Package ‘posterior’

March 14, 2023

Title Tools for Working with Posterior Distributions

Version 1.4.1

Date 2023-03-09

Description Provides useful tools for both users and developers of packages
for fitting Bayesian models or working with output from Bayesian models.
The primary goals of the package are to:
(a) Efficiently convert between many different useful formats of
draws (samples) from posterior or prior distributions.
(b) Provide consistent methods for operations commonly performed on draws,
for example, subsetting, binding, or mutating draws.
(c) Provide various summaries of draws in convenient formats.
(d) Provide lightweight implementations of state of the art posterior
inference diagnostics. References: Vehtari et al. (2021)

Depends R (>= 3.2.0)

Imports methods, abind, checkmate, rlang (>= 1.0.6), stats, tibble (>=
3.1.0), vctrs (>= 0.5.0), tensorA, pillar, distributional,
parallel, matrixStats

Suggests testthat (>= 2.1.0), caret (>= 6.0.84), gbm (>= 2.1.8),
randomForest (>= 4.6.14), e1071 (>= 1.7.3), dplyr, tidyr,
knitr, ggplot2, rmarkdown

License BSD_3_clause + file LICENSE

Encoding UTF-8


BugReports https://github.com/stan-dev/posterior/issues

RoxygenNote 7.2.3

VignetteBuilder knitr

NeedsCompilation no

Author Paul-Christian Bürkner [aut, cre],
Jonah Gabry [aut],
Matthew Kay [aut],
topics documented:

Aki Vehtari [aut],
Måns Magnusson [ctb],
Rok Češnovar [ctb],
Ben Lambert [ctb],
Ozan Adigüzel [ctb],
Jacob Socolar [ctb]

Maintainer  Paul-Christian Bürkner <paul.buerkner@gmail.com>
Repository  CRAN
Date/Publication  2023-03-14 09:10:02 UTC

R topics documented:

posterior-package ......................................................... 3
as_rvar ................................................................. 4
as_rvar_factor ......................................................... 6
bind_draws ............................................................... 7
chol.rvar ............................................................... 8
diag.rvar-method ...................................................... 9
diagnostics .............................................................. 10
dissent ................................................................. 11
draws ................................................................. 12
draws-index ............................................................ 13
draws_array ........................................................... 15
draws_df ............................................................... 16
draws_list ............................................................. 18
draws_matrix .......................................................... 20
draws_of .............................................................. 21
draws_rvars ............................................................ 22
draws_summary .......................................................... 24
drop.rvar-method ...................................................... 26
entropy ................................................................. 27
ess_basic .............................................................. 29
ess_bulk .............................................................. 30
ess_mean ............................................................... 32
ess_quantile ........................................................... 33
ess_sd ................................................................. 34
ess_tail ............................................................... 35
example_draws .......................................................... 37
extract_variable ......................................................... 38
extract_variable_matrix ............................................. 39
for_each_draw .......................................................... 40
is_rvar .............................................................. 41
is_rvar_factor ........................................................ 42
match ................................................................. 42
mcse_mean ............................................................ 43
mcse_quantile .......................................................... 45
mcse_sd .............................................................. 46
The `posterior` package is intended to provide useful tools for both users and developers of packages for fitting Bayesian models or working with output from Bayesian models. The primary goals of the package are to:
• Efficiently convert between many different useful formats of draws (samples) from posterior or prior distributions.
• Provide consistent methods for operations commonly performed on draws, for example, subsetting, binding, or mutating draws.
• Provide various summaries of draws in convenient formats.
• Provide lightweight implementations of state of the art posterior inference diagnostics.

Package options
The following options are used to format and print draws objects, as in print.draws_array(), print.draws_df(), print.draws_list(), print.draws_matrix(), and print.draws_rvars():
• posterior.max_draws: Maximum number of draws to print.
• posterior.max_iterations: Maximum number of iterations to print.
• posterior.max_chains: Maximum number of chains to print.
• posterior.max_variables: Maximum number of variables to print.

The following options are used to format and print rvar objects, as in print.rvar() and print.draws_rvars():
• posterior.rvar_summary: What style of summary to display: "mean_sd" displays mean ± sd, "median_mad" displays median ± mad.
• posterior.digits: How many significant digits are displayed. This defaults to a smaller value (2) thangetOption("digits") because rvars print two numbers (point summary and uncertainty) next to each other.

The following option is used to construct new rvar objects, as in rfun() and rdo():
• posterior.rvar_ndraws: The number of draws used to construct new random variables when this number cannot be determined from existing arguments (e.g., other rvars passed to a function).

The following options are used to control warning messages:
• posterior.warn_on_merge_chains: (logical) Some operations will trigger an automatic merging of chains, for example, because chains do not match between two objects involved in a binary operation. Whether this causes a warning can be controlled by this option.

---

as_rvar

Coerce to a random variable

Description
Convert x to an rvar object.

Usage
as_rvar(x, dim = NULL, dimnames = NULL, nchains = NULL)
as_rvar_numeric(x, dim = NULL, dimnames = NULL, nchains = NULL)
Arguments

- **x**: (multiple options) An object that can be converted to an *rvar*, such as a vector, array, or an *rvar* itself.
- **dim**: (integer vector) One or more integers giving the maximal indices in each dimension to override the dimensions of the *rvar* to be created (see `dim()`). If NULL (the default), `dim` is determined by the input. NOTE: This argument controls the dimensions of the *rvar*, not the underlying array, so you cannot change the number of draws using this argument.
- **dimnames**: (list) Character vectors giving the names in each dimension to override the names of the dimensions of the *rvar* to be created (see `dimnames()`). If NULL (the default), this is determined by the input. NOTE: This argument controls the names of the dimensions of the *rvar*, not the underlying array.
- **nchains**: (positive integer) The number of chains. The default is 1.

Details

For objects that are already *rvars*, returns them (with modified dimensions if `dim` is not NULL).

For numeric or logical vectors or arrays, returns an *rvar* with a single draw and the same dimensions as `x`. This is in contrast to the *rvar()* constructor, which treats the first dimension of `x` as the draws dimension. As a result, as_rvar() is useful for creating constants.

While as_rvar() attempts to pick the most suitable subtype of *rvar* based on the type of `x` (possibly returning a an *rvar_factor* or *rvar_ordered*), as_rvar_numeric() always coerces the draws of the output *rvar* to be numeric, and always returns a base *rvar*, never a subtype.

Value

An object of class "rvar" (or one of its subtypes) representing a random variable.

See Also

- `rvar()` to construct *rvars* directly. See rdo(), rfun(), and rvar_rng() for higher-level interfaces for creating *rvars*.

Examples

```r
# You can use as_rvar() to create "constant" rvars (having only one draw):
x <- as_rvar(1)
x

# Such constants can be of arbitrary shape:
as_rvar(1:4)
as_rvar(matrix(1:10, nrow = 5))
as_rvar(array(1:12, dim = c(2, 3, 2)))

# as_rvar_numeric() coerces subtypes of rvar to the base rvar type
y <- as_rvar_factor(c("a", "b", "c"))
y
```
as_rvar_factor

Coerce to a factor random variable

Description

Convert x to an rvar_factor or rvar_ordered object.

Usage

as_rvar_factor(x, dim = NULL, dimnames = NULL, nchains = NULL, ...)
as_rvar_ordered(x, dim = NULL, dimnames = NULL, nchains = NULL, ...)

Arguments

x (multiple options) An object that can be converted to an rvar, such as a vector, array, or an rvar itself.
dim (integer vector) One or more integers giving the maximal indices in each dimension to override the dimensions of the rvar to be created (see dim()). If NULL (the default), dim is determined by the input. NOTE: This argument controls the dimensions of the rvar, not the underlying array, so you cannot change the number of draws using this argument.
dimnames (list) Character vectors giving the names in each dimension to override the names of the dimensions of the rvar to be created (see dimnames()). If NULL (the default), this is determined by the input. NOTE: This argument controls the names of the dimensions of the rvar, not the underlying array.
nchains (positive integer) The number of chains. The default is 1.
...

Arguments passed on to base::factor

levels an optional vector of the unique values (as character strings) that x might have taken. The default is the unique set of values taken by as.character(x), sorted into increasing order of x. Note that this set can be specified as smaller than sort(unique(x)).
labels either an optional character vector of labels for the levels (in the same order as levels after removing those in exclude), or a character string of length 1. Duplicated values in labels can be used to map different values of x to the same factor level.
exclude a vector of values to be excluded when forming the set of levels. This may be factor with the same level set as x or should be a character.
ordered logical flag to determine if the levels should be regarded as ordered (in the order given).
nmax an upper bound on the number of levels; see ‘Details’.
bind_draws

Details

For objects that are already rvars, returns them (with modified dimensions if dim is not NULL), possibly adding levels using the unique values of the draws of the rvar (if the object is not already factor-like).

For numeric, logical, factor, or character vectors or arrays, returns an rvar_factor or rvar_ordered with a single draw and the same dimensions as x. This is in contrast to the rvar_factor() and rvar_ordered() constructors, which treats the first dimension of x as the draws dimension. As a result, as_rvar_factor() and as_rvar_ordered() are useful for creating constants.

Value

An object of class "rvar_factor" or "rvar_ordered" representing a random variable.

See Also

rvar(), rvar_factor(), and rvar_ordered() to construct rvars directly. See rdo(), rfun(), and rvar_rng() for higher-level interfaces for creating rvars.

Examples

# You can use as_rvar_factor() to create "constant" rvars (having only one draw):
x <- as_rvar_factor("a")
x

# Such constants can be of arbitrary shape:
as_rvar_factor(letters[1:4])
as_rvar_ordered(matrix(letters[1:10], nrow = 5))
as_rvar_factor(array(letters[1:12], dim = c(2, 3, 2)))

bind_draws Bind draws objects together

Description

Bind multiple draws objects together to form a single draws object.

Usage

bind_draws(x, ...)

## S3 method for class 'draws_matrix'
bind_draws(x, ..., along = "variable")

## S3 method for class 'draws_array'
bind_draws(x, ..., along = "variable")
chol.rvar

Cholesky decomposition of random matrix

Description

Cholesky decomposition of an rvar containing a matrix.

Usage

## S3 method for class 'rvar'
chol(x, ...)
Arguments

x  (rvar) A 2-dimensional rvar.

... Additional parameters passed on to chol.tensor()

Value

An rvar containing the upper triangular factor of the Cholesky decomposition, i.e., the matrix $R$ such that $R'R = x$.

Description

Extract the diagonal of a matrix or construct a matrix, including random matrices (2-dimensional rvars). Makes base::diag() generic.

Usage

```r
## S4 method for signature 'rvar'
diag(x = 1, nrow, ncol, names = TRUE)
```

Arguments

x  (numeric,rvar) a matrix, vector, 1D array, missing, or a 1- or 2-dimensional rvar.

nrow, ncol  optional dimensions for the result when x is not a matrix.

names  (when x is a matrix) logical indicating if the resulting vector, the diagonal of x, should inherit names from dimnames(x) if available.

Details

Makes base::diag() into a generic function. See that function’s documentation for usage with numerics and for usage of diag<-, which is also supported by rvar.

Value

For rvars, has two modes:

1. x is a matrix-like rvar: it returns the diagonal as a vector-like rvar
2. x is a vector-like rvar: it returns a matrix-like rvar with x as the diagonal and zero for off-diagonal entries.

See Also

base::diag()
Examples

# Sigma is a 3x3 covariance matrix
Sigma <- as_draws_rvars(example_draws("multi_normal"))$Sigma
Sigma

diag(Sigma)

diag(Sigma) <- 1:3
Sigma

diag(as_rvar(1:3))

diagnostics

List of available convergence diagnostics

Description

A list of available diagnostics and links to their individual help pages.

Details

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ess_basic()</td>
<td>Basic version of effective sample size</td>
</tr>
<tr>
<td>ess_bulk()</td>
<td>Bulk effective sample size</td>
</tr>
<tr>
<td>ess_tail()</td>
<td>Tail effective sample size</td>
</tr>
<tr>
<td>ess_quantile()</td>
<td>Effective sample sizes for quantiles</td>
</tr>
<tr>
<td>ess_sd()</td>
<td>Effective sample sizes for standard deviations</td>
</tr>
<tr>
<td>mcse_mean()</td>
<td>Monte Carlo standard error for the mean</td>
</tr>
<tr>
<td>mcse_quantile()</td>
<td>Monte Carlo standard error for quantiles</td>
</tr>
<tr>
<td>mcse_sd()</td>
<td>Monte Carlo standard error for standard deviations</td>
</tr>
<tr>
<td>rhat_basic()</td>
<td>Basic version of Rhat</td>
</tr>
<tr>
<td>rhat()</td>
<td>Improved, rank-based version of Rhat</td>
</tr>
<tr>
<td>rstar()</td>
<td>R* diagnostic</td>
</tr>
</tbody>
</table>

Value

See individual functions for a description of return types.
Description

Dissention, for measuring dispersion in draws from ordinal distributions.

Usage

dissent(x)

## Default S3 method:
dissent(x)

## S3 method for class 'rvar'
dissent(x)

Arguments

x (multiple options) A vector to be interpreted as draws from an ordinal distribution, such as:

- A factor
- A numeric (should be integer or integer-like)
- An rvar, rvar_factor, or rvar_ordered

Details

Calculates Tastle and Wierman’s (2007) dissention measure:

\[
- \sum_{i=1}^{n} p_i \log_2 \left( 1 - \frac{|x_i - E(x)|}{\max(x) - \min(x)} \right)
\]

This ranges from 0 (all probability in one category) through 0.5 (uniform) to 1 (bimodal: all probability split equally between the first and last category).

Value

If x is a factor or numeric, returns a length-1 numeric vector with a value between 0 and 1 (inclusive) giving the dissention of x.

If x is an rvar, returns an array of the same shape as x, where each cell is the dissention of the draws in the corresponding cell of x.

References

Examples

set.seed(1234)

levels <- c("lowest", "low", "neutral", "high", "highest")

# a bimodal distribution: high dissention
x <- ordered(sample(levels, 4000, replace = TRUE, prob = c(0.45, 0.04, 0.02, 0.04, 0.45)), levels = levels)
dissent(x)

# a unimodal distribution: low dissention
y <- ordered(sample(levels, 4000, replace = TRUE, prob = c(0.95, 0.02, 0.015, 0.01, 0.005)), levels = levels)
dissent(y)

# both together, as an rvar
xy <- c(rvar(x), rvar(y))
xy
dissent(xy)

draws

Transform to draws objects

description

Try to transform an R object to a format supported by the posterior package.

Usage

as_draws(x, ...)

is_draws(x)

Arguments

x

(draws) A draws object or another R object for which the method is defined.

