`, `npars()`, `nsams()`, `nsims()`, `nterms()`

Examples

```
library(nlist)

nchains(nlist(x = 1:2))
nlists <- nlists(nlist(x = c(2, 9)), nlist(x = c(1, 7)))
nchains(nlists)
nchains(split_chains(nlists))
```

`nchains.mcmcarray` *Number of Chains*

Description

Gets the number of chains of an MCMC object.

Usage

```
## S3 method for class 'mcmcarray'
nchains(x, ...)
```

Arguments

`x`	An object.
`...`	Other arguments passed to methods.

Value

An integer scalar of the number of chains.

See Also

Other MCMC dimensions: `niters()`, `npars()`, `nsams()`, `nsims()`, `nterms()`

Examples

```
library(nlist)

nchains(nlist(x = 1:2))
nlists <- nlists(nlist(x = c(2, 9)), nlist(x = c(1, 7)))
nchains(nlists)
nchains(split_chains(nlists))
```

`nchains.mcmcr` *Number of Chains*

Description

Gets the number of chains of an MCMC object.

Usage

```
## S3 method for class 'mcmcr'
nchains(x, ...)
```

Arguments

`x`	An object.
`...`	Other arguments passed to methods.

Value

An integer scalar of the number of chains.

See Also

Other MCMC dimensions: `niters()`, `npars()`, `nsams()`, `nsims()`, `nterms()`

Examples

```
library(nlist)

nchains(nlist(x = 1:2))
nlists <- nlists(nlist(x = c(2, 9)), nlist(x = c(1, 7)))
nchains(nlists)
nchains(split_chains(nlists))
```

`nchains.mcmcrs` *Number of Chains*

Description

Gets the number of chains of an MCMC object.

Usage

```
## S3 method for class 'mcmcrs'
nchains(x, ...)
```

Arguments

`x`	An object.
`...`	Other arguments passed to methods.

Value

An integer scalar of the number of chains.

See Also

Other MCMC dimensions: `niters()`, `npars()`, `nsams()`, `nsims()`, `nterms()`

Examples

```
library(nlist)

nchains(nlist(x = 1:2))
nlists <- nlists(nlist(x = c(2, 9)), nlist(x = c(1, 7)))
nchains(nlists)
nchains(split_chains(nlists))
```

niters.mcarray *Number of Iterations*

Description

Gets the number of iterations (in a chain) of an MCMC object.

Usage

```
## S3 method for class 'mcarray'
niters(x, ...)
```

Arguments

`x`	An object.
`...`	Other arguments passed to methods.

Value

An integer scalar of the number of iterations.

See Also

Other MCMC dimensions: `nchains()`, `npars()`, `nsams()`, `nsims()`, `nterms()`

Examples

```
library(nlist)

niters(nlist(x = 1:2))
niters(nlists(nlist(x = c(2, 9)), nlist(x = c(1, 7))))
```

niters.mcmc *Number of Iterations*

Description

Gets the number of iterations (in a chain) of an MCMC object.

Usage

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

Arguments

`x`	An object.
`...`	Other arguments passed to methods.

Value

An integer scalar of the number of iterations.

See Also

Other MCMC dimensions: `nchains()`, `npars()`, `nsams()`, `nsims()`, `nterms()`

Examples

```
library(nlist)

niters(nlist(x = 1:2))
niters(nlists(nlist(x = c(2, 9)), nlist(x = c(1, 7))))
```

`niters.mcmc.list` *Number of Iterations*

Description

Gets the number of iterations (in a chain) of an MCMC object.

Usage

```
## S3 method for class 'mcmc.list'
niters(x, ...)
```

Arguments

`x`	An object.
`...`	Other arguments passed to methods.

Value

An integer scalar of the number of iterations.

See Also

Other MCMC dimensions: `nchains()`, `npars()`, `nsams()`, `nsims()`, `nterms()`

Examples

```
library(nlist)

niters(nlist(x = 1:2))
niters(nlists(nlist(x = c(2, 9)), nlist(x = c(1, 7))))
```

`niters.mcmcarray` *Number of Iterations*

Description

Gets the number of iterations (in a chain) of an MCMC object.

Usage

```
## S3 method for class 'mcmcarray'
niters(x, ...)
```

Arguments

`x`	An object.
`...`	Other arguments passed to methods.

Value

An integer scalar of the number of iterations.

See Also

Other MCMC dimensions: `nchains()`, `npars()`, `nsams()`, `nsims()`, `nterms()`

Examples

```
library(nlist)

niters(nlist(x = 1:2))
niters(nlists(nlist(x = c(2, 9)), nlist(x = c(1, 7))))
```

`niters.mcmcr` *Number of Iterations*

Description

Gets the number of iterations (in a chain) of an MCMC object.

