Package ‘universals’

July 8, 2020

Title S3 Generics for Bayesian Analyses

Version 0.0.3

Description Provides S3 generic methods and some default implementations
for Bayesian analyses that generate Markov Chain Monte Carlo (MCMC) samples.
The purpose of 'universals' is to reduce package dependencies and conflicts.
The 'nlist' package implements many of the methods for its 'nlist' class.

License MIT + file LICENSE

URL `https://github.com/poissonconsulting/universals`

BugReports `https://github.com/poissonconsulting/universals/issues`

Depends R (>= 3.4)

Suggests covr, nlist, testthat

Encoding UTF-8

Language en-US

LazyData true

RoxygenNote 7.1.1.9000

NeedsCompilation no

Author Joe Thorley [aut, cre] (<https://orcid.org/0000-0002-7683-4592>),
Kirill Müller [ctb] (<https://orcid.org/0000-0002-1416-3412>),
Poisson Consulting [cph, fnd]

Maintainer Joe Thorley <`joe@poissonconsulting.ca`>

Repository CRAN

Date/Publication 2020-07-08 06:20:06 UTC

R topics documented:

`bind_chains` *Bind by Chains.*

Description

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

Usage

```
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()`

`collapse_chains` *Collapse Chains*

Description

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

Usage

```
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()`

`converged` *Converged*

Description

Tests whether an object has converged.

Usage

```
converged(x, ...)
```

Arguments

`x`	An object.
`...`	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()`

`converged_pars` *Converged Parameters*

Description

Tests whether each parameter of an object has converged.

Usage

```
converged_pars(x, ...)
```

Arguments

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

Value

A uniquely named logical vector indicating whether each parameter has converged.

See Also

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

`converged_terms` *Converged Terms*

Description

Tests whether each term of an object has converged.

Usage

```
converged_terms(x, ...)
```

Arguments

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

Value

A list of uniquely named logical objects with whether each term has converged.

See Also

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

`dims` *Dimensions*

Description

Gets the dimensions of an object.

Usage

```
dims(x, ...)

## Default S3 method:
dims(x, ...)

## S3 method for class 'factor'
dims(x, ...)
```

Arguments

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

Details

Unlike `base::dim()`, dims returns the length of an atomic vector.

Value

An integer vector of the dimensions.

See Also

`base::dim()`

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

Examples

```
dims(numeric(0))
dims(1:3)
dims(factor("a"))
dims(matrix(1:4, nrow = 2L))
dims(array(1:9, dim = c(3L,1L,3L)))
dims(ToothGrowth)
dims(Titanic)
```

esr *Effective Sampling Rate*

Description

Calculates the effective sampling rate (`esr`).

Usage

```
esr(x, ...)
```

Arguments

`x`	An object.
`...`	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_pars` *Effective Sampling Rate for Parameters*

Description

Calculates the effective sampling rate (`esr`) for each parameter.

Usage

```
esr_pars(x, ...)
```

Arguments

`x`	An object.
`...`	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 uniquely named numeric vector of values between 0 and 1 indicating the esr value for each parameter.

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_terms()`, `esr()`, `rhat_pars()`, `rhat_terms()`, `rhat()`

`esr_terms` *Effective Sampling Rate for Terms*

Description

Calculates the effective sampling rate (`esr`) for each term.

Usage

```
esr_terms(x, ...)
```

Arguments

`x`	An object.
`...`	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 list of uniquely named numeric objects with values between 0 and 1 indicating the effectively sampling rate for each term.

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()`, `rhat_pars()`, `rhat_terms()`, `rhat()`

`estimates` *Estimates*

Description

Calculates the estimates for an MCMC object.

Usage

```
estimates(x, ...)
```

Arguments

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

Value

A list of uniquely named numeric objects.

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)
```

`nchains` *Number of Chains*

Description

Gets the number of chains of an MCMC object.

Usage

```
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))
```

`ndims` *Number of Dimensions*

Description

Gets the number of dimensions of an object as returned by `dims()`.

The default methods returns the length of `dims()`.

Usage

```
ndims(x, ...)

## Default S3 method:
ndims(x, ...)

## S3 method for class 'matrix'
ndims(x, ...)

## S3 method for class 'data.frame'
ndims(x, ...)
```

Arguments

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

Details

For matrices `ndims()` is always 2L.

For data frames `ndims()` is always 2L.

Value

A integer scalar of the number of dimensions.

See Also

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

Examples

```
ndims(character(0))
ndims(1:3)
ndims(matrix(1))
ndims(data.frame())
ndims(array(1:9, dim = c(3,1,3)))
```

`niters` *Number of Iterations*

Description

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

Usage

```
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` *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

```
npars(x, ...)

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

Arguments

`x`	An object.
`...`	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 — *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

```
npdims(x, ...)

## Default S3 method:
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)))
```

nsams — *Number of Samples*

Description

Gets the number of sample values (simulations * terms) of an MCMC object.

The default methods returns the product of `nsims()` and `nterms()`.

Usage

```
nsams(x, ...)

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

Arguments

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

Value

An integer scalar of the number of samples.

See Also

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

Examples

```
library(nlist)

nsams(nlist())
nsams(nlist(x = 1))
nsams(nlist(x = 2:3))
nlist <- nlist(x = 2:3, y = matrix(1:9))
nsams(nlist)
nsams(nlists(nlist, nlist))
```

nsims *Number of Simulations*

Description

Gets the number of simulations (iterations * chains) of an MCMC object.

The default methods returns the product of `nchains()` and `niters()`.

Usage

```
nsims(x, ...)

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

Arguments

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

Value

An integer scalar of the number of simulations.

See Also

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

Examples

```
library(nlist)

nsims(nlists())
nsims(nlists(nlist()))
```

`nterms` *Number of Terms*

Description

Gets the number of terms of an MCMC object.

Usage

```
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))
))
```

`pars` *Parameter Names*

Description

Gets the parameter names.

Usage

```
pars(x, ...)
```

Arguments

`x`	An object.
`...`	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` *Parameter Dimensions*

Description

Gets the dimensions of each parameter of an object.

Usage

```
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` *R-hat*

Description

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

Usage

```
rhat(x, ...)
```

Arguments

`x`	An object.
`...`	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_pars` *R-hat Parameters*

Description

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

Usage

```
rhat_pars(x, ...)
```

Arguments

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

Details

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

Value

An uniquely named numeric atomic with values >= 1 indicating the rhat value for each parameter.

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_terms()`, `rhat()`

`rhat_terms` *R-hat Terms*

Description

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

Usage

```
rhat_terms(x, ...)
```

Arguments

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

Details

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

Value

A list of uniquely named numeric objects with values >= 1 indicating the rhat value for each term.

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()`

`set_pars` *Set Parameters*

Description

Sets an object's parameter names.

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

Usage

```
set_pars(x, value, ...)

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` *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

```
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))
```

Index