... Arguments passed to individual methods (if applicable).

details

The class "draws" is the parent class of all supported formats, which also have their own subclasses of the form "draws_{format}" (e.g. "draws_array").
Value

If possible, a draws object in the closest supported format to \( x \). The formats are linked to in the See Also section below.

See Also

Other formats: draws_array(), draws_df(), draws_list(), draws_matrix(), draws_rvars()

Examples

```r
# create some random draws
x <- matrix(rnorm(30), nrow = 10)
colnames(x) <- c("a", "b", "c")
str(x)

# transform to a draws object
y <- as_draws(x)
str(y)

# remove the draws classes from the object
class(y) <- class(y)[-c(1:2)]
str(y)
```

---

draws-index  

Index draws objects

Description

Index variables, iterations, chains, and draws.

Usage

variables(x, ...)

variables(x) <- value

iteration_ids(x)

chain_ids(x)

draw_ids(x)

nvariables(x, ...)

niterations(x)

nchains(x)
ndraws(x)

Arguments

x (draws) A draws object or another R object for which the method is defined.

... Arguments passed to individual methods (if applicable).

value (character vector) For variables(x) <- value, the new variable names to use.

Details

The methods variables(), iteration_ids(), chain_ids(), and draw_ids() return vectors of all variables, iterations, chains, and draws, respectively. In contrast, the methods nvariables(), niterations(), nchains(), and ndraws() return the number of variables, iterations, chains, and draws, respectively.

variables(x) <- value allows you to modify the vector of variable names, similar to how names(x) <- value works for vectors and lists. For renaming specific variables, rename_variables() may offer a more convenient approach.

Value

For variables(), a character vector.

For iteration_ids(), chain_ids(), and draw_ids(), an integer vector.

For niterations(), nchains(), and ndraws(), a scalar integer.

Examples

x <- example_draws()

variables(x)
nvariables(x)
variables(x) <- letters[1:nvariables(x)]

iteration_ids(x)
niterations(x)

chain_ids(x)
nchains(x)

draw_ids(x)
ndraws(x)
The `as_draws_array()` methods convert objects to the `draws_array` format. The `draws_array()` function creates an object of the `draws_array` format based on a set of numeric vectors. See Details.

### Usage

```r
draws_array(x, ...)  
## Default S3 method:  
draws_array(x, ...)  
## S3 method for class 'draws_array'  
draws_array(x, ...)  
## S3 method for class 'draws_matrix'  
draws_array(x, ...)  
## S3 method for class 'draws_df'  
draws_array(x, ...)  
## S3 method for class 'draws_list'  
draws_array(x, ...)  
## S3 method for class 'draws_rvars'  
draws_array(x, ...)  
## S3 method for class 'mcmc'  
draws_array(x, ...)  
## S3 method for class 'mcmc.list'  
draws_array(x, ...)  
draws_array(..., .nchains = 1)  
is_draws_array(x)
```

### Arguments

- `x` An object to convert to a `draws_array` object.
- `...` For `as_draws_array()`: Arguments passed to individual methods (if applicable). For `draws_array()`: Named arguments containing numeric vectors each defining a separate variable.
.nchains (positive integer) The number of chains. The default is 1.

Details

Objects of class "draws_array" are 3-D arrays with dimensions "iteration", "chain", and "variable". See Examples.

Value

A draws_array object, which has classes c("draws_array", "draws", "array").

See Also

Other formats: draws_df(), draws_list(), draws_matrix(), draws_rvars(), draws

Examples

x1 <- as_draws_array(example_draws())
class(x1)
print(x1)
str(x1)

x2 <- draws_array(a = rnorm(10), b = rnorm(10), c = 1)
class(x2)
print(x2)
str(x2)

draws_df

The draws_df format

draws_df

Description

The as_draws_df() methods convert objects to the draws_df format. The draws_df() function creates an object of the draws_df format based on a set of numeric vectors. See Details.

Usage

as_draws_df(x, ...)

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

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

## S3 method for class 'draws_df'
as_draws_df(x, ...)

## S3 method for class 'draws_matrix'
draws_df

as_draws_df(x, ...)

## S3 method for class 'draws_array'
as_draws_df(x, ...)

## S3 method for class 'draws_list'
as_draws_df(x, ...)

## S3 method for class 'draws_rvars'
as_draws_df(x, ...)

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

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

draws_df(..., .nchains = 1)

is_draws_df(x)

Arguments

x

An object to convert to a draws_df object.

...  

For as_draws_df(): Arguments passed to individual methods (if applicable).

For draws_df(): Named arguments containing numeric vectors each defining a separate variable.

.nchains

(positive integer) The number of chains. The default is 1.

Details

Objects of class "draws_df" are tibble data frames. They have one column per variable as well as additional metadata columns ".iteration", ".chain", and ".draw". The difference between the ".iteration" and ".draw" columns is that the former is relative to the MCMC chain while the latter ignores the chain information and has all unique values. See Examples.

If a data.frame-like object is supplied to as_draws_df that contains columns named ".iteration" or ".chain", they will be treated as iteration and chain indices, respectively. See Examples.

Value

A draws_df object, which has classes c("draws_df", "draws", class(tibble::tibble())).

See Also

Other formats: draws_array(), draws_list(), draws_matrix(), draws_rvars(), draws
Examples

```r
x1 <- as_draws_df(example_draws())
class(x1)
print(x1)
str(x1)

x2 <- draws_df(a = rnorm(10), b = rnorm(10), c = 1)
class(x2)
print(x2)
str(x2)

# the difference between iteration and draw is clearer when contrasting
# the head and tail of the data frame
print(head(x1), reserved = TRUE, max_variables = 2)
print(tail(x1), reserved = TRUE, max_variables = 2)

# manually supply chain information
xnew <- data.frame(mu = rnorm(10), .chain = rep(1:2, each = 5))
xnew <- as_draws_df(xnew)
print(xnew)
```

draws_list

The draws_list format

Description

The `as_draws_list()` methods convert objects to the draws_list format. The `draws_list()` function creates an object of the draws_list format based on a set of numeric vectors. See Details.

Usage

```r
as_draws_list(x, ...)

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

## S3 method for class 'draws_list'
as_draws_list(x, ...)

## S3 method for class 'draws_matrix'
as_draws_list(x, ...)

## S3 method for class 'draws_array'
as_draws_list(x, ...)

## S3 method for class 'draws_df'
as_draws_list(x, ...)
```
## S3 method for class 'draws_rvars'

as_draws_list(x, ...)

## S3 method for class 'mcmc'

as_draws_list(x, ...)

## S3 method for class 'mcmc.list'

as_draws_list(x, ...)

draws_list(..., .nchains = 1)

is_draws_list(x)

### Arguments

- **x**
  An object to convert to a `draws_list` object.

- **...**
  For `as_draws_list()`: Arguments passed to individual methods (if applicable).
  For `draws_list()`: Named arguments containing numeric vectors each defining a separate variable.

- **.nchains**
  (positive integer) The number of chains. The default is 1.

### Details

Objects of class "draws_list" are lists with one element per MCMC chain. Each of these elements is itself a named list of numeric vectors with one vector per variable. The length of each vector is equal to the number of saved iterations per chain. See `Examples`.

### Value

A `draws_list` object, which has classes c("draws_list", "draws", "list").

### See Also

Other formats: `draws_array()`, `draws_df()`, `draws_matrix()`, `draws_rvars()`, `draws`

### Examples

```r
x1 <- as_draws_list(example_draws())
class(x1)
print(x1)
str(x1)

x2 <- draws_list(a = rnorm(10), b = rnorm(10), c = 1)
class(x2)
print(x2)
str(x2)
```
**draws_matrix**

---

### Description

The `as_draws_matrix()` methods convert objects to the `draws_matrix` format. The `draws_matrix()` function creates an object of the `draws_matrix` format based on a set of numeric vectors. See **Details**.

### Usage

```r
as_draws_matrix(x, ...)  ## Default S3 method:
as_draws_matrix(x, ...)  ## S3 method for class 'draws_matrix'
as_draws_matrix(x, ...)  ## S3 method for class 'draws_array'
as_draws_matrix(x, ...)  ## S3 method for class 'draws_df'
as_draws_matrix(x, ...)  ## S3 method for class 'draws_list'
as_draws_matrix(x, ...)  ## S3 method for class 'draws_rvars'
as_draws_matrix(x, ...)  ## S3 method for class 'mcmc'
as_draws_matrix(x, ...)  ## S3 method for class 'mcmc.list'
draws_matrix(..., .nchains = 1)

is_draws_matrix(x)
```

### Arguments

- **x**
  - An object to convert to a `draws_matrix` object.

- **...**
  - For `as_draws_matrix()`: Arguments passed to individual methods (if applicable). For `draws_matrix()`: Named arguments containing numeric vectors each defining a separate variable.
draws_of

.nchains (positive integer) The number of chains. The default is 1.

Details

Objects of class "draws_matrix" are matrices (2-D arrays) with dimensions "draw" and "variable". See Examples.

Value

A draws_matrix object, which has classes c("draws_matrix", "draws", "matrix").

See Also

Other formats: draws_array(), draws_df(), draws_list(), draws_rvars(), draws

Examples

x1 <- as_draws_matrix(example_draws())
class(x1)
print(x1)
str(x1)

x2 <- draws_matrix(a = rnorm(10), b = rnorm(10), c = 1)
class(x2)
print(x2)
str(x2)

draws_of

Get/set array of draws underlying a random variable

Description

Gets/sets the array-representation that backs an rvar. Should be used rarely.

Usage

draws_of(x, with_chains = FALSE)

draws_of(x, with_chains = FALSE) <- value

Arguments

x (rvar) An rvar object.

with_chains (logical) Should the array of draws include a dimension for chains? If FALSE (the default), chains are not included and the array has dimension c(ndraws(x), dim(x)). If TRUE, chains are included and the array has dimension c(niterations(x), nchains(x), dim(x)).

value (array) An array of values to use as the backing array of x.
Details

While `rvars` implement fast versions of basic math operations (including matrix multiplication), sometimes you may need to bypass the `rvar` abstraction to do what you need to do more efficiently. `draws_of()` allows you to get / set the underlying array of draws in order to do that.

`rvars` represent draws internally using arrays of arbitrary dimension, which is returned by `draws_of(x)` and can be set using `draws_of(x) <- value`. The first dimension of these arrays is the index of the draws. If `with_chains = TRUE`, then the dimensions of the returned array are modified so that the first dimension is the index of the iterations and the second dimension is the index of the chains.

Value

If `with_chains = FALSE`, an array with dimensions `c(ndraws(x), dim(x))`.

If `with_chains = TRUE`, an array with dimensions `c(niterations(x), nchains(x), dim(x))`.

Examples

```r
x <- rvar(1:10, nchains = 2)
x
# draws_of() without arguments will return the array of draws without
# chain information (first dimension is draw)
draws_of(x)
# draws_of() with with_chains = TRUE will reshape the returned array to
# include chain information in the second dimension
draws_of(x, with_chains = TRUE)
# you can also set draws using draws_of(). When with_chains = FALSE the
# existing chain information will be retained ...
draws_of(x) <- 2:11
x
# when with_chains = TRUE the chain information will be set by the
# second dimension of the assigned array
draws_of(x, with_chains = TRUE) <- array(2:11, dim = c(2,5))
x
```

---

**draws_rvars**

*The draws_rvars format*

**Description**

The `as_draws_rvars()` methods convert objects to the `draws_rvars` format. The `draws_rvars()` function creates an object of the `draws_rvars` format based on a set of numeric vectors. See Details.
draws_rvars

Usage

as_draws_rvars(x, ...)

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

## S3 method for class 'draws_rvars'
as_draws_rvars(x, ...)

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

## S3 method for class 'draws_matrix'
as_draws_rvars(x, ...)

## S3 method for class 'draws_array'
as_draws_rvars(x, ...)

## S3 method for class 'draws_df'
as_draws_rvars(x, ...)

## S3 method for class 'draws_list'
as_draws_rvars(x, ...)

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

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

draws_rvars(..., .nchains = 1)

is_draws_rvars(x)

Arguments

x An object to convert to a draws_rvars object.

... For as_draws_rvars(): Arguments passed to individual methods (if applicable). For draws_rvars(): Named arguments containing numeric vectors each defining a separate variable.

.nchains (positive integer) The number of chains. The default is 1.

Details

Objects of class "draws_rvars" are lists of rvar objects. See Examples.
Value

A `draws_rvars` object, which has classes `c("draws_rvars", "draws", "list")`.

See Also

Other formats: `draws_array()`, `draws_df()`, `draws_list()`, `draws_matrix()`, `draws`

Examples

```r
x1 <- as_draws_rvars(example_draws())
class(x1)
print(x1)
str(x1)

x2 <- draws_rvars(a = rnorm(10), b = rnorm(10), c = 1)
class(x2)
print(x2)
str(x2)
```

---

**draws_summary**

**Summaries of draws objects**

Description

The `summarise_draws()` (and `summarize_draws()`) methods provide a quick way to get a table of summary statistics and diagnostics. These methods will convert an object to a `draws` object if it isn't already. For convenience, a `summary()` method for `draws` and `rvar` objects are also provided as an alias for `summarise_draws()` if the input object is a `draws` or `rvar` object.

Usage

```r
summarise_draws(.x, ...)
summarize_draws(.x, ...)

## S3 method for class 'draws'
summarise_draws(
  .x,
  ..., 
  .args = list(),
  .num_args =getOption("posterior.num_args", list()),
  .cores = 1
)

## S3 method for class 'draws'
summary(object, ...)

## S3 method for class 'rvar'
```
draws_summary

summarise_draws(.x, ...)

## S3 method for class 'rvar'
summary(object, ...)

default_summary_measures()

default_convergence_measures()

default_mcse_measures()

Arguments

- `.x`, `object` (draws) A draws object or one coercible to a draws object.
- `...` Name-value pairs of summary or diagnostic functions. The provided names will be used as the names of the columns in the result unless the function returns a named vector, in which case the latter names are used. The functions can be specified in any format supported by `as_function()`. See Examples.
- `.args` (named list) Optional arguments passed to the summary functions.
- `.num_args` (named list) Optional arguments passed to `num()` for pretty printing of summaries. Can be controlled globally via the `posterior.num_args` option.
- `.cores` (positive integer) The number of cores to use for computing summaries for different variables in parallel. Coerced to integer if possible, otherwise errors. The default is `.cores = 1`, in which case no parallelization is implemented. By default, a socket cluster is used on Windows and forks otherwise.

Details

The default summary functions used are the ones specified by `default_summary_measures()` and `default_convergence_measures()`:

default_summary_measures()

- `mean()`
- `median()`
- `sd()`
- `mad()`
- `quantile2()`

default_convergence_measures()

- `rhat()`
- `ess_bulk()`
- `ess_tail()`

The `var()` function should not be used to compute variances due to its inconsistent behavior with matrices. Instead, please use `distributional::variance()`.