Usage

```
## S3 method for class 'mcmcr'
niters(x, ...)
```

Arguments

`x`	An object.
`...`	Other arguments passed to methods.

Value

An integer scalar of the number of iterations.

See Also

Other MCMC dimensions: `nchains()`, `npars()`, `nsams()`, `nsims()`, `nterms()`

Examples

```
library(nlist)

niters(nlist(x = 1:2))
niters(nlists(nlist(x = c(2, 9)), nlist(x = c(1, 7))))
```

`niters.mcmcrs` *Number of Iterations*

Description

Gets the number of iterations (in a chain) of an MCMC object.

Usage

```
## S3 method for class 'mcmcrs'
niters(x, ...)
```

Arguments

`x`	An object.
`...`	Other arguments passed to methods.

Value

An integer scalar of the number of iterations.

See Also

Other MCMC dimensions: `nchains()`, `npars()`, `nsams()`, `nsims()`, `nterms()`

Examples

```
library(nlist)

niters(nlist(x = 1:2))
niters(nlists(nlist(x = c(2, 9)), nlist(x = c(1, 7))))
```

`npars.mcarray` *Number of Parameters*

Description

Gets the number of parameters of an object.

The default methods returns the length of `pars()` if none are `NA`, otherwise it returns `NA`.

Usage

```
## S3 method for class 'mcarray'
npars(x, scalar = NULL, ...)
```

Arguments

`x`	An object.
`scalar`	A logical scalar specifying whether to include all parameters (NULL), only scalars (TRUE) or all parameters except scalars (FALSE).
`...`	Other arguments passed to methods.

Value

An integer scalar of the number of parameters.

See Also

`pars()`

Other MCMC dimensions: `nchains()`, `niters()`, `nsams()`, `nsims()`, `nterms()`

Other parameters: `pars()`, `set_pars()`

`npars.mcmcarray` *Number of Parameters*

Description

Gets the number of parameters of an object.

The default methods returns the length of `pars()` if none are `NA`, otherwise it returns `NA`.

Usage

```
## S3 method for class 'mcmcarray'
npars(x, scalar = NULL, ...)
```

Arguments

`x`	An object.
`scalar`	A logical scalar specifying whether to include all parameters (NULL), only scalars (TRUE) or all parameters except scalars (FALSE).
`...`	Other arguments passed to methods.

Value

An integer scalar of the number of parameters.

See Also

`pars()`

Other MCMC dimensions: `nchains()`, `niters()`, `nsams()`, `nsims()`, `nterms()`

Other parameters: `pars()`, `set_pars()`

`npars.mcmcr` *Number of Parameters*

Description

Gets the number of parameters of an object.

The default methods returns the length of `pars()` if none are `NA`, otherwise it returns `NA`.

Usage

```
## S3 method for class 'mcmcr'
npars(x, scalar = NULL, ...)
```

Arguments

`x`	An object.
`scalar`	A logical scalar specifying whether to include all parameters (NULL), only scalars (TRUE) or all parameters except scalars (FALSE).
`...`	Other arguments passed to methods.

Value

An integer scalar of the number of parameters.

See Also

`pars()`

Other MCMC dimensions: `nchains()`, `niters()`, `nsams()`, `nsims()`, `nterms()`

Other parameters: `pars()`, `set_pars()`

`npdims.mcmc.list` *Number of Parameter Dimensions*

Description

Gets the number of the dimensions of each parameter of an object.

The default methods returns the length of each element of `pdims()` as an integer vector.

Usage

```
## S3 method for class 'mcmc.list'
npdims(x, ...)
```

Arguments

`x`	An object.
`...`	Other arguments passed to methods.

Value

A named integer vector of the number of dimensions of each parameter.

See Also

Other dimensions: `dims()`, `ndims()`, `pdims()`

Examples

```
library(nlist)

npdims(nlist(x = 1:3))
npdims(nlist(y = 3, zz = matrix(2:5, 2)))
```

`npdims.mcmcarray` *Number of Parameter Dimensions*

Description

Gets the number of the dimensions of each parameter of an object.

The default methods returns the length of each element of `pdims()` as an integer vector.

Usage

```
## S3 method for class 'mcmcarray'
npdims(x, ...)
```

Arguments

`x`	An object.
`...`	Other arguments passed to methods.

Value

A named integer vector of the number of dimensions of each parameter.

See Also

Other dimensions: `dims()`, `ndims()`, `pdims()`

Examples

```
library(nlist)

npdims(nlist(x = 1:3))
npdims(nlist(y = 3, zz = matrix(2:5, 2)))
```

`npdims.mcmcr` *Number of Parameter Dimensions*

Description

Gets the number of the dimensions of each parameter of an object.

The default methods returns the length of each element of `pdims()` as an integer vector.

Usage

```
## S3 method for class 'mcmcr'
npdims(x, ...)
```

Arguments

`x`	An object.
`...`	Other arguments passed to methods.

Value

A named integer vector of the number of dimensions of each parameter.

See Also

Other dimensions: `dims()`, `ndims()`, `pdims()`

Examples

```
library(nlist)

npdims(nlist(x = 1:3))
npdims(nlist(y = 3, zz = matrix(2:5, 2)))
```

`nterms.mcmc` *Number of Terms*

Description

Gets the number of terms of an MCMC object.

Usage

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

Arguments

`x`	An object.
`...`	Other arguments passed to methods.

Value

A integer scalar of the number of terms.

See Also

Other MCMC dimensions: `nchains()`, `niters()`, `npars()`, `nsams()`, `nsims()`

Examples

```
library(nlist)

nterms(nlist(x = 2))
nterms(nlist(x = NA_real_))
nterms(nlist(x = 3, zz = matrix(2:5, 2)))
nterms(nlists(
  nlist(y = 3, zz = matrix(2:5, 2)),
  nlist(y = 5, zz = matrix(1:4, 2))
))
```

`nterms.mcmc.list` *Number of Terms*

Description

Gets the number of terms of an MCMC object.

Usage

```
## S3 method for class 'mcmc.list'
nterms(x, ...)
```

Arguments

`x`	An object.
`...`	Other arguments passed to methods.

Value

A integer scalar of the number of terms.

See Also

Other MCMC dimensions: `nchains()`, `niters()`, `npars()`, `nsams()`, `nsims()`

Examples

```
library(nlist)

nterms(nlist(x = 2))
nterms(nlist(x = NA_real_))
nterms(nlist(x = 3, zz = matrix(2:5, 2)))
nterms(nlists(
  nlist(y = 3, zz = matrix(2:5, 2)),
  nlist(y = 5, zz = matrix(1:4, 2))
))
```

`nterms.mcmcarray` *Number of Terms*

Description

Gets the number of terms of an MCMC object.

Usage

```
## S3 method for class 'mcmcarray'
nterms(x, ...)
```

Arguments

`x`	An object.
`...`	Other arguments passed to methods.

Value

A integer scalar of the number of terms.

See Also

Other MCMC dimensions: `nchains()`, `niters()`, `npars()`, `nsams()`, `nsims()`

Examples

```
library(nlist)

nterms(nlist(x = 2))
nterms(nlist(x = NA_real_))
nterms(nlist(x = 3, zz = matrix(2:5, 2)))
nterms(nlists(
  nlist(y = 3, zz = matrix(2:5, 2)),
  nlist(y = 5, zz = matrix(1:4, 2))
))
```

`nterms.mcmcr` *Number of Terms*

Description

Gets the number of terms of an MCMC object.

Usage

```
## S3 method for class 'mcmcr'
nterms(x, ...)
```

Arguments

`x`	An object.
`...`	Other arguments passed to methods.

Value

A integer scalar of the number of terms.

See Also

Other MCMC dimensions: `nchains()`, `niters()`, `npars()`, `nsams()`, `nsims()`

Examples

```
library(nlist)

nterms(nlist(x = 2))
nterms(nlist(x = NA_real_))
nterms(nlist(x = 3, zz = matrix(2:5, 2)))
nterms(nlists(
  nlist(y = 3, zz = matrix(2:5, 2)),
  nlist(y = 5, zz = matrix(1:4, 2))
))
```

`nterms.mcmcrs` *Number of Terms*

Description

Gets the number of terms of an MCMC object.

Usage

```
## S3 method for class 'mcmcrs'
nterms(x, ...)
```

Arguments

`x`	An object.
`...`	Other arguments passed to methods.

Value

A integer scalar of the number of terms.

See Also

Other MCMC dimensions: `nchains()`, `niters()`, `npars()`, `nsams()`, `nsims()`

Examples

```
library(nlist)

nterms(nlist(x = 2))
nterms(nlist(x = NA_real_))
nterms(nlist(x = 3, zz = matrix(2:5, 2)))
nterms(nlists(
  nlist(y = 3, zz = matrix(2:5, 2)),
  nlist(y = 5, zz = matrix(1:4, 2))
))
```

`parameters` *Parameter Names*

Description

Gets the parameter names.