Value

The `summarise_draws()` methods return a tibble data frame. The first column ("variable") contains the variable names and the remaining columns contain summary statistics and diagnostics. The functions `default_summary_measures()`, `default_convergence_measures()`, and `default_mcse_measures()` return character vectors of names of the default measures.

See Also

diagnostics for a list of available diagnostics and links to their individual help pages.

Examples

```r
x <- example_draws("eight_schools")
class(x)
str(x)

summarise_draws(x)
summarise_draws(x, "mean", "median")
summarise_draws(x, mean, mcse = mcse_mean)
summarise_draws(x, ~quantile(.x, probs = c(0.4, 0.6)))

# using default_*_meaures()
summarise_draws(x, default_summary_measures())
summarise_draws(x, default_convergence_measures())
summarise_draws(x, default_mcse_measures())

# compute variance of variables
summarise_draws(x, var = distributional::variance)

# illustrate use of ".args"
ws <- rexp(ndraws(x))
summarise_draws(x, weighted.mean, .args = list(w = ws))

# adjust how numerical summaries are printed
summarise_draws(x, .num_args = list(sigfig = 2, notation = "dec"))
```

---

**drop,rvar-method**  
*Drop redundant dimensions*

Description

Delete the dimensions of an rvar which are of size one. See `base::drop()`

Usage

```r
## S4 method for signature 'rvar'
drop(x)
```
**entropy**

**Arguments**

- **x** (rvar) an rvar.

**Value**

An rvar with the same length as x, but where any entry equal to 1 in dim(x) has been removed. The exception is if dim(x) == 1, in which case dim(drop(x)) == 1 as well (this is because rvars, unlike numerics, never have NULL dimensions).

**Examples**

```r
# Sigma is a 3x3 covariance matrix
Sigma <- as_draws_rvars(example_draws("multi_normal"))$Sigma
Sigma
Sigma[1, ]
drop(Sigma[1, ])

# equivalently ...
Sigma[1, drop = TRUE]
```

---

**entropy**  
**Normalized entropy**

**Description**

Normalized entropy, for measuring dispersion in draws from categorical distributions.

**Usage**

```r
entropy(x)
## Default S3 method:
entropy(x)
## S3 method for class 'rvar'
entropy(x)
```

**Arguments**

- **x** (multiple options) A vector to be interpreted as draws from a categorical distribution, such as:
  - A factor
  - A numeric (should be integer or integer-like)
  - An rvar, rvar_factor, or rvar_ordered
Details

Calculates the normalized Shannon entropy of the draws in \( x \). This value is the entropy of \( x \) divided by the maximum entropy of a distribution with \( n \) categories, where \( n \) is length(unique(\( x \))) for numeric vectors and length(levels(\( x \))) for factors:

\[
- \sum_{i=1}^{n} p_i \log(p_i) \over \log(n)
\]

This scales the output to be between 0 (all probability in one category) and 1 (uniform). This form of normalized entropy is referred to as \( H_{REL} \) in Wilcox (1967).

Value

If \( x \) is a factor or numeric, returns a length-1 numeric vector with a value between 0 and 1 (inclusive) giving the normalized Shannon entropy of \( x \).

If \( x \) is an rvar, returns an array of the same shape as \( x \), where each cell is the normalized Shannon entropy of the draws in the corresponding cell of \( x \).

References


Examples

```r
set.seed(1234)

levels <- c("a", "b", "c", "d", "e")

# a uniform distribution: high normalized entropy
x <- factor(
  sample(levels, 4000, replace = TRUE, prob = c(0.2, 0.2, 0.2, 0.2, 0.2)),
  levels = levels
)
entropy(x)

# a unimodal distribution: low normalized entropy
y <- factor(
  sample(levels, 4000, replace = TRUE, prob = c(0.95, 0.02, 0.015, 0.01, 0.005)),
  levels = levels
)
entropy(y)

# both together, as an rvar
xy <- c(rvar(x), rvar(y))
xy
entropy(xy)
```
### Description

Compute the basic effective sample size (ESS) estimate for a single variable as described in Gelman et al. (2013) with some changes according to Vehtari et al. (2021). For practical applications, we strongly recommend the improved ESS convergence diagnostics implemented in `ess_bulk()` and `ess_tail()`. See Vehtari (2021) for an in-depth comparison of different effective sample size estimators.

### Usage

```r
ess_basic(x, ...)
```

```r
## Default S3 method:
ess_basic(x, split = TRUE, ...)
```

```r
## S3 method for class 'rvar'
ess_basic(x, split = TRUE, ...)
```

### Arguments

- **x**
  - (multiple options) One of:
    - A matrix of draws for a single variable (iterations x chains). See `extract_variable_matrix()`.
    - An `rvar`.
  - `...` Arguments passed to individual methods (if applicable).

- **split**
  - (logical) Should the estimate be computed on split chains? The default is `TRUE`.

### Value

If the input is an array, returns a single numeric value. If any of the draws is non-finite, that is, `NA`, `NaN`, `Inf`, or `-Inf`, the returned output will be (numeric) `NA`. Also, if all draws within any of the chains of a variable are the same (constant), the returned output will be (numeric) `NA` as well. The reason for the latter is that, for constant draws, we cannot distinguish between variables that are supposed to be constant (e.g., a diagonal element of a correlation matrix is always 1) or variables that just happened to be constant because of a failure of convergence or other problems in the sampling process.

If the input is an `rvar`, returns an array of the same dimensions as the `rvar`, where each element is equal to the value that would be returned by passing the draws array for that element of the `rvar` to this function.
References


See Also

Other diagnostics: `ess_bulk()`, `ess_quantile()`, `ess_sd()`, `ess_tail()`, `mcse_mean()`, `mcse_quantile()`, `mcse_sd()`, `rhat_basic()`, `rhat()`, `rstar()`

Examples

```r
mu <- extract_variable_matrix(example_draws(), "mu")
ess_basic(mu)

d <- as_draws_rvars(example_draws("multi_normal"))
ess_basic(d$Sigma)
```

---

**ess_bulk**

*Bulk effective sample size (bulk-ESS)*

Description

Compute a bulk effective sample size estimate (bulk-ESS) for a single variable. Bulk-ESS is useful as a diagnostic for the sampling efficiency in the bulk of the posterior. It is defined as the effective sample size for rank normalized values using split chains. For the tail effective sample size see `ess_tail()`. See Vehtari (2021) for an in-depth comparison of different effective sample size estimators.

Usage

```r
ess_bulk(x, ...)
```

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

## S3 method for class 'rvar'
ess_bulk(x, ...)

Arguments

-\texttt{x} (multiple options) One of:
  - A matrix of draws for a single variable (iterations x chains). See \texttt{extract_variable_matrix()}.
  - An \texttt{rvar}.

... Arguments passed to individual methods (if applicable).

Value

If the input is an array, returns a single numeric value. If any of the draws is non-finite, that is, \texttt{NA}, \texttt{NaN}, \texttt{Inf}, or \texttt{-Inf}, the returned output will be (numeric) \texttt{NA}. Also, if all draws within any of the chains of a variable are the same (constant), the returned output will be (numeric) \texttt{NA} as well. The reason for the latter is that, for constant draws, we cannot distinguish between variables that are supposed to be constant (e.g., a diagonal element of a correlation matrix is always 1) or variables that just happened to be constant because of a failure of convergence or other problems in the sampling process.

If the input is an \texttt{rvar}, returns an array of the same dimensions as the \texttt{rvar}, where each element is equal to the value that would be returned by passing the draws array for that element of the \texttt{rvar} to this function.

References


See Also

Other diagnostics: \texttt{ess_basic()}, \texttt{ess_quantile()}, \texttt{ess_sd()}, \texttt{ess_tail()}, \texttt{mcse_mean()}, \texttt{mcse_quantile()}, \texttt{mcse_sd()}, \texttt{rhat_basic()}, \texttt{rhat()}, \texttt{rstar()}

Examples

\begin{verbatim}
mu <- extract_variable_matrix(example_draws(), "mu")
ess_bulk(mu)

d <- as_draws_rvars(example_draws("multi_normal"))
ess_bulk(d$Sigma)
\end{verbatim}
**Effective sample size for the mean**

**Description**

Compute an effective sample size estimate for a mean (expectation) estimate of a single variable.

**Usage**

```
ess_mean(x, ...)
```

## S3 method for class 'rvar'

```
ess_mean(x, ...)
```

**Arguments**

- `x` (multiple options) One of:
  - A matrix of draws for a single variable (iterations x chains). See `extract_variable_matrix()`.
  - An `rvar`.
- `...` Arguments passed to individual methods (if applicable).

**Value**

If the input is an array, returns a single numeric value. If any of the draws is non-finite, that is, `NA`, `NaN`, `Inf`, or `-Inf`, the returned output will be (numeric) `NA`. Also, if all draws within any of the chains of a variable are the same (constant), the returned output will be (numeric) `NA` as well. The reason for the latter is that, for constant draws, we cannot distinguish between variables that are supposed to be constant (e.g., a diagonal element of a correlation matrix is always 1) or variables that just happened to be constant because of a failure of convergence or other problems in the sampling process.

If the input is an `rvar`, returns an array of the same dimensions as the `rvar`, where each element is equal to the value that would be returned by passing the draws array for that element of the `rvar` to this function.

**References**


**Examples**

```r
mu <- extract_variable_matrix(example_draws(), "mu")
ess_mean(mu)

d <- as_draws_rvars(example_draws("multi_normal"))
ess_mean(d$Sigma)
```
**ess_quantile**  
*Effective sample sizes for quantiles*

**Description**
Compute effective sample size estimates for quantile estimates of a single variable.

**Usage**

```r
ess_quantile(x, probs = c(0.05, 0.95), ...)  
## Default S3 method:  
ess_quantile(x, probs = c(0.05, 0.95), names = TRUE, ...)  
## S3 method for class 'rvar'  
ess_quantile(x, probs = c(0.05, 0.95), names = TRUE, ...)  

ess_median(x, ...)  
## Default S3 method:  
ess_mean(x, ...)  
```

**Arguments**

- `x`  
  (multiple options) One of:
  - A matrix of draws for a single variable (iterations x chains). See `extract_variable_matrix()`.
  - An `rvar`.

- `probs`  
  (numeric vector) Probabilities in \([0, 1]\).

- `...`  
  Arguments passed to individual methods (if applicable).

- `names`  
  (logical) Should the result have a `names` attribute? The default is `TRUE`, but use `FALSE` for improved speed if there are many values in `probs`.

**Value**

If the input is an array, returns a numeric vector with one element per quantile. If any of the draws is non-finite, that is, `NA`, `NaN`, `Inf`, or `-Inf`, the returned output will be a vector of (numeric) `NA` values. Also, if all draws of a variable are the same (constant), the returned output will be a vector of (numeric) `NA` values as well. The reason for the latter is that, for constant draws, we cannot distinguish between variables that are supposed to be constant (e.g., a diagonal element of a correlation matrix is always 1) or variables that just happened to be constant because of a failure of convergence or other problems in the sampling process.

If the input is an `rvar` and `length(probs) == 1`, returns an array of the same dimensions as the `rvar`, where each element is equal to the value that would be returned by passing the draws array for that element of the `rvar` to this function. If `length(probs) > 1`, the first dimension of the result indexes the input probabilities; i.e. the result has dimension `c(length(probs), dim(x))`. 
References


See Also

Other diagnostics: ess_basic(), ess_bulk(), ess_sd(), ess_tail(), mcse_mean(), mcse_quantile(), mcse_sd(), rhat_basic(), rhat(), rstar()

Examples

```r
mu <- extract_variable_matrix(example_draws(), "mu")
ess_quantile(mu, probs = c(0.1, 0.9))

d <- as_draws_rvars(example_draws("multi_normal"))
ess_quantile(d$mu, probs = c(0.1, 0.9))
```

---

**ess_sd**

*Effective sample size for the standard deviation*

**Description**

Compute an effective sample size estimate for the standard deviation (SD) estimate of a single variable. This is defined as the effective sample size estimate for the absolute deviation from mean.

**Usage**

```r
ess_sd(x, ...)
```

## Default S3 method:
```r
ess_sd(x, ...)
```

## S3 method for class 'rvar'
```r
ess_sd(x, ...)
```

**Arguments**

- **x** (multiple options) One of:
  - A matrix of draws for a single variable (iterations x chains). See `extract_variable_matrix()`.
  - An rvar.

- **...** Arguments passed to individual methods (if applicable).
**Value**

If the input is an array, returns a single numeric value. If any of the draws is non-finite, that is, NA, NaN, Inf, or -Inf, the returned output will be (numeric) NA. Also, if all draws within any of the chains of a variable are the same (constant), the returned output will be (numeric) NA as well. The reason for the latter is that, for constant draws, we cannot distinguish between variables that are supposed to be constant (e.g., a diagonal element of a correlation matrix is always 1) or variables that just happened to be constant because of a failure of convergence or other problems in the sampling process.

If the input is an rvar, returns an array of the same dimensions as the rvar, where each element is equal to the value that would be returned by passing the draws array for that element of the rvar to this function.

**References**


**See Also**

Other diagnostics: ess_basic(), ess_bulk(), ess_quantile(), ess_tail(), mcse_mean(), mcse_quantile(), mcse_sd(), rhat_basic(), rhat(), rstar()  

**Examples**

```r
mu <- extract_variable_matrix(example_draws(), "mu")
ess_sd(mu)

d <- as_draws_rvars(example_draws("multi_normal"))
ess_sd(d$Sigma)
```

---

**Description**

Compute a tail effective sample size estimate (tail-ESS) for a single variable. Tail-ESS is useful as a diagnostic for the sampling efficiency in the tails of the posterior. It is defined as the minimum of the effective sample sizes for 5% and 95% quantiles. For the bulk effective sample size see ess_bulk(). See Vehtari (2021) for an in-depth comparison of different effective sample size estimators.
Usage

ess_tail(x, ...)

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

## S3 method for class 'rvar'
ess_tail(x, ...)

Arguments

x (multiple options) One of:

- A matrix of draws for a single variable (iterations x chains). See `extract_variable_matrix()`.
- An rvar.

... Arguments passed to individual methods (if applicable).

Value

If the input is an array, returns a single numeric value. If any of the draws is non-finite, that is, NA, NaN, Inf, or -Inf, the returned output will be (numeric) NA. Also, if all draws within any of the chains of a variable are the same (constant), the returned output will be (numeric) NA as well. The reason for the latter is that, for constant draws, we cannot distinguish between variables that are supposed to be constant (e.g., a diagonal element of a correlation matrix is always 1) or variables that just happened to be constant because of a failure of convergence or other problems in the sampling process.