Usage

```
parameters(x, ...)

## S3 method for class 'mcmc'
parameters(x, scalar = NULL, terms = FALSE, ...)

## S3 method for class 'mcmc.list'
parameters(x, scalar = NULL, terms = FALSE, ...)

## S3 method for class 'mcmcr'
parameters(x, scalar = NULL, terms = FALSE, ...)

## S3 method for class 'mcmcrs'
parameters(x, scalar = NULL, terms = FALSE, ...)
```

Arguments

`x`	An object.
`...`	Unused.
`scalar`	A logical scalar specifying whether to include all parameters (NULL), only scalars (TRUE) or all parameters except scalars (FALSE).
`terms`	A logical scalar specifying whether to provide the parameters for each term.

Details

Soft-deprecated for pars()

Value

A character vector of the names of the parameters.

Methods (by class)

- `mcmc`: Parameters mcmc
- `mcmc.list`: Parameters mcmc.list
- `mcmcr`: Parameters mcmcr
- `mcmcrs`: Parameters mcmcrs

`parameters<-` *Set Parameters*

Description

Sets an object's parameter names.

Usage

```
parameters(x) <- value
```

Arguments

`x`	An object.
`value`	A character vector of the new parameter names.

Details

`value` must be a unique character vector of the same length as the object's parameters.

Soft-deprecated for pars<-()

Value

The modified object.

`params` *Parameter Descriptions*

Description

Parameter Descriptions

Arguments

`x`	An object.
`scalar`	A logical scalar specifying whether to include all parameters (NULL), only scalars (TRUE) or all parameters except scalars (FALSE).
`terms`	A logical scalar specifying whether to provide the parameters for each term.
`nas`	A flag specifying whether to also fill missing values.
`nthin`	A positive integer of the thinning rate.
`by`	A string indicating whether to determine by "term", "parameter" or "all".
`as_df`	A flag indicating whether to return the values as a data frame versus a named list.
`fun`	A function that given a numeric vector returns a numeric scalar.
`bound`	flag specifying whether to bind mcmcrs objects by their chains before calculating rhat.
`rhat`	The maximum rhat value.
`esr`	The minimum effective sampling rate.
`na_rm`	A flag specifying whether to ignore missing values.
`parameters`	A character vector (or NULL) of the parameters to subset by.
`iterations`	An integer vector (or NULL) of the iterations to subset by.
`...`	Unused.

pars.mcmc *Parameter Names*

Description

Gets the parameter names.

Usage

```
## S3 method for class 'mcmc'
pars(x, scalar = NULL, terms = FALSE, ...)
```

Arguments

`x`	An object.
`scalar`	A logical scalar specifying whether to include all parameters (NULL), only scalars (TRUE) or all parameters except scalars (FALSE).
`terms`	A logical scalar specifying whether to provide the parameters for each term.
`...`	Other arguments passed to methods.

Value

A character vector of the names of the parameters.

See Also

Other parameters: `npars()`, `set_pars()`

Examples

```
library(nlist)

pars(nlist(zz = 1, y = 3:6))
```

pars.mcmc.list *Parameter Names*

Description

Gets the parameter names.

Usage

```
## S3 method for class 'mcmc.list'
pars(x, scalar = NULL, terms = FALSE, ...)
```

Arguments

`x`	An object.
`scalar`	A logical scalar specifying whether to include all parameters (NULL), only scalars (TRUE) or all parameters except scalars (FALSE).
`terms`	A logical scalar specifying whether to provide the parameters for each term.
`...`	Other arguments passed to methods.

Value

A character vector of the names of the parameters.

See Also

Other parameters: `npars()`, `set_pars()`

Examples

```
library(nlist)

pars(nlist(zz = 1, y = 3:6))
```

`pars.mcmcr` *Parameter Names*

Description

Gets the parameter names.

Usage

```
## S3 method for class 'mcmcr'
pars(x, scalar = NULL, terms = FALSE, ...)
```

Arguments

`x`	An object.
`scalar`	A logical scalar specifying whether to include all parameters (NULL), only scalars (TRUE) or all parameters except scalars (FALSE).
`terms`	A logical scalar specifying whether to provide the parameters for each term.
`...`	Other arguments passed to methods.

Value

A character vector of the names of the parameters.

See Also

Other parameters: `npars()`, `set_pars()`

Examples

```
library(nlist)

pars(nlist(zz = 1, y = 3:6))
```

`pars.mcmcrs` *Parameter Names*

Description

Gets the parameter names.

Usage

```
## S3 method for class 'mcmcrs'
pars(x, scalar = NULL, terms = FALSE, ...)
```

Arguments

`x`	An object.
`scalar`	A logical scalar specifying whether to include all parameters (NULL), only scalars (TRUE) or all parameters except scalars (FALSE).
`terms`	A logical scalar specifying whether to provide the parameters for each term.
`...`	Other arguments passed to methods.

Value

A character vector of the names of the parameters.

See Also

Other parameters: `npars()`, `set_pars()`

Examples

```
library(nlist)

pars(nlist(zz = 1, y = 3:6))
```

`pdims.mcarray` *Parameter Dimensions*

Description

Gets the dimensions of each parameter of an object.

Usage

```
## S3 method for class 'mcarray'
pdims(x, ...)
```

Arguments

`x`	An object.
`...`	Other arguments passed to methods.

Value

A named list of integer vectors of the dimensions of each parameter.

See Also

Other dimensions: `dims()`, `ndims()`, `npdims()`

Examples

```
library(nlist)

pdims(nlist(x = 1:3))
pdims(nlist(y = 3, zz = matrix(2:5, 2)))
```

`pdims.mcmc` *Parameter Dimensions*

Description

Gets the dimensions of each parameter of an object.

Usage

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

Arguments

`x`	An object.
`...`	Other arguments passed to methods.

Value

A named list of integer vectors of the dimensions of each parameter.

See Also

Other dimensions: `dims()`, `ndims()`, `npdims()`

Examples

```
library(nlist)

pdims(nlist(x = 1:3))
pdims(nlist(y = 3, zz = matrix(2:5, 2)))
```

`pdims.mcmc.list` *Parameter Dimensions*

Description

Gets the dimensions of each parameter of an object.

Usage

```
## S3 method for class 'mcmc.list'
pdims(x, ...)
```

Arguments

`x`	An object.
`...`	Other arguments passed to methods.

Value

A named list of integer vectors of the dimensions of each parameter.

See Also

Other dimensions: `dims()`, `ndims()`, `npdims()`

Examples

```
library(nlist)

pdims(nlist(x = 1:3))
pdims(nlist(y = 3, zz = matrix(2:5, 2)))
```

`pdims.mcmcarray` *Parameter Dimensions*

Description

Gets the dimensions of each parameter of an object.

Usage

```
## S3 method for class 'mcmcarray'
pdims(x, ...)
```

Arguments

`x`	An object.
`...`	Other arguments passed to methods.

Value

A named list of integer vectors of the dimensions of each parameter.

See Also

Other dimensions: `dims()`, `ndims()`, `npdims()`

Examples

```
library(nlist)

pdims(nlist(x = 1:3))
pdims(nlist(y = 3, zz = matrix(2:5, 2)))
```

`pdims.mcmcr` *Parameter Dimensions*

Description

Gets the dimensions of each parameter of an object.

Usage

```
## S3 method for class 'mcmcr'
pdims(x, ...)
```

Arguments

`x`	An object.
`...`	Other arguments passed to methods.

Value

A named list of integer vectors of the dimensions of each parameter.

See Also

Other dimensions: `dims()`, `ndims()`, `npdims()`

Examples

```
library(nlist)

pdims(nlist(x = 1:3))
pdims(nlist(y = 3, zz = matrix(2:5, 2)))
```

`rhat.mcarray` *R-hat*

Description

Calculates an R-hat (potential scale reduction factor) value.

Usage

```
## S3 method for class 'mcarray'
rhat(x, by = "all", as_df = FALSE, na_rm = FALSE, ...)
```

Arguments

`x`	An object.
`by`	A string indicating whether to determine by "term", "parameter" or "all".
`as_df`	A flag indicating whether to return the values as a data frame versus a named list.
`na_rm`	A flag specifying whether to ignore missing values.
`...`	Other arguments passed to methods.

Details

By default the uncorrected, unfolded, univariate, split R-hat value.

Value

A number >= 1 indicating the rhat value.

References

Gelman, A., and Rubin, D.B. 1992. Inference from Iterative Simulation Using Multiple Sequences. Statistical Science 7(4): 457–472.

See Also

Other convergence: `converged_pars()`, `converged_terms()`, `converged()`, `esr_pars()`, `esr_terms()`, `esr()`, `rhat_pars()`, `rhat_terms()`

rhat.mcmc *R-hat*

Description

Calculates an R-hat (potential scale reduction factor) value.