If the input is an rvar, returns an array of the same dimensions as the rvar, where each element is equal to the value that would be returned by passing the draws array for that element of the rvar to this function.

References


See Also

Other diagnostics: `ess_basic()`, `ess_bulk()`, `ess_quantile()`, `ess_sd()`, `mcse_mean()`, `mcse_quantile()`, `mcse_sd()`, `rhat_basic()`, `rhat()`, `rstar()`

Examples

```r
mu <- extract_variable_matrix(example_draws(), "mu")
ess_tail(mu)

d <- as_draws_rvars(example_draws("multi_normal"))
```
example_draws

```r
ess_tail(d$sigma)
```

---

**example_draws**  
*Example draws objects*

### Description

Objects for use in examples, vignettes, and tests.

### Usage

```r
example_draws(example = "eight_schools")
```

### Arguments

- `example` *(string)*: The name of the example draws object. See **Details** for available options.

### Details

The following example draws objects are available.

- **eight_schools**: A `draws_array` object with 100 iterations from each of 4 Markov chains obtained by fitting the eight schools model described in Gelman et al. (2013) with Stan. The variables are:
  - `mu`: Overall mean of the eight schools
  - `tau`: Standard deviation between schools
  - `theta`: Individual means of each of the eight schools

- **multi_normal**: A `draws_array` object with 100 iterations from each of the 4 Markov chains obtained by fitting a 3-dimensional multivariate normal model to 100 simulated observations. The variables are:
  - `mu`: Mean parameter vector of length 3
  - `Sigma`: Covariance matrix of dimension 3 x 3

### Value

A `draws` object.

### Note

These objects are only intended to be used in demonstrations and tests. They contain fewer iterations and chains than recommended for performing actual inference.

### References

Examples

draws_eight_schools <- example_draws("eight_schools")
summarise_draws(draws_eight_schools)

draws_multi_normal <- example_draws("multi_normal")
summarise_draws(draws_multi_normal)

---

**extract_variable**  
*Extract draws of a single variable*

Description

Extract a vector of draws of a single variable.

Usage

```r
extract_variable(x, variable, ...)
```

## Default S3 method:
```
extract_variable(x, variable, ...)
```

## S3 method for class 'draws'
```
extract_variable(x, variable, ...)
```

## S3 method for class 'draws_rvars'
```
extract_variable(x, variable, ...)
```

Arguments

- `x` (draws) A draws object or another R object for which the method is defined.
- `variable` (string) The name of the variable to extract.
- `...` Arguments passed to individual methods (if applicable).

Value

A numeric vector of length equal to the number of draws.

Examples

```
x <- example_draws()
mu <- extract_variable(x, variable = "mu")
str(mu)
```
extract_variable_matrix

Extract matrix of a single variable

Description

Extract an iterations x chains matrix of draws of a single variable. This is primarily used for convergence diagnostic functions such as \texttt{rhat()}. 

Usage

\begin{verbatim}
extract_variable_matrix(x, variable, ...)

## Default S3 method:
extract_variable_matrix(x, variable, ...)

## S3 method for class 'draws'
extract_variable_matrix(x, variable, ...)

## S3 method for class 'draws_rvars'
extract_variable_matrix(x, variable, ...)
\end{verbatim}

Arguments

- \textit{x} (draws) A draws object or another \texttt{R} object for which the method is defined.
- \textit{variable} (string) The name of the variable to extract.
- \textit{...} Arguments passed to individual methods (if applicable).

Value

A matrix with dimension iterations x chains.

Examples

\begin{verbatim}
x <- example_draws()
mu <- extract_variable_matrix(x, variable = "mu")
dim(mu)
rhat(mu)
\end{verbatim}
for_each_draw  

Loop over draws

Description

Executes an expression once for every draw in a draws object. Used primarily for its side effects and returns the input x invisibly.

Usage

for_each_draw(x, expr)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>(draws) A draws object or another R object for which the method is defined.</td>
</tr>
<tr>
<td>expr</td>
<td>(expression) A bare expression that can contain references to variables in x by name. This expression will be executed once per draw of x, where references to variables in x resolve to the value of that variable in that draw. The expression supports quasiquotation.</td>
</tr>
</tbody>
</table>

Details

If x is not in the draws_rvars format, it is first converted to that format. This allows the variables in x to include their dimensions (i.e, to act as R vectors and arrays) when being referred to in expr.

Within expr, use .draw to refer to the draw index, which will be a value between 1 and n_draws(x). expr is executed in the calling environment of for_each_draw(), so it can use variables in that environment (however, due to the use of data masking, to modify variables in that environment, one must use <<-.)

Value

As for_each_draw() is used primarily for its side effects (the expression executed for each draw of x), it returns the input x invisibly.

Examples

eight_schools <- as_draws_rvars(example_draws())

# 1. A simple example --- looping over draws and printing each draw
# NOTE: You probably don't want to do this in practice! This example is
# just intended to show what for_each_draw() is doing. If you just want to
# print the draws of an rvar, it is probably better to use draws_of()
for_each_draw(eight_schools, {
    print(mu)
})
# 2. A more complex example --- building a parallel coordinates plot
# First, construct the plot bounds
plot(1, type = "n",
    xlim = c(1, length(eight_schools$theta)),
    ylim = range(range(eight_schools$theta)),
    xlab = "school", ylab = "theta"
)

# Then, use for_each_draw() to make a parallel coordinates plot of all draws
# of eight_schools$theta. Use resample_draws(eight_schools, n = ...) 
# in place of eight_schools if a smaller sample is desired for the plot.
for_each_draw(eight_schools, {
    lines(seq_along(theta), theta, col = rgb(1, 0, 0, 0.05))
})

# Finally, add means and 90% intervals
lines(seq_along(eight_schools$theta), mean(eight_schools$theta))
with(summarise_draws(eight_schools$theta),
    segments(seq_along(eight_schools$theta), y0 = q5, y1 = q95)
)

---

### is_rvar

Is x a random variable?

#### Description

Test if x is an rvar.

#### Usage

```
is_rvar(x)
```

#### Arguments

- **x**
  - (any object) An object to test.

#### Value

TRUE if x is an rvar, FALSE otherwise.

#### See Also

- `as_rvar()` to convert objects to rvars.
**is_rvar_factor**  
*Is x a factor random variable?*

**Description**  
Test if x is an rvar_factor or rvar_ordered.

**Usage**  

```r
is_rvar_factor(x)

is_rvar_ordered(x)
```

**Arguments**  

- `x` (any object) An object to test.

**Value**  

TRUE if x is an rvar_factor or rvar_ordered, FALSE otherwise.

**See Also**  

`as_rvar_factor()` and `as_rvar_ordered()` to convert objects to rvar_factors and rvar_ordereds.

---

**match**  
*Value Matching*

**Description**  
Generic version of `base::match()`. For base vectors, returns a vector of the positions of (first) matches of its first argument in its second. For rvars, returns an rvar of the matches.

**Usage**  

```r
match(x, table, ...)

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

# S3 method for class 'rvar'
match(x, ...)

x %in% table
```
mcse_mean

Arguments

x (multiple options) the values to be matched. Can be:
  • A base vector: see base::match()
  • An rvar

table (vector) the values to be matched against.

Arguments passed on to base::match

nomatch the value to be returned in the case when no match is found. Note that it is coerced to integer.

incomparables a vector of values that cannot be matched. Any value in x matching a value in this vector is assigned the nomatch value. For historical reasons, FALSE is equivalent to NULL.

Details

For more information on how match behaves with base vectors, see base::match().

When x is an rvar, the draws of x are matched against table using base::match(), and the result is returned as an rvar.

The implementation of %in% here is identical to base::%in%, except it uses the generic version of match() so that non-base vectors (such as rvars) are supported.

Value

When x is a base vector, a vector of the same length as x.

When x is an rvar, an rvar the same shape as x.

Examples

x <- rvar(c("a","b","b","c","d"))
x %in% c("b","d")

# for additional examples, see base::match()

---

mcse_mean Monte Carlo standard error for the mean

Description

Compute the Monte Carlo standard error for the mean (expectation) of a single variable.
Usage

mcse_mean(x, ...)

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

## S3 method for class 'rvar'
mcse_mean(x, ...)

Arguments

x
   (multiple options) One of:
   • A matrix of draws for a single variable (iterations x chains). See extract_variable_matrix().
   • An rvar.
...
   Arguments passed to individual methods (if applicable).

Value

If the input is an array, returns a single numeric value. If any of the draws is non-finite, that is, NA, NaN, Inf, or -Inf, the returned output will be (numeric) NA. Also, if all draws within any of the chains of a variable are the same (constant), the returned output will be (numeric) NA as well. The reason for the latter is that, for constant draws, we cannot distinguish between variables that are supposed to be constant (e.g., a diagonal element of a correlation matrix is always 1) or variables that just happened to be constant because of a failure of convergence or other problems in the sampling process.

If the input is an rvar, returns an array of the same dimensions as the rvar, where each element is equal to the value that would be returned by passing the draws array for that element of the rvar to this function.

References


See Also

Other diagnostics: ess_basic(), ess_bulk(), ess_quantile(), ess_sd(), ess_tail(), mcse_quantile(), mcse_sd(), rhat_basic(), rhat(), rstar()

Examples

mu <- extract_variable_matrix(example_draws(), "mu")
mcse_mean(mu)

d <- as_draws_rvars(example_draws("multi_normal"))
mcse_mean(d$Sigma)
mcse_quantile

Monte Carlo standard error for quantiles

Description

Compute Monte Carlo standard errors for quantile estimates of a single variable.

Usage

mcse_quantile(x, probs = c(0.05, 0.95), ...)

## Default S3 method:
mcse_quantile(x, probs = c(0.05, 0.95), names = TRUE, ...)

## S3 method for class 'rvar'
mcse_quantile(x, probs = c(0.05, 0.95), names = TRUE, ...)

mcse_median(x, ...)

Arguments

x  (multiple options) One of:
  • A matrix of draws for a single variable (iterations x chains). See `extract_variable_matrix()`.
  • An `rvar`.

probs (numeric vector) Probabilities in `[0, 1]`.

... Arguments passed to individual methods (if applicable).

names (logical) Should the result have a `names` attribute? The default is `TRUE`, but use `FALSE` for improved speed if there are many values in `probs`.

Value

If the input is an array, returns a numeric vector with one element per quantile. If any of the draws is non-finite, that is, `NA`, `NaN`, `Inf`, or `-Inf`, the returned output will be a vector of (numeric) `NA` values. Also, if all draws of a variable are the same (constant), the returned output will be a vector of (numeric) `NA` values as well. The reason for the latter is that, for constant draws, we cannot distinguish between variables that are supposed to be constant (e.g., a diagonal element of a correlation matrix is always 1) or variables that just happened to be constant because of a failure of convergence or other problems in the sampling process.

If the input is an `rvar` and `length(probs) == 1`, returns an array of the same dimensions as the `rvar`, where each element is equal to the value that would be returned by passing the draws array for that element of the `rvar` to this function. If `length(probs) > 1`, the first dimension of the result indexes the input probabilities; i.e. the result has dimension `c(length(probs), dim(x))`. 
mcse_sd

References


See Also

Other diagnostics: `ess_basic()`, `ess_bulk()`, `ess_quantile()`, `ess_sd()`, `ess_tail()`, `mcse_mean()`, `mcse_sd()`, `rhat_basic()`, `rhat()`, `rstar()`

Examples

```r
mu <- extract_variable_matrix(example_draws(), "mu")
mcse_quantile(mu, probs = c(0.1, 0.9))

d <- as_draws_rvars(example_draws("multi_normal"))
mcse_quantile(d$mu)
```

---

mcse_sd

**Monte Carlo standard error for the standard deviation**

Description

Compute the Monte Carlo standard error for the standard deviation (SD) of a single variable without assuming normality using moments of moments and first order Taylor series approximation (Kenney and Keeping, 1951, p. 141).

Usage

```r
mcse_sd(x, ...)
```

## Default S3 method:
```r
mcse_sd(x, ...)
```

## S3 method for class 'rvar'
```r
mcse_sd(x, ...)
```

Arguments

- `x` (multiple options) One of:
  - A matrix of draws for a single variable (iterations x chains). See `extract_variable_matrix()`.
  - An `rvar`.

- `...` Arguments passed to individual methods (if applicable).
### Value

If the input is an array, returns a single numeric value. If any of the draws is non-finite, that is, NA, NaN, Inf, or -Inf, the returned output will be (numeric) NA. Also, if all draws within any of the chains of a variable are the same (constant), the returned output will be (numeric) NA as well. The reason for the latter is that, for constant draws, we cannot distinguish between variables that are supposed to be constant (e.g., a diagonal element of a correlation matrix is always 1) or variables that just happened to be constant because of a failure of convergence or other problems in the sampling process.

If the input is an rvar, returns an array of the same dimensions as the rvar, where each element is equal to the value that would be returned by passing the draws array for that element of the rvar to this function.

### References


### See Also

Other diagnostics: `ess_basic()`, `ess_bulk()`, `ess_quantile()`, `ess_sd()`, `ess_tail()`, `mcse_mean()`, `mcse_quantile()`, `rhat_basic()`, `rhat()`, `rstar()`

### Examples

```r
mu <- extract_variable_matrix(example_draws(), "mu")
misce_sd(mu)

d <- as_draws_rvars(example_draws("multi_normal"))
misce_sd(d$Sigma)
```

---

### Description

Merge chains of draws objects into a single chain. Some operations will trigger an automatic merging of chains, for example, because chains do not match between two objects involved in a binary operation. By default, no warning will be issued when this happens but you can activate one via `options(posterior.warn_on_merge_chains = TRUE)`.
Usage

merge_chains(x, ...)

## S3 method for class 'draws_matrix'
merge_chains(x, ...)

## S3 method for class 'draws_array'
merge_chains(x, ...)

## S3 method for class 'draws_df'
merge_chains(x, ...)

## S3 method for class 'draws_list'
merge_chains(x, ...)

## S3 method for class 'rvar'
merge_chains(x, ...)

## S3 method for class 'draws_rvars'
merge_chains(x, ...)

Arguments

x (draws) A draws object or another R object for which the method is defined.
...
Arguments passed to individual methods (if applicable).

Value

A draws object of the same class as x.

Examples

x <- example_draws()

# draws_array with 4 chains, 100 iters each
str(x)

# draws_array with 1 chain of 400 iterations
str(merge_chains(x))

---

modal_category  Modal category

Description

Modal category of a vector.
modal_category

Usage
modal_category(x)

## Default S3 method:
modal_category(x)

## S3 method for class 'rvar'
modal_category(x)

Arguments

x  (multiple options) A vector to be interpreted as draws from a categorical distribution, such as:
  • A factor
  • A numeric (should be integer or integer-like)
  • An rvar, rvar_factor, or rvar_ordered

Details

Finds the modal category (i.e., most frequent value) in x. In the case of ties, returns the first tie.

Value

If x is a factor or numeric, returns a length-1 vector containing the modal value.

If x is an rvar, returns an array of the same shape as x, where each cell is the modal value of the draws in the corresponding cell of x.

Examples

x <- factor(c("a","b","b","c","d"))
modal_category(x)

# in the case of ties, the first tie is returned
y <- factor(c("a","c","c","d","d"))
modal_category(y)

# both together, as an rvar
xy <- c(rvar(x), rvar(y))
xy
modal_category(xy)
mutate_variables  

Mutate variables in draws objects

Description

Mutate variables in a `draws` object.

Usage

```r
mutate_variables(.x, ...)
```

## S3 method for class 'draws_matrix'
```r
mutate_variables(.x, ...)
```

## S3 method for class 'draws_array'
```r
mutate_variables(.x, ...)
```

## S3 method for class 'draws_df'
```r
mutate_variables(.x, ...)
```

## S3 method for class 'draws_list'
```r
mutate_variables(.x, ...)
```

## S3 method for class 'draws_rvars'
```r
mutate_variables(.x, ...)
```

Arguments

- `.x` (draws) A `draws` object.
- `...` Name-value pairs of expressions, each with either length 1 or the same length as in the entire input (i.e., number of iterations or draws). The name of each argument will be the name of a new variable, and the value will be its corresponding value. Use a `NULL` value in `mutate_variables` to drop a variable. New variables overwrite existing variables of the same name.

Details

In order to mutate variables in `draws_matrix` and `draws_array` objects, they are transformed to `draws_df` objects first and then transformed back after mutation. As those transformations are quite expensive for larger number of draws, we recommend using `mutate_variables` on `draws_df` and `draws_list` objects if speed is an issue.

In `draws_rvars` objects, the output of each expression in `...` is coerced to an `rvar` object if it is not already one using `as_rvar()`.

Value

Returns a `draws` object of the same format as `.x`, with variables mutated according to the expressions provided in `...`. 

### order_draws

#### See Also

`variables, rename_variables`

#### Examples

```r
x <- as_draws_df(example_draws())
x <- subset(x, variable = c("mu", "tau"))
mutate_variables(x, tau2 = tau^2)
mutate_variables(x, scale = 1.96 * tau, lower = mu - scale)
```

---

### order_draws

#### Order draws objects

Order `draws` objects according to iteration and chain number. By default, `draws` objects are ordered but subsetting or extracting parts of them may leave them in an unordered state.

#### Usage

```r
order_draws(x, ...)
```

#### Arguments

- `x` (draws) A `draws` object or another `R` object for which the method is defined.
- `...` Arguments passed to individual methods (if applicable).
print.draws_array

Value

A draws object of the same class as x.

See Also

repair_draws()

Examples

x <- as_draws_array(example_draws())
dimnames(x[10:5, 4:3, ])
dimnames(order_draws(x[10:5, 4:3, ]))

print.draws_array  Print draws_array objects

Description

Pretty printing for draws_array objects.

Usage

## S3 method for class 'draws_array'
print(
  x,
  digits = 2,
  max_iterations = getOption("posterior.max_iterations", 5),
  max_chains = getOption("posterior.max_chains", 8),
  max_variables = getOption("posterior.max_variables", 4),
  reserved = FALSE,
  ...
)

Arguments

x (draws) A draws object or another R object for which the method is defined.
digits (nonnegative integer) The minimum number of significant digits to print. If NULL, defaults to getOption("posterior.digits", 2).
max_iterations (positive integer) The maximum number of iterations to print. Can be controlled globally via the "posterior.max_iterations" option.
max_chains (positive integer) The maximum number of chains to print. Can be controlled globally via the "posterior.max_chains" option.
max_variables (positive integer) The maximum number of variables to print. Can be controlled globally via the "posterior.max_variables" option.
reserved (logical) Should reserved variables be included in the output? Defaults to FALSE. See reserved_variables for an overview of currently reserved variable names.
... Further arguments passed to the underlying print() methods.
print.draws_df

Value

A draws object of the same class as x.

Examples

```r
x <- as_draws_array(example_draws())
print(x)
```

---

**print.draws_df**  
*Print draws_df objects*

**Description**

Pretty printing for draws_df objects.

**Usage**

```r
## S3 method for class 'draws_df'
print(
  x,  
  digits = 2,  
  max_draws = getOption("posterior.max_draws", 10),  
  max_variables = getOption("posterior.max_variables", 8),  
  reserved = FALSE,  
  ...  
)
```

**Arguments**

- **x**  
  (draws) A draws object or another R object for which the method is defined.
- **digits**  
  (nonnegative integer) The minimum number of significant digits to print. If NULL, defaults to ```getOption("posterior.digits", 2)```.
- **max_draws**  
  (positive integer) The maximum number of draws to print. Can be controlled globally via the "posterior.max_draws" option.
- **max_variables**  
  (positive integer) The maximum number of variables to print. Can be controlled globally via the "posterior.max_variables" option.
- **reserved**  
  (logical) Should reserved variables be included in the output? Defaults to FALSE. See ```reserved_variables``` for an overview of currently reserved variable names.
- **...**  
  Further arguments passed to the underlying print() methods.

**Value**

A draws object of the same class as x.
print.draws_list

Examples

x <- as_draws_df(example_draws())
print(x)

print.draws_list
Print draws_list objects

Description

Pretty printing for draws_list objects.

Usage

## S3 method for class 'draws_list'
print(
  x,
  digits = 2,
  max_iterations = getOption("posterior.max_iterations", 10),
  max_chains = getOption("posterior.max_chains", 2),
  max_variables = getOption("posterior.max_variables", 4),
  reserved = FALSE,
  ...
)

Arguments

x (draws) A draws object or another R object for which the method is defined.
digits (nonnegative integer) The minimum number of significant digits to print. If
NULL, defaults to getOption("posterior.digits", 2).
max_iterations (positive integer) The maximum number of iterations to print. Can be controlled
globally via the "posterior.max_iterations" option.
max_chains (positive integer) The maximum number of chains to print. Can be controlled
globally via the "posterior.max_chains" option.
max_variables (positive integer) The maximum number of variables to print. Can be controlled
globally via the "posterior.max_variables" option.
reserved (logical) Should reserved variables be included in the output? Defaults to FALSE.
See reserved_variables for an overview of currently reserved variable names.
... Further arguments passed to the underlying print() methods.

Value

A draws object of the same class as x.
Examples

```r
x <- as_draws_list(example_draws())
print(x)
```

Description

Pretty printing for `draws_matrix` objects.

Usage

```r
## S3 method for class 'draws_matrix'
print(
  x, 
  digits = 2,
  max_draws = getOption("posterior.max_draws", 10),
  max_variables = getOption("posterior.max_variables", 8),
  reserved = FALSE,
  ...
)
```

Arguments

- `x` (draws) A draws object or another R object for which the method is defined.
- `digits` (nonnegative integer) The minimum number of significant digits to print. If NULL, defaults to `getOption("posterior.digits", 2)`.
- `max_draws` (positive integer) The maximum number of draws to print. Can be controlled globally via the "posterior.max_draws" option.
- `max_variables` (positive integer) The maximum number of variables to print. Can be controlled globally via the "posterior.max_variables" option.
- `reserved` (logical) Should reserved variables be included in the output? Defaults to FALSE. See `reserved_variables` for an overview of currently reserved variable names.
- `...` Further arguments passed to the underlying `print()` methods.

Value

A draws object of the same class as `x`.

Examples

```r
x <- as_draws_matrix(example_draws())
print(x)
```
print.draws_rvars

Print draws_rvars objects

Description
Pretty printing for draws_rvars objects.

Usage
## S3 method for class 'draws_rvars'
print(
  x,
  digits = 2,
  max_variables = getOption("posterior.max_variables", 8),
  summary = getOption("posterior.rvar_summary", "mean_sd"),
  reserved = FALSE,
  ...
)

Arguments
x (draws) A draws object or another R object for which the method is defined.
digits (nonnegative integer) The minimum number of significant digits to print. If NULL, defaults to getOption("posterior.digits", 2).
max_variables (positive integer) The maximum number of variables to print. Can be controlled globally via the "posterior.max_variables" option.
summary (string) The style of summary to display:
  • "mean_sd" displays mean ± sd
  • "median_mad" displays median ± mad
  • "mode_entropy" displays mode <entropy>, and is used automatically for rvar_factors. It shows normalized entropy, which ranges from 0 (all probability in one category) to 1 (uniform). See entropy().
  • "mode_dissent" displays mode <dissent>, and is used automatically for rvar_ordered. It shows Tastle and Wierman's (2007) dissention measure, which ranges from 0 (all probability in one category) through 0.5 (uniform) to 1 (bimodal: all probability split equally between the first and last category). See dissent().
  • NULL uses getOption("posterior.rvar_summary") (default "mean_sd")
reserved (logical) Should reserved variables be included in the output? Defaults to FALSE. See reserved_variables for an overview of currently reserved variable names.
... Further arguments passed to the underlying print() methods.

Value
A draws object of the same class as x.
Examples

```r
x <- as_draws_rvars(example_draws())
print(x)
```

Description

Print summaries of draws objects.

Usage

```r
## S3 method for class 'draws_summary'
print(x, ..., num_args = NULL)
```

Arguments

- `x` (draws_summary) A "draws_summary" object as output by `summarise_draws()`.
- `...` Additional arguments passed to `tibble::print.tbl_df()`.
- `num_args` (named list) Optional arguments passed to `num()` for pretty printing of summaries. If NULL (the default), uses the arguments stored in the "num_args" attribute of `x`, as set by the `.num_args` argument of `summarise_draws()`, which itself can be controlled globally via the `posterior.num_args` option.

Value

An invisible version of the input object.

Examples

```r
x <- example_draws("eight_schools")
# adjust how summaries are printed when calling summarise_draws()...
summarise_draws(x, .num_args = list(sigfig = 2, notation = "dec"))

# ... or when printing
s <- summarise_draws(x)
print(s, num_args = list(sigfig = 2, notation = "dec"))
print(s, num_args = list(digits = 3))
```
Description

Printing and formatting methods for rvars.

Usage

```r
## S3 method for class 'rvar'
print(
  x,
  ..., 
  summary = NULL,
  digits = NULL,
  color = TRUE,
  width = getOption("width")
)

## S3 method for class 'rvar'
format(x, ..., summary = NULL, digits = NULL, color = FALSE)

## S3 method for class 'rvar'
str(
  object,
  ..., 
  summary = NULL,
  vec.len = NULL,
  indent.str = paste(rep.int(" ", max(0, nest.lev + 1)), collapse = "."),
  nest.lev = 0,
  give.attr = TRUE
)
```

Arguments

- **x, object**  (rvar) The rvar to print.
- **...**  Further arguments passed to the underlying print() methods.
- **summary**  (string) The style of summary to display:
  - "mean_sd" displays mean ± sd
  - "median_mad" displays median ± mad
  - "mode_entropy" displays mode <entropy>, and is used automatically for rvar_factors. It shows normalized entropy, which ranges from 0 (all probability in one category) to 1 (uniform). See entropy().
  - "mode_dissent" displays mode <dissent>, and is used automatically for rvar_ordereds. It shows Tastle and Wierman’s (2007) dissention measure, which ranges from 0 (all probability in one category) through 0.5 (uniform)
to 1 (bimodal: all probability split equally between the first and last category). See dissent().

- NULL uses getOption("posterior.rvar_summary") (default "mean_sd")

digits (nonnegative integer) The minimum number of significant digits to print. If NULL, defaults to getOption("posterior.digits", 2).

color (logical) Whether or not to use color when formatting the output. If TRUE, the pillar::style_num() functions may be used to produce strings containing control sequences to produce colored output on the terminal.

width The maximum width used to print out lists of factor levels for rvar_factors. See format().

vec.len (nonnegative integer) How many 'first few' elements are displayed of each vector. If NULL, defaults to getOption("str")$vec.len, which defaults to 4.

indent.str (string) The indentation string to use.

nest.lev (nonnegative integer) Current nesting level in the recursive calls to str().

give.attr (logical) If TRUE (default), show attributes as sub structures.

Details

print() and str() print out rvar objects by summarizing each element in the random variable with either its mean±sd or median±mad, depending on the value of summary. Both functions use the format() implementation for rvar objects under the hood, which returns a character vector in the mean±sd or median±mad form.

Value

For print(), an invisible version of the input object.

For str(), nothing; i.e. invisible(NULL).

For format(), a character vector of the same dimensions as x where each entry is of the form "mean±sd" or "median±mad", depending on the value of summary.

References


Examples

set.seed(5678)
x = rbind(
  cbind(rvar(rnorm(1000, 1)), rvar(rnorm(1000, 2))),
  cbind(rvar(rnorm(1000, 3)), rvar(rnorm(1000, 4)))
)

print(x)
print(x, summary = "median_mad")
quantile2

*Compute Quantiles*

**Description**

Compute quantiles of a sample and return them in a format consistent with other summary functions in the *posterior* package.

**Usage**

```r
quantile2(x, probs = c(0.05, 0.95), na.rm = FALSE, ...)  
## Default S3 method:
quantile2(x, probs = c(0.05, 0.95), na.rm = FALSE, names = TRUE, ...)

## S3 method for class 'rvar'
quantile2(x, probs = c(0.05, 0.95), na.rm = FALSE, names = TRUE, ...)
```

**Arguments**

- `x` (multiple options) One of:
  - A matrix of draws for a single variable (iterations x chains). See `extract_variable_matrix()`.
  - An `rvar`.
- `probs` (numeric vector) Probabilities in \([0, 1]\).
- `na.rm` (logical) Should NA and NaN values be removed from `x` prior to computing quantiles? The default is FALSE.
- `...` Arguments passed to individual methods (if applicable) and then on to `stats::quantile()`.
- `names` (logical) Should the result have a names attribute? The default is TRUE, but use FALSE for improved speed if there are many values in `probs`.

**Value**

A numeric vector of length `length(probs)`. If `names = TRUE`, it has a `names` attribute with names like "q5", "q95", etc, based on the values of `probs`.

**Examples**

```r
mu <- extract_variable_matrix(example_draws(), "mu")
quantile2(mu)
```
**rdo**  
*Execute expressions of random variables*

**Description**

Execute (nearly) arbitrary R expressions that may include *rvars*, producing a new *rvar*.

**Usage**

\[ \text{rdo(expr, dim = NULL, ndraws = NULL)} \]

**Arguments**

- **expr** (expression) A bare expression that can (optionally) contain *rvars*. The expression supports quasiquotation.