Usage

```
## S3 method for class 'mcmc'
rhat(x, by = "all", as_df = FALSE, na_rm = FALSE, ...)
```

Arguments

`x`	An object.
`by`	A string indicating whether to determine by "term", "parameter" or "all".
`as_df`	A flag indicating whether to return the values as a data frame versus a named list.
`na_rm`	A flag specifying whether to ignore missing values.
`...`	Other arguments passed to methods.

Details

By default the uncorrected, unfolded, univariate, split R-hat value.

Value

A number >= 1 indicating the rhat value.

References

Gelman, A., and Rubin, D.B. 1992. Inference from Iterative Simulation Using Multiple Sequences. Statistical Science 7(4): 457–472.

See Also

Other convergence: `converged_pars()`, `converged_terms()`, `converged()`, `esr_pars()`, `esr_terms()`, `esr()`, `rhat_pars()`, `rhat_terms()`

rhat.mcmc.list *R-hat*

Description

Calculates an R-hat (potential scale reduction factor) value.

Usage

```
## S3 method for class 'mcmc.list'
rhat(x, by = "all", as_df = FALSE, na_rm = FALSE, ...)
```

Arguments

x	An object.
by	A string indicating whether to determine by "term", "parameter" or "all".
as_df	A flag indicating whether to return the values as a data frame versus a named list.
na_rm	A flag specifying whether to ignore missing values.
...	Other arguments passed to methods.

Details

By default the uncorrected, unfolded, univariate, split R-hat value.

Value

A number >= 1 indicating the rhat value.

References

Gelman, A., and Rubin, D.B. 1992. Inference from Iterative Simulation Using Multiple Sequences. Statistical Science 7(4): 457–472.

See Also

Other convergence: `converged_pars()`, `converged_terms()`, `converged()`, `esr_pars()`, `esr_terms()`, `esr()`, `rhat_pars()`, `rhat_terms()`

`rhat.mcmcarray` *R-hat*

Description

Calculates an R-hat (potential scale reduction factor) value.

Usage

```
## S3 method for class 'mcmcarray'
rhat(x, by = "all", as_df = FALSE, na_rm = FALSE, ...)
```

Arguments

`x`	An object.
`by`	A string indicating whether to determine by "term", "parameter" or "all".
`as_df`	A flag indicating whether to return the values as a data frame versus a named list.
`na_rm`	A flag specifying whether to ignore missing values.
`...`	Other arguments passed to methods.

Details

By default the uncorrected, unfolded, univariate, split R-hat value.

Value

A number >= 1 indicating the rhat value.

References

Gelman, A., and Rubin, D.B. 1992. Inference from Iterative Simulation Using Multiple Sequences. Statistical Science 7(4): 457–472.

See Also

Other convergence: `converged_pars()`, `converged_terms()`, `converged()`, `esr_pars()`, `esr_terms()`, `esr()`, `rhat_pars()`, `rhat_terms()`

rhat.mcmcr *R-hat*

Description

Calculates an R-hat (potential scale reduction factor) value.

Usage

```
## S3 method for class 'mcmcr'
rhat(x, by = "all", as_df = FALSE, na_rm = FALSE, ...)
```

Arguments

x	An object.
by	A string indicating whether to determine by "term", "parameter" or "all".
as_df	A flag indicating whether to return the values as a data frame versus a named list.
na_rm	A flag specifying whether to ignore missing values.
...	Other arguments passed to methods.

Details

By default the uncorrected, unfolded, univariate, split R-hat value.

Value

A number >= 1 indicating the rhat value.

References

Gelman, A., and Rubin, D.B. 1992. Inference from Iterative Simulation Using Multiple Sequences. Statistical Science 7(4): 457–472.

See Also

Other convergence: `converged_pars()`, `converged_terms()`, `converged()`, `esr_pars()`, `esr_terms()`, `esr()`, `rhat_pars()`, `rhat_terms()`

Examples

```
rhat(mcmcr_example)
rhat(mcmcr_example, by = "parameter")
rhat(mcmcr_example, by = "term")
rhat(mcmcr_example, by = "term", as_df = TRUE)
```

rhat.mcmcrs *R-hat*

Description

Calculates an R-hat (potential scale reduction factor) value.

Usage

```
## S3 method for class 'mcmcrs'
rhat(x, by = "all", as_df = FALSE, na_rm = FALSE, bound = FALSE, ...)
```

Arguments

x	An object.
by	A string indicating whether to determine by "term", "parameter" or "all".
as_df	A flag indicating whether to return the values as a data frame versus a named list.
na_rm	A flag specifying whether to ignore missing values.
bound	flag specifying whether to bind mcmcrs objects by their chains before calculating rhat.
...	Other arguments passed to methods.

Details

By default the uncorrected, unfolded, univariate, split R-hat value.