- **dim** (integer vector) One or more integers giving the maximal indices in each dimension to override the dimensions of the *rvar* to be created (see *dim()*). If NULL (the default), dim is determined by the input. **NOTE:** This argument controls the dimensions of the *rvar*, not the underlying array, so you cannot change the number of draws using this argument.

- **ndraws** (positive integer) The number of draws used to construct new random variables if no *rvars* are supplied in *expr*. If NULL, getOption("posterior.rvar_ndraws") is used (default 4000). If *expr* contains *rvars*, the number of draws in the provided *rvars* is used instead of the value of this argument.

**Details**

This function evaluates *expr* possibly multiple times, once for each draw of the *rvars* it contains, then returns a new *rvar* representing the output of those expressions. To identify *rvars*, rdo() searches the calling environment for any variables named in *expr* for which *is_rvar()* evaluates to TRUE. If *expr* contains no *rvars*, then it will be executed *ndraws* times and an *rvar* with that many draws returned.

**rdo()** is not necessarily *fast* (in fact in some cases it may be very slow), but it has the advantage of allowing a nearly arbitrary R expression to be executed against *rvars* simply by wrapping it with *rdo( ... )*. This makes it especially useful as a prototyping tool. If you create code with *rdo()*, and it is unacceptably slow for your application, consider rewriting it using math operations directly on *rvars* (which should be fast), using *rvar_rng()* and/or using operations directly on the arrays that back the *rvars* (via *draws_of()*).

**Value**

An *rvar*.

**See Also**

Other *rfun*: *rfun()* , *rvar_rng()*
Examples

```r
mu <- rdo(rnorm(10, mean = 1:10, sd = 1))
sigma <- rdo(rgamma(1, shape = 1, rate = 1))
x <- rdo(rnorm(10, mu, sigma))
x
```

rename_variables	 Rename variables in draws objects

Description

Rename variables in a `draws` object.

Usage

```r
rename_variables(.x, ...)
```

## S3 method for class 'draws'
```r
rename_variables(.x, ...)
```

Arguments

- `.x` (draws) A `draws` object.
- `...` One or more expressions, separated by commas, indicating the variables to rename. The variable names can be unquoted (`new_name = old_name`) or quoted ("new_name" = "old_name"). For non-scalar variables, all elements can be renamed together ("new_name" = "old_name") or they can be renamed individually ("new_name[1]" = "old_name[1]").

Value

Returns a `draws` object of the same format as `.x`, with variables renamed according to the expressions provided in `...`.

See Also

`variables`, `mutate_variables`

Examples

```r
x <- as_draws_df(example_draws())
variables(x)

x <- rename_variables(x, mean = mu, sigma = tau)
variables(x)
```
**repair_draws**

```r
x <- rename_variables(x, b = 'theta[1]') # or b = "theta[1]"
variables(x)
```

# rename all elements of 'theta' at once
```r
x <- rename_variables(x, alpha = theta)
variables(x)
```

---

**repair_draws**  
*Repair indices of draws objects*

---

**Description**

Repair indices of draws objects so that iterations, chains, and draws are continuously and consistently numbered.

**Usage**

```r
repair_draws(x, order = TRUE, ...)
## S3 method for class 'draws_matrix'
repair_draws(x, order = TRUE, ...)
## S3 method for class 'draws_array'
repair_draws(x, order = TRUE, ...)
## S3 method for class 'draws_df'
repair_draws(x, order = TRUE, ...)
## S3 method for class 'draws_list'
repair_draws(x, order = TRUE, ...)
## S3 method for class 'draws_rvars'
repair_draws(x, order = TRUE, ...)
## S3 method for class 'rvar'
repair_draws(x, order = TRUE, ...)
```

**Arguments**

- **x** *(draws)* A draws object or another R object for which the method is defined.
- **order** *(logical)* Should draws be ordered (via order_draws()) before repairing indices? Defaults to TRUE.
- **...** Arguments passed to individual methods (if applicable).

**Value**

A draws object of the same class as `x`. 
resample_draws

Description

Resample draws objects according to provided weights, for example weights obtained through importance sampling.

Usage

resample_draws(x, ...)

## S3 method for class 'draws'
resample_draws(x, weights = NULL, method = "stratified", ndraws = NULL, ...)

## S3 method for class 'rvar'
resample_draws(x, ...)

Arguments

x (draws) A draws object or another R object for which the method is defined.

... Arguments passed to individual methods (if applicable).

weights (numeric vector) A vector of positive weights of length ndraws(x). The weights will be internally normalized. If weights is not specified, an attempt will be made to extract any weights already stored in the draws object (via weight_draws()). If no weights are stored in the draws object, equal weight is supplied to each draw. How exactly the weights influence the resampling depends on the method argument.

method (string) The resampling method to use:
- "simple": simple random resampling with replacement
- "simple_no_replace": simple random resampling without replacement
- "stratified": stratified resampling with replacement
- "deterministic": deterministic resampling with replacement

See Also

order_draws()

Examples

x <- as_draws_array(example_draws())
(x <- x[1:5, 3:4, ])
repair_draws(x)
Currently, "stratified" is the default as it has comparably low variance and bias with respect to ideal resampling. The latter would sample perfectly proportional to the weights, but this is not possible in practice due to the finite number of draws available. For more details about resampling methods, see Kitagawa (1996).

**ndraws**

(positive integer) The number of draws to be returned. By default ndraws is set internally to the total number of draws in x if sensible.

**Details**

Upon usage of `resample_draws()`, chains will automatically be merged due to subsetting of individual draws (see `subset_draws` for details). Also, weights stored in the draws object will be removed in the process, as resampling invalidates existing weights.

**Value**

A draws object of the same class as x.

**References**


**See Also**

`resample_draws()`

**Examples**

```r
x <- as_draws_df(example_draws())

# random weights for justr for demonstration
w <- runif(ndraws(x), 0, 10)

# use default stratified sampling
x_rs <- resample_draws(x, weights = w)
summarise_draws(x_rs, default_summary_measures())

# use simple random sampling
x_rs <- resample_draws(x, weights = w, method = "simple")
summarise_draws(x_rs, default_summary_measures())
```
reserved_variables

Description
Get names of reserved variables from objects in the posterior package.

Usage
reserved_variables(x, ...)
## Default S3 method:
reserved_variables(x, ...)

## S3 method for class 'draws_matrix'
reserved_variables(x, ...)

## S3 method for class 'draws_array'
reserved_variables(x, ...)

## S3 method for class 'draws_df'
reserved_variables(x, ...)

## S3 method for class 'draws_list'
reserved_variables(x, ...)

## S3 method for class 'draws_rvars'
reserved_variables(x, ...)

Arguments
x (draws) A draws object or another R object for which the method is defined.
... Arguments passed to individual methods (if applicable).

Details
reserved_variables() returns the names of reserved variables in use by an object.
The following variables names are currently reserved for special use cases in all draws formats:
  • .log_weight: Log weights per draw (see weight_draws).
Further, specific for the draws_df format, there are three additional reserved variables:
  • .chain: Chain index per draw
  • .iteration: Iteration index within each chain
  • .draw: Draw index across chains
More reserved variables may be added in the future.
Value

A character vector of reserved variables used in x.

Examples

```r
x <- example_draws()
reserved_variables(x)

# if we add weights, the `.log_weight` reserved variable is used
x <- weight_draws(x, rexp(ndraws(x)))
reserved_variables(x)
```

rfun                   
Create functions of random variables

Description

Function that create functions that can accept and/or produce rvars.

Usage

```r
rfun(.f, rvar_args = NULL, ndraws = NULL)
```

Arguments

- `.f` (multiple options) A function to turn into a function that accepts and/or produces random variables:
  - A function
  - A one-sided formula that can be parsed by `rlang::as_function()`
- `rvar_args` (character vector) The names of the arguments of `.f` that should be allowed to accept rvars as arguments. If NULL (the default), all arguments to `.f` are turned into arguments that accept rvars.
- `ndraws` (positive integer). The number of draws used to construct new random variables if no rvars are supplied as arguments to the returned function. If NULL, `getOption("posterior.rvar_ndraws")` is used (default 4000). If any arguments to the returned function contain rvars, the number of draws in the provided rvars is used instead of the value of this argument.

Details

This function wraps an existing function (.f) such that it returns rvars containing whatever type of data .f would normally return.

The returned function, when called, executes .f possibly multiple times, once for each draw of the rvars passed to it, then returns a new rvar representing the output of those function evaluations.
If the arguments contain no `rvars`, then `.f` will be executed `ndraws` times and an `rvar` with that many draws returned.

Functions created by `rfun()` are not necessarily *fast* (in fact in some cases they may be very slow), but they have the advantage of allowing a nearly arbitrary R functions to be executed against `rvars` simply by wrapping them with `rfun()`. This makes it especially useful as a prototyping tool. If you create code with `rfun()` and it is unacceptably slow for your application, consider rewriting it using math operations directly on `rvars` (which should be fast), using `rvar_rng()`, and/or using operations directly on the arrays that back the `rvars` (via `draws_of()`).

**Value**

A function with the same argument specification as `.f`, but which can accept and return `rvars`.

**See Also**

Other `rfun`: `rdo()`, `rvar_rng()`

**Examples**

```r
rvar_norm <- rfun(rnorm)
rvar_gamma <- rfun(rgamma)
mu <- rvar_norm(10, mean = 1:10, sd = 1)
sigma <- rvar_gamma(1, shape = 1, rate = 1)
x <- rvar_norm(10, mu, sigma)
x
```

---

**rhat**

*Rhat convergence diagnostic*

**Description**

Compute the Rhat convergence diagnostic for a single variable as the maximum of rank normalized split-Rhat and rank normalized folded-split-Rhat as proposed in Vehtari et al. (2021).

**Usage**

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

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

## S3 method for class 'rvar'
rhat(x, ...)

---
Arguments

- `x` (multiple options) One of:
  - A matrix of draws for a single variable (iterations x chains). See `extract_variable_matrix()`.
  - An `rvar`.
  - Arguments passed to individual methods (if applicable).

Value

If the input is an array, returns a single numeric value. If any of the draws is non-finite, that is, `NA`, `NaN`, `Inf`, or `-Inf`, the returned output will be (numeric) `NA`. Also, if all draws within any of the chains of a variable are the same (constant), the returned output will be (numeric) `NA` as well. The reason for the latter is that, for constant draws, we cannot distinguish between variables that are supposed to be constant (e.g., a diagonal element of a correlation matrix is always 1) or variables that just happened to be constant because of a failure of convergence or other problems in the sampling process.

If the input is an `rvar`, returns an array of the same dimensions as the `rvar`, where each element is equal to the value that would be returned by passing the draws array for that element of the `rvar` to this function.

References


See Also

Other diagnostics: `ess_basic()`, `ess_bulk()`, `ess_quantile()`, `ess_sd()`, `ess_tail()`, `mcse_mean()`, `mcse_quantile()`, `mcse_sd()`, `rhat_basic()`, `rstar()`

Examples

```r
mu <- extract_variable_matrix(example_draws(), "mu")
rhat(mu)

d <- as_draws_rvars(example_draws("multi_normal"))
rhat(d$Sigma)
```

---

**rhat_basic**  
*Basic version of the Rhat convergence diagnostic*

Description

Compute the basic Rhat convergence diagnostic for a single variable as described in Gelman et al. (2013) with some changes according to Vehtari et al. (2021). For practical applications, we strongly recommend the improved Rhat convergence diagnostic implemented in `rhat()`.
Usage

rhat_basic(x, ...)

## Default S3 method:
rhat_basic(x, split = TRUE, ...)

## S3 method for class 'rvar'
rhat_basic(x, split = TRUE, ...)

Arguments

x (multiple options) One of:

- A matrix of draws for a single variable (iterations x chains). See extract_variable_matrix()
- An rvar.

... Arguments passed to individual methods (if applicable).

split (logical) Should the estimate be computed on split chains? The default is TRUE.

Value

If the input is an array, returns a single numeric value. If any of the draws is non-finite, that is, NA, NaN, Inf, or -Inf, the returned output will be (numeric) NA. Also, if all draws within any of the chains of a variable are the same (constant), the returned output will be (numeric) NA as well. The reason for the latter is that, for constant draws, we cannot distinguish between variables that are supposed to be constant (e.g., a diagonal element of a correlation matrix is always 1) or variables that just happened to be constant because of a failure of convergence or other problems in the sampling process.

If the input is an rvar, returns an array of the same dimensions as the rvar, where each element is equal to the value that would be returned by passing the draws array for that element of the rvar to this function.

References


See Also

Other diagnostics: ess_basic(), ess_bulk(), ess_quantile(), ess_sd(), ess_tail(), mcse_mean(), mcse_quantile(), mcse_sd(), rhat(), rstar()

Examples

mu <- extract_variable_matrix(example_draws(), "mu")
rhat_basic(mu)
\begin{verbatim}
d <- as_draws_rvars(example_draws("multi_normal"))
rhat_basic(d$Sigma)
\end{verbatim}

\section*{rstar}
\textit{Calculate R* convergence diagnostic}

\subsection*{Description}
The \texttt{rstar()} function generates a measure of convergence for MCMC draws based on whether it is possible to determine the Markov chain that generated a draw with probability greater than chance. To do so, it fits a machine learning classifier to a training set of MCMC draws and evaluates its predictive accuracy on a testing set: giving the ratio of accuracy to predicting a chain uniformly at random.

\subsection*{Usage}
\begin{verbatim}
rstar(
x, 
  split = TRUE, 
  uncertainty = FALSE, 
  method = "rf", 
  hyperparameters = NULL, 
  training_proportion = 0.7, 
  nsimulations = 1000, 
  ... 
)
\end{verbatim}

\subsection*{Arguments}
\begin{itemize}
\item \texttt{x} \hspace{1cm} (draws) A \texttt{draws_df} object or one coercible to a \texttt{draws_df} object.
\item \texttt{split} \hspace{1cm} (logical) Should the estimate be computed on split chains? The default is \texttt{TRUE}.
\item \texttt{uncertainty} \hspace{1cm} (logical). Indicates whether to provide a vector of R* values representing uncertainty in the calculated value (if \texttt{TRUE}) or a single value (if \texttt{FALSE}). The default is \texttt{TRUE}.
\item \texttt{method} \hspace{1cm} (string) The machine learning classifier to use (must be available in the \texttt{caret} package). The default is \texttt{"rf"}, which calls the random forest classifier.
\item \texttt{hyperparameters} \hspace{1cm} (named list) Hyperparameter settings passed to the classifier. The default for the random forest classifier (\texttt{method = "rf"}) is \texttt{list(mtry = floor(sqrt(nvariables(x))))}. The default for the gradient-based model (\texttt{method = "gbm"}) is \texttt{list(interaction.depth = 3, n.trees = 50, shrinkage = 0.1, n.minobsinnode = 10)}.
\item \texttt{training_proportion} \hspace{1cm} (positive real) The proportion (in \((0, 1)\)) of iterations in used to train the classifier. The default is \texttt{0.7}.
\end{itemize}
nsimulations (positive integer) The number of R* values in the returned vector if uncertainty is TRUE. The default is 1000.