Value

A number >= 1 indicating the rhat value.

References

Gelman, A., and Rubin, D.B. 1992. Inference from Iterative Simulation Using Multiple Sequences. Statistical Science 7(4): 457–472.

See Also

Other convergence: `converged_pars()`, `converged_terms()`, `converged()`, `esr_pars()`, `esr_terms()`, `esr()`, `rhat_pars()`, `rhat_terms()`

Examples

```
rhat(mcmcrs(mcmcr_example, mcmcr_example))
rhat(mcmcrs(mcmcr_example, mcmcr_example), bound = TRUE)
```

`set_pars.mcmc` *Set Parameters*

Description

Sets an object's parameter names.

The assignment version pars<-() forwards to `set_pars()`.

Usage

```
## S3 method for class 'mcmc'
set_pars(x, value, ...)
```

Arguments

`x`	An object.
`value`	A character vector of the new parameter names.
`...`	Other arguments passed to methods.

Details

`value` must be a unique character vector of the same length as the object's parameters.

Value

The modified object.

See Also

Other parameters: `npars()`, `pars()`

Examples

```
library(nlist)

nlist <-  nlist(x = 1, y = 3:4)
pars(nlist) <- c("a", "b")
nlist
set_pars(nlist, c("z", "c1"))
```

`set_pars.mcmc.list` *Set Parameters*

Description

Sets an object's parameter names.

The assignment version pars<-() forwards to `set_pars()`.

Usage

```
## S3 method for class 'mcmc.list'
set_pars(x, value, ...)
```

Arguments

`x`	An object.
`value`	A character vector of the new parameter names.
`...`	Other arguments passed to methods.

Details

`value` must be a unique character vector of the same length as the object's parameters.

Value

The modified object.

See Also

Other parameters: `npars()`, `pars()`

Examples

```
library(nlist)

nlist <-  nlist(x = 1, y = 3:4)
pars(nlist) <- c("a", "b")
nlist
set_pars(nlist, c("z", "c1"))
```

set_pars.mcmcr *Set Parameters*

Description

Sets an object's parameter names.

The assignment version pars<-() forwards to `set_pars()`.

Usage

```
## S3 method for class 'mcmcr'
set_pars(x, value, ...)
```

Arguments

`x`	An object.
`value`	A character vector of the new parameter names.
`...`	Other arguments passed to methods.

Details

`value` must be a unique character vector of the same length as the object's parameters.

Value

The modified object.

See Also

Other parameters: `npars()`, `pars()`

Examples

```
library(nlist)

nlist <-  nlist(x = 1, y = 3:4)
pars(nlist) <- c("a", "b")
nlist
set_pars(nlist, c("z", "c1"))
```

set_pars.mcmcrs *Set Parameters*

Description

Sets an object's parameter names.

The assignment version pars<-() forwards to `set_pars()`.

Usage

```
## S3 method for class 'mcmcrs'
set_pars(x, value, ...)
```

Arguments

`x`	An object.
`value`	A character vector of the new parameter names.
`...`	Other arguments passed to methods.

Details

`value` must be a unique character vector of the same length as the object's parameters.

Value

The modified object.

See Also

Other parameters: `npars()`, `pars()`

Examples

```
library(nlist)

nlist <-  nlist(x = 1, y = 3:4)
pars(nlist) <- c("a", "b")
nlist
set_pars(nlist, c("z", "c1"))
```

split_chains.mcmcarray *Split Chains*

Description

Splits each of an MCMC object's chains in half to double the number of chains and halve the number of iterations.

Usage

```
## S3 method for class 'mcmcarray'
split_chains(x, ...)
```

Arguments

x	An object.
...	Other arguments passed to methods.

Value

The modified object.

See Also

Other MCMC manipulations: bind_chains(), collapse_chains(), estimates()

Examples

```
library(nlist)

nlists <- nlists(nlist(x = c(2, 9)), nlist(x = c(1, 7)))
nchains(nlists)
nchains(split_chains(nlists))
```

split_chains.mcmcr *Split Chains*

Description

Splits each of an MCMC object's chains in half to double the number of chains and halve the number of iterations.

Usage

```
## S3 method for class 'mcmcr'
split_chains(x, ...)
```

Arguments

`x`	An object.
`...`	Other arguments passed to methods.

Value

The modified object.

See Also

Other MCMC manipulations: `bind_chains()`, `collapse_chains()`, `estimates()`

Examples

```
library(nlist)

nlists <- nlists(nlist(x = c(2, 9)), nlist(x = c(1, 7)))
nchains(nlists)
nchains(split_chains(nlists))
```

---

subset *Subset an MCMC Object*

Description

Subsets an MCMC object by its chains, iterations and/or parameters.