... Other arguments passed to caret::train().

Details

The rstar() function provides a measure of MCMC convergence based on whether it is possible to determine the chain that generated a particular draw with a probability greater than chance. To do so, it fits a machine learning classifier to a subset of the original MCMC draws (the training set) and evaluates its predictive accuracy on the remaining draws (the testing set). If predictive accuracy exceeds chance (i.e. predicting the chain that generated a draw uniformly at random), the diagnostic measure R* will be above 1, indicating that convergence has yet to occur. This statistic is recently developed, and it is currently unclear what is a reasonable threshold for diagnosing convergence.

The statistic, R*, is stochastic, meaning that each time the test is run, unless the random seed is fixed, it will generally produce a different result. To minimize the implications of this stochasticity, it is recommended to repeatedly run this function to calculate a distribution of R*; alternatively, an approximation to this distribution can be obtained by setting uncertainty = TRUE, although this approximation of uncertainty will generally have a lower mean.

By default, a random forest classifier is used (method = "rf"), which tends to perform best for target distributions of around 4 dimensions and above. For lower dimensional targets, gradient boosted models (called via method = "gbm") tend to have a higher classification accuracy. On a given MCMC sample, it is recommended to try both of these classifiers.

Value

A numeric vector of length 1 (by default) or length nsimulations (if uncertainty = TRUE).

References


See Also

Other diagnostics: ess_basic(), ess_bulk(), ess_quantile(), ess_sd(), ess_tail(), mcse_mean(), mcse_quantile(), mcse_sd(), rhat_basic(), rhat()

Examples

if (require("caret", quietly = TRUE) & require("randomForest", quietly = TRUE)) {
  x <- example_draws("eight_schools")
  print(rstar(x))
  print(rstar(x, split = FALSE))
  print(rstar(x, method = "gbm"))
  # can pass additional arguments to methods
  print(rstar(x, method = "gbm", verbose = FALSE))

  # with uncertainty, returns a vector of R* values
  hist(rstar(x, uncertainty = TRUE))}
hist(rstar(x, uncertainty = TRUE, nsimulations = 100))

# can use other classification methods in caret library
print(rstar(x, method = "knn"))
}

---

rvar

Random variables of arbitrary dimension

**Description**

Random variables backed by arrays of arbitrary dimension

**Usage**

```r
rvar(
  x = double(),
  dim = NULL,
  dimnames = NULL,
  nchains = NULL,
  with_chains = FALSE
)
```

**Arguments**

- **x**
  - (multiple options) The object to convert to an `rvar`:
    - A vector of draws from a distribution.
    - An array where the first dimension represents draws from a distribution. The resulting `rvar` will have dimension `dim(x)[-1]`; that is, everything except the first dimension is used for the shape of the variable, and the first dimension is used to index draws from the distribution (see `Examples`). Optionally, if `with_chains == TRUE`, the first dimension indexes the iteration and the second dimension indexes the chain (see `with_chains`).
    - An `rvar`.

- **dim**
  - (integer vector) One or more integers giving the maximal indices in each dimension to override the dimensions of the `rvar` to be created (see `dim()`). If NULL (the default), `dim` is determined by the input. **NOTE:** This argument controls the dimensions of the `rvar`, not the underlying array, so you cannot change the number of draws using this argument.

- **dimnames**
  - (list) Character vectors giving the names in each dimension to override the names of the dimensions of the `rvar` to be created (see `dimnames()`). If NULL (the default), this is determined by the input. **NOTE:** This argument controls the names of the dimensions of the `rvar`, not the underlying array.

- **nchains**
  - (positive integer) The number of chains. The if NULL (the default), 1 is used unless `x` is already an `rvar`, in which case the number of chains it has is used.
with_chains (logical) Does x include a dimension for chains? If FALSE (the default), chains are not included, the first dimension of the input array should index draws, and the nchains argument can be used to determine the number of chains. If TRUE, the nchains argument is ignored and the second dimension of x is used to index chains. Internally, the array will be converted to a format without the chain index. Ignored when x is already an rvar.

Details

The "rvar" class internally represents random variables as arrays of arbitrary dimension, where the first dimension is used to index draws from the distribution. Most mathematical operators and functions are supported, including efficient matrix multiplication and vector and array-style indexing. The intent is that an rvar works as closely as possible to how a base vector/matrix/array does, with a few differences:

- The default behavior when subsetting is not to drop extra dimensions (i.e. the default drop argument for [ is FALSE, not TRUE).
- Rather than base R-style recycling, rvars use a limited form of broadcasting: if an operation is being performed on two vectors with different size of the same dimension, the smaller vector will be recycled up to the size of the larger one along that dimension so long as it has size 1.

For functions that expect base numeric arrays and for which rvars cannot be used directly as arguments, you can use rfun() or rdo() to translate your code into code that executes across draws from one or more random variables and returns a random variable as output. Typically rdo() offers the most straightforward translation.

As rfun() and rdo() incur some performance cost, you can also operate directly on the underlying array using the draws_of() function. To re-use existing random number generator functions to efficiently create rvars, use rvar_rng().

Value

An object of class "rvar" representing a random variable.

See Also

as_rvar() to convert objects to rvars. See rdo(), rfun(), and rvar_rng() for higher-level interfaces for creating rvars.

Examples

```r
set.seed(1234)

# To create a "scalar" 'rvar', pass a one-dimensional array or a vector
# whose length (here '4000') is the desired number of draws:
x <- rvar(rnorm(4000, mean = 1, sd = 1))
x

# Create random vectors by adding an additional dimension:
n <- 4  # length of output vector
```
rvar-dist

Density, CDF, and quantile functions of random variables

Description

The probability density function (density()), cumulative distribution function (cdf()), and quantile function / inverse CDF (quantile()) of an rvar.

Usage

## S3 method for class 'rvar'
density(x, at, ...)

## S3 method for class 'rvar_factor'
density(x, at, ...)

## S3 method for class 'rvar'
cdf(x, q, ...)

## S3 method for class 'rvar_factor'
cdf(x, q, ...)

## S3 method for class 'rvar_ordered'
cdf(x, q, ...)

## S3 method for class 'rvar'
quantile(x, probs, ...)
## S3 method for class 'rvar_factor'
quantile(x, probs, ...)

## S3 method for class 'rvar_ordered'
quantile(x, probs, ...)

### Arguments

- **x** (rvar) An `rvar` object.
- **...** Additional arguments passed onto underlying methods:
  - For `density()`, these are passed to `stats::density()`.
  - For `cdf()`, these are ignored.
  - For `quantile()`, these are passed to `stats::quantile()`.
- **q, at** (numeric vector) One or more quantiles.
- **probs** (numeric vector) One or more probabilities in \([0,1]\).

### Value

If `x` is a scalar `rvar`, returns a vector of the same length as the input (q, at, or probs) containing values from the corresponding function of the given `rvar`.

If `x` has length greater than 1, returns an array with dimensions `c(length(y), dim(x))` where `y` is q, at, or probs, where each `result[i,...]` is the value of the corresponding function, `f(y[i])`, for the corresponding cell in the input array, `x[...]`.

### Examples

```r
set.seed(1234)
x = rvar(rnorm(100))
density(x, seq(-2, 2, length.out = 10))
cdf(x, seq(-2, 2, length.out = 10))
quantile(x, ppoints(10))

x2 = c(rvar(rnorm(100, mean = -0.5)), rvar(rnorm(100, mean = 0.5)))
density(x2, seq(-2, 2, length.out = 10))
cdf(x2, seq(-2, 2, length.out = 10))
quantile(x2, ppoints(10))
```

---

**rvar-matmult**

Matrix multiplication of random variables

### Description

Matrix multiplication of random variables.
Usage

x %**% y

Arguments

x (multiple options) The object to be postmultiplied by y:
  • An rvar
  • A numeric vector or matrix
  • A logical vector or matrix

If a vector is used, it is treated as a row vector.

y (multiple options) The object to be premultiplied by x:
  • An rvar
  • A numeric vector or matrix
  • A logical vector or matrix

If a vector is used, it is treated as a column vector.

Details

If x or y are vectors, they are converted into matrices prior to multiplication, with x converted to a row vector and y to a column vector. Numerics and logicals can be multiplied by rvars and are broadcasted across all draws of the rvar argument. Tensor multiplication is used to efficiently multiply matrices across draws, so if either x or y is an rvar, x %**% y will be much faster than rdo(x %*% y).

Because rvar is an S3 class and S3 classes cannot properly override %*, rvars use %**% for matrix multiplication.

Value

An rvar representing the matrix product of x and y.

Examples

# d has mu (mean vector of length 3) and Sigma (3x3 covariance matrix)
d <- as_draws_rvars(example_draws("multi_normal"))
d$Sigma

# trivial example: multiplication by a non-random matrix
d$Sigma %**% diag(1:3)

# Decompose Sigma into R s.t. R'R = Sigma ...
R <- chol(d$Sigma)
# ... and recreate Sigma using matrix multiplication
t(R) %**% R
rvar-summaries-over-draws

Summaries of random variables within array elements, over draws

Description

Compute summaries within elements of an rvar and over draws of each element, producing an array of the same shape as the input random variable (except in the case of range(), see Details).

Usage

E(x, ...)

## S3 method for class 'rvar'
mean(x, ...)

Pr(x, ...)

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

## S3 method for class 'logical'
Pr(x, ...)

## S3 method for class 'rvar'
Pr(x, ...)

## S3 method for class 'rvar'
median(x, ...)

## S3 method for class 'rvar'
min(x, ...)

## S3 method for class 'rvar'
max(x, ...)

## S3 method for class 'rvar'
sum(x, ...)

## S3 method for class 'rvar'
prod(x, ...)

## S3 method for class 'rvar'
all(x, ...)

## S3 method for class 'rvar'
any(x, ...)


## S3 method for class 'rvar'
Summarize(...)

## S3 method for class 'rvar'
variance(x, ...)

var(x, ...)

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

## S3 method for class 'rvar'
var(x, ...)

sd(x, ...)

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

## S3 method for class 'rvar'
sd(x, ...)

mad(x, ...)

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

## S3 method for class 'rvar'
mad(x, ...)

## S3 method for class 'rvar_ordered'
mad(x, ...)

## S3 method for class 'rvar'
range(x, ...)

## S3 method for class 'rvar'
is.finite(x)

## S3 method for class 'rvar'
is.infinite(x)

## S3 method for class 'rvar'
is.nan(x)

## S3 method for class 'rvar'
is.na(x)

rvar-summaries-over-draws
Arguments

- **x**: (rvar) An `rvar`.
- ... Further arguments passed to underlying functions (e.g., `base::mean()` or `base::median()`), such as `na.rm`.

Details

Summaries include expectations (`E()` or `mean()`), probabilities (`Pr()`), medians (`median()`), spread (`var()`, `variance()`, `sd()`, `mad()`), sums and products (`sum()`, `prod()`), extrema and ranges (`min()`, `max()`, `range()`), logical summaries (`all()`, `any()`), and special value predicates (`is.finite()`, `is.infinite()`, `is.nan()`, `is.na()`).

Unless otherwise stated, these functions return a numeric array with the same shape (same dimensions) as the input `rvar`, `x`.

- `range(x)` returns an array with dimensions `c(2, dim(x))`, where the last dimension contains the minimum and maximum values.
- `is.infinite(x)`, `is.nan(x)`, and `is.na(x)` return logical arrays, where each element is TRUE if any draws in its corresponding element in `x` match the predicate. Each elements in the result of `is.finite(x)` is TRUE if all draws in the corresponding element in `x` are finite.

Both `E()`, `mean()`, and `Pr()` return the means of each element in the input. `Pr()` additionally checks that the provided `rvar` is a logical variable (hence, taking its expectation results in a probability).

For consistency, `E()` and `Pr()` are also defined for base arrays so that they can be used as summary functions in `summarise_draws()`.

Value

A numeric or logical vector with the same dimensions as the given random variable, where each entry in the vector is the mean, median, or variance of the corresponding entry in `x`.

See Also

- `rvar-summaries-within-draws` for summary functions within draws. `rvar-dist` for density, CDF, and quantile functions of random variables.
- Other `rvar-summaries`: `rvar-summaries-within-draws`, `rvar_is_finite()`

Examples

```r
set.seed(5678)
x = rvar_rng(rnorm, 4, mean = 1:4, sd = 2)

# These should all be ~= c(1, 2, 3, 4)
E(x)
mean(x)
median(x)

# This ...
Pr(x < 1.5)
# ... should be about the same as this:
```
Summaries of random variables over array elements, within draws

Description

Compute summaries of random variables over array elements and within draws, producing a new random variable of length 1 (except in the case of rvar_range(), see Details).

Usage

rvar_mean(..., na.rm = FALSE)
rvar_median(..., na.rm = FALSE)
rvar_sum(..., na.rm = FALSE)
rvar_prod(..., na.rm = FALSE)
rvar_min(..., na.rm = FALSE)
rvar_max(..., na.rm = FALSE)
rvar_sd(..., na.rm = FALSE)
rvar_var(..., na.rm = FALSE)
rvar_mad(..., constant = 1.4826, na.rm = FALSE)
rvar_range(..., na.rm = FALSE)
rvar_quantile(..., probs, names = FALSE, na.rm = FALSE)
rvar_all(..., na.rm = FALSE)
rvar_any(..., na.rm = FALSE)

Arguments

... (rvar) One or more rvars.
na.rm (logical) Should NAs be removed from the input before summaries are computed? The default is FALSE.
**rvar_apply**

Random variable resulting from a function applied over margins of an array or random variable

---

constant  
(scalar real) For rvar_mad(), a scale factor for computing the median absolute deviation. See the details of stats::mad() for the justification for the default value.

probs  
(numeric vector) For rvar_quantile(), probabilities in [0, 1].

names  
(logical) For rvar_quantile(), if TRUE, the result has a names attribute.

**Details**

These functions compute statistics within each draw of the random variable. For summaries over draws (such as expectations), see rvar-summaries-over-draws.

Each function defined here corresponds to the base function of the same name without the rvar_ prefix (e.g., rvar_mean() calls mean() under the hood, etc).

**Value**

An rvar of length 1 (for range(), length 2; for quantile(), length equal to length(probs)) with the same number of draws as the input rvar(s) containing the summary statistic computed within each draw of the input rvar(s).

**See Also**

- rvar-summaries-over-draws for summary functions across draws (e.g. expectations). rvar-dist for density, CDF, and quantile functions of random variables.