Usage

```
## S3 method for class 'mcmc'
subset(x, iters = NULL, pars = NULL, iterations = NULL, parameters = NULL, ...)

## S3 method for class 'mcmc.list'
subset(
  x,
  chains = NULL,
  iters = NULL,
  pars = NULL,
  iterations = NULL,
  parameters = NULL,
  ...
)

## S3 method for class 'mcmcarray'
subset(x, chains = NULL, iters = NULL, iterations = NULL, ...)

## S3 method for class 'mcmcr'
subset(
  x,
  chains = NULL,
  iters = NULL,
  pars = NULL,
  iterations = NULL,
  parameters = NULL,
  ...
)

## S3 method for class 'mcmcrs'
subset(
  x,
  chains = NULL,
  iters = NULL,
  pars = NULL,
  iterations = NULL,
  parameters = NULL,
  ...
)
```

Arguments

`x`	The MCMC object to subset
`iters`	An integer vector of iterations.
`pars`	A character vector of parameter names.
`iterations`	An integer vector (or NULL) of the iterations to subset by.
`parameters`	A character vector (or NULL) of the parameters to subset by.
`...`	Unused.
`chains`	An integer vector of chains.

Methods (by class)

- `mcmc`: Subset an mcmc object
- `mcmc.list`: Subset an mcmc.list object
- `mcmcarray`: Subset an mcmcarray object
- `mcmcr`: Subset an mcmcr object
- `mcmcrs`: Subset an mcmcrs object

Examples

```
subset(mcmcr_example,
  chains = 2L, iters = 1:100,
  pars = c("beta", "alpha")
)
```

tidy.mcmc *Turn an object into a tidy tibble*

Description

Turn an object into a tidy tibble

Usage

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

Arguments

`x`	An object to be converted into a tidy `tibble::tibble()`.
`...`	Additional arguments to tidying method.

Value

A `tibble::tibble()` with information about model components.

Methods

No methods found in currently loaded packages.

tidy.mcmc.list *Turn an object into a tidy tibble*

Description

Turn an object into a tidy tibble

Usage

```
## S3 method for class 'mcmc.list'
tidy(x, ...)
```

Arguments

`x`	An object to be converted into a tidy `tibble::tibble()`.
`...`	Additional arguments to tidying method.

Value

A `tibble::tibble()` with information about model components.

Methods

No methods found in currently loaded packages.

tidy.mcmc.mcmcr *Turn an object into a tidy tibble*

Description

Turn an object into a tidy tibble

Usage

```
## S3 method for class 'mcmc.mcmcr'
tidy(x, ...)
```

Arguments

`x`	An object to be converted into a tidy `tibble::tibble()`.
`...`	Additional arguments to tidying method.

Value

A `tibble::tibble()` with information about model components.

Methods

No methods found in currently loaded packages.

vld_mcmcr *Validate MCMC Objects*

Description

Validates class and structure of MCMC objects.

Usage

```
vld_mcmcarray(x)

vld_mcmcr(x)

vld_mcmcrs(x)
```

Arguments

`x`	The object to check.

Details

To just validate class use `chk::vld_s3_class()`.

Value

A flag indicating whether the object was validated.

Functions

- `vld_mcmcarray`: Validate `mcmcarray-object()`
- `vld_mcmcr`: Validate `mcmcr-object()`
- `vld_mcmcrs`: Validate `mcmcrs-object()`

See Also

`chk_mcmcr()`

Examples

```
#' vld_mcmcarray
vld_mcmcarray(1)

# vld_mcmcr
vld_mcmcr(1)
vld_mcmcr(mcmcr::mcmcr_example)

# vld_mcmcrs
vld_mcmcrs(1)
```

`zero` *Zero MCMC Sample Values*

Description

Zeros an MCMC object's sample values.

Usage

```
zero(x, ...)

## S3 method for class 'mcarray'
zero(x, ...)

## S3 method for class 'mcmcarray'
zero(x, ...)

## S3 method for class 'mcmcr'
zero(x, pars = NULL, ...)
```

Arguments

`x`	The MCMC object.
`...`	Unused
`pars`	A character vector (or NULL) of the pars to zero.

Details

It is used for removing the effect of a random effect where the expected value is 0.

Value

The MCMC

Methods (by class)

- `mcarray`: Zero an mcarray object
- `mcmcarray`: Zero an mcmcarray object
- `mcmcr`: Zero an mcmcr object

Examples

```
zero(mcmcr_example, pars = "beta")
```

Index