Other rvar-summaries: rvar-summaries-over-draws, rvar_is_finite()

**Examples**

```r
set.seed(5678)
x = rvar_rng(rnorm, 4, mean = 1:4, sd = 2)

# These will give similar results to mean(1:4),  
# median(1:4), sum(1:4), prod(1:4), etc
rvar_mean(x)
rvar_median(x)
rvar_sum(x)
rvar_prod(x)
rvar_range(x)
rvar_quantile(x, probs = c(0.25, 0.5, 0.75), names = TRUE)
```

---

**rvar_apply**
Description

Returns an rvar obtained by applying a function to margins of an array or rvar. Acts like apply(), except that the function supplied (.f) should return an rvar, and the final result is always an rvar.

Usage

rvar_apply(.x, .margin, .f, ...)

Arguments

.x
An array or an rvar.

.margin
(multiple options) The subscripts which the function will be applied over:

- An integer vector. E.g., for a matrix 1 indicates rows, 2 indicates columns, c(1, 2) indicates rows and columns.
- A character vector of dimension names if .x has named dimensions.

.f
(function) The function to be applied. The function .f must return an rvar and the dimensions of the result of .f applied to each margin of .x must be able to be broadcasted to a common shape (otherwise the resulting rvar cannot be simplified). See Details.

...
Optional arguments passed to .f.

Details

This function acts much like apply(), except that the function passed to it (.f) must return rvars, and the result is simplified into an rvar. Unlike apply(), it also keeps the dimensions of the returned values along each margin, rather than simplifying each margin to a vector, and if the results of .f do not all have the same dimensions, it applies the rvar broadcasting rules to bind results together rather than using vector recycling.

If you wish to apply functions over rvars where the result is not intended to be simplified into an rvar, you can use the standard apply(), lapply(), sapply(), or vapply() functions.

Value

An rvar.

If the result of each call to .f returns an rvar of dimension d after being broadcast to a common shape, then rvar_apply() returns an rvar of dimension c(d, dim(.x)[.margin]). If the last dimension of the result would be 1, it is dropped (other dimensions equal to 1 are retained). If d is 0, the result has length 0 but not necessarily the ‘correct’ dimension.

See Also

as_rvar() to convert objects to rvars. See rdo(), rfun(), and rvar_rng() for higher-level interfaces for creating rvars.
Examples

```
set.seed(3456)
x <- rvar_rng(rnorm, 24, mean = 1:24)
        dim(x) <- c(2,3,4)

# we can find the distributions of marginal means of the above array
# using rvar_mean along with rvar_apply
rvar_apply(x, 1, rvar_mean)
rvar_apply(x, 2:3, rvar_mean)
```

---

**rvar_factor**

Factor random variables of arbitrary dimension

Description

Random variables backed by factor-like arrays of arbitrary dimension.

Usage

```
rvar_factor(
x = factor(),
dim = NULL,
dimnames = NULL,
nchains = NULL,
with_chains = FALSE,
...
)
```

```
rvar_ordered(
x = ordered(NULL),
dim = NULL,
dimnames = NULL,
nchains = NULL,
with_chains = FALSE,
...
)
```

Arguments

- `x` (multiple options) The object to convert to an `rvar`:
  - A vector of draws from a distribution.
  - An array where the first dimension represents draws from a distribution. The resulting `rvar` will have dimension `dim(x)[-1]`; that is, everything except the first dimension is used for the shape of the variable, and the first


dimension is used to index draws from the distribution (see Examples). Optionally, if with_chains == TRUE, the first dimension indexes the iteration and the second dimension indexes the chain (see with_chains).

- An rvar.

**dim**

(integer vector) One or more integers giving the maximal indices in each dimension to override the dimensions of the rvar to be created (see dim()). If NULL (the default), `dim` is determined by the input. **NOTE:** This argument controls the dimensions of the rvar, not the underlying array, so you cannot change the number of draws using this argument.

**dimnames**

(list) Character vectors giving the names in each dimension to override the names of the dimensions of the rvar to be created (see dimnames()). If NULL (the default), this is determined by the input. **NOTE:** This argument controls the names of the dimensions of the rvar, not the underlying array.

**nchains**

(positive integer) The number of chains. The if NULL (the default), 1 is used unless x is already an rvar, in which case the number of chains it has is used.

**with_chains**

(logical) Does x include a dimension for chains? If FALSE (the default), chains are not included, the first dimension of the input array should index draws, and the nchains argument can be used to determine the number of chains. If TRUE, the nchains argument is ignored and the second dimension of x is used to index chains. Internally, the array will be converted to a format without the chain index. Ignored when x is already an rvar.

... Arguments passed on to base::factor

**levels** an optional vector of the unique values (as character strings) that x might have taken. The default is the unique set of values taken by as.character(x), sorted into increasing order of x. Note that this set can be specified as smaller than sort(unique(x)).

**labels** either an optional character vector of labels for the levels (in the same order as levels after removing those in exclude), or a character string of length 1. Duplicated values in labels can be used to map different values of x to the same factor level.

**exclude** a vector of values to be excluded when forming the set of levels. This may be factor with the same level set as x or should be a character.

**ordered** logical flag to determine if the levels should be regarded as ordered (in the order given).

**nmax** an upper bound on the number of levels; see ‘Details’.

**Details**

A subtype of rvar() that represents a (possibly multidimensional) sample of a factor or an ordered factor. It is otherwise very similar to the basic rvar(): it is backed by a multidimensional array with draws as the first dimension. The primary difference is that the backing array has class "factor" (for rvar_factor()) or c("ordered", "factor") (for rvar_ordered()). If you pass a factor or ordered factor to rvar() it will automatically return an object with the classes "rvar_factor" or c("rvar_ordered", "rvar_factor").

See rvar() for more details on the internals of the random variable datatype.
Value

An object of class "rvar_factor" representing a factor-like random variable.

See Also

as_rvar_factor() to convert objects to rvar_factors. See rdo(), rfun(), and rvar_rng() for higher-level interfaces for creating rvars.

Examples

set.seed(1234)

# To create a "scalar" rvar_factor, pass a one-dimensional array or a vector
# whose length (here 4000) is the desired number of draws:
x <- rvar(sample(c("a","a","a","b","c"), 4000, replace = TRUE))
x

# Create random vectors by adding an additional dimension:
x_array <- array(c(
sample(c("a","a","a","b","c"), 4000, replace = TRUE),
sample(c("a","a","b","c","c"), 4000, replace = TRUE),
sample(c("b","b","b","b","c"), 4000, replace = TRUE),
sample(c("d","d","b","b","c"), 4000, replace = TRUE)
), dim = c(4000, 4))
rvar_factor(x_array)

# You can also create ordered factors
rvar_ordered(x)

# arguments of factor() and ordered() are passed down by the constructor
# e.g. we can reorder levels of an ordered factor:
rvar_ordered(x_array, levels = c("d","c","b","a"))

# Unlike base factors, rvar factors can be matrices or arrays:
rvar_factor(x_array, dim = c(2, 2))

# If the input to rvar_factor() is an array with a "levels" attribute, it
# will use those as the levels of the factor
y_array <- t(array(rbinom(3000, 1, c(0.1, 0.5, 0.9)) + 1, dim = c(3, 1000)))
rvar(y_array)
# with levels
attr(y_array, "levels") = c("a", "b")
rvar_factor(y_array)

rvar_is_finite Special value predicates for random variables
Description

Compute special value predicates (checking for finite / infinite values, NaN, and NA) on all draws within a random variable, returning a random variable.

Usage

rvar_is_finite(x)
rvar_is_infinite(x)
rvar_is_nan(x)
rvar_is_na(x)

Arguments

x (rvar) An rvar.

Details

These functions return a new rvar that is the result of applying is.finite(), is.infinite(), is.nan(), or is.na() to every draw in the input random variable.

Value

A logical rvar of the same length as the input.

See Also

rvar-summaries-over-draws for summary functions across draws, including implementations of is.finite(), is.infinite(), is.nan(), and is.na() for rvars.

Other rvar-summaries: rvar-summaries-over-draws, rvar-summaries-within-draws

Examples

x <- rvar(c(1, Inf, -Inf, NaN, NA))
x

rvar_is_finite(x)
rvar_is_infinite(x)
rvar_is_nan(x)
rvar_is_na(x)
rvar_rng

Create random variables from existing random number generators

Description

Specialized alternative to rdo() or rfun() for creating rvars from existing random-number generator functions (such as rnorm(), rbinom(), etc).

Usage

rvar_rng(.f, n, ..., ndraws = NULL)

Arguments

.f (function) A function (or string naming a function) representing a random-number generating function that follows the pattern of base random number generators (like rnorm(), rbinom(), etc). It must:

• Have a first argument, n, giving the number of draws to take from the distribution
• Have vectorized parameter arguments
• Return a single vector of length n

n (positive integer) The length of the output rvar vector (not the number of draws).

... Arguments passed to .f. These arguments may include rvars, so long as they are vectors only (no multidimensional rvars are allowed).

ndraws (positive integer) The number of draws used to construct the returned random variable if no rvars are supplied in .... If NULL,getOption("posterior.rvar_ndraws") is used (default 4000). If ... contains rvars, the number of draws in the provided rvars is used instead of the value of this argument.

Details

This function unwraps the arrays underlying the input rvars in ... and then passes them to .f, relying on the vectorization of .f to evaluate it across draws from the input rvars. This is why the arguments of .f must be vectorized. It asks for n times the number of draws in the input rvars (or ndraws if none are given) draws from the random number generator .f, then reshapes the output from .f into an rvar with length n.

rvar_rng() is a fast alternative to rdo() or rfun(), but you must ensure that .f satisfies the preconditions described above for the result to be correct. Most base random number generators satisfy these conditions. It is advisable to test against rdo() or rfun() (which should be correct, but slower) if you are uncertain.

Value

A single-dimensional rvar of length n.
split_chains

See Also

Other rfun: rdo(), rfun()

Examples

mu <- rvar_rng(rnorm, 10, mean = 1:10, sd = 1)
sigma <- rvar_rng(rgamma, 1, shape = 1, rate = 1)
x <- rvar_rng(rnorm, 10, mu, sigma)
x

split_chains

Split Chains

Description

Split chains by halving the number of iterations per chain and doubling the number of chains.

Usage

split_chains(x, ...)

Arguments

x (draws) A draws object or another R object for which the method is defined.
...
Arguments passed to individual methods (if applicable).

Value

A draws object of the same class as x.

Examples

x <- example_draws()
niterations(x)
nchains(x)

x <- split_chains(x)
niterations(x)
nchains(x)
subset_draws Subset draws objects

Description
Subset draws objects by variables, iterations, chains, and draws indices.

Usage
subset_draws(x, ...)

## S3 method for class 'draws_matrix'
subset_draws(
x,
  variable = NULL,
  iteration = NULL,
  chain = NULL,
  draw = NULL,
  regex = FALSE,
  unique = TRUE,
  ...
)

## S3 method for class 'draws_array'
subset_draws(
x,
  variable = NULL,
  iteration = NULL,
  chain = NULL,
  draw = NULL,
  regex = FALSE,
  unique = TRUE,
  ...
)

## S3 method for class 'draws_df'
subset_draws(
x,
  variable = NULL,
  iteration = NULL,
  chain = NULL,
  draw = NULL,
  regex = FALSE,
  unique = TRUE,
  ...
)
## S3 method for class 'draws_list'
subset_draws(
  x,
  variable = NULL,
  iteration = NULL,
  chain = NULL,
  draw = NULL,
  regex = FALSE,
  unique = TRUE,
  ...
)

## S3 method for class 'draws_rvars'
subset_draws(
  x,
  variable = NULL,
  iteration = NULL,
  chain = NULL,
  draw = NULL,
  regex = FALSE,
  unique = TRUE,
  ...
)

## S3 method for class 'rvar'
subset_draws(x, variable = NULL, ...)

## S3 method for class 'draws'
subset(x, ...)

### Arguments

- **x** (draws) A draws object or another R object for which the method is defined.
- **variable** (character vector) The variables to select. All elements of non-scalar variables can be selected at once.
- **iteration** (integer vector) The iteration indices to select.
- **chain** (integer vector) The chain indices to select.
- **draw** (integer vector) The draw indices to select. Subsetting draw indices will lead to an automatic merging of chains via `merge_chains`.
- **regex** (logical) Should variable should be treated as a (vector of) regular expressions? Any variable in x matching at least one of the regular expressions will be selected. Defaults to `FALSE`.
- **unique** (logical) Should duplicated selection of chains, iterations, or draws be allowed? If `TRUE` (the default) only unique chains, iterations, and draws are selected regardless of how often they appear in the respective selecting arguments.
To ensure that multiple consecutive subsetting operations work correctly, `subset()` repairs the draws object before and after subsetting.

**Value**

A draws object of the same class as `x`.

**Examples**

```r
x <- example_draws()
subset_draws(x, variable = c("mu", "tau"))
subset_draws(x, chain = 2)
subset_draws(x, iteration = 5:10, chain = 3:4)

# extract the first chain twice
subset_draws(x, chain = c(1, 1), unique = FALSE)

# extract all elements of 'theta'
subset_draws(x, variable = "theta")
```

**Description**

Thin draws objects to reduce their size and autocorrelation in the chains.

**Usage**

```r
thin_draws(x, thin, ...)
```

```r
## S3 method for class 'draws'
thin_draws(x, thin, ...)

## S3 method for class 'rvar'
thin_draws(x, thin, ...)
```

**Arguments**

- `x` (draws) A draws object or another R object for which the method is defined.
- `thin` (positive integer) The period for selecting draws.
- `...` Arguments passed to individual methods (if applicable).

**Value**

A draws object of the same class as `x`. 

| thin_draws | Thin draws objects |
weights.draws

Examples

x <- example_draws()
niterations(x)

x <- thin_draws(x, thin = 5)
niterations(x)

weights.draws  Extract Weights from Draws Objects

Description

Extract weights from draws objects, with one weight per draw. See weight_draws for details how to add weights to draws objects.

Usage

## S3 method for class 'draws'
weights(object, log = FALSE, normalize = TRUE, ...)

Arguments

object (draws) A draws object.
log (logical) Should the weights be returned on the log scale? Defaults to FALSE.
normalize (logical) Should the weights be normalized to sum to 1 on the standard scale? Defaults to TRUE.
...
Arguments passed to individual methods (if applicable).

Value

A vector of weights, with one weight per draw.

See Also

weight_draws, resample_draws

Examples

x <- example_draws()

# sample some random weights for illustration
wts <- rexp(ndraws(x))
head(wts)

# add weights
x <- weight_draws(x, weights = wts)
# extract weights
head(weights(x)) # defaults to normalized weights
head(weights(x, normalize=FALSE)) # recover original weights
head(weights(x, log=TRUE)) # get normalized log-weights

# add weights which are already on the log scale
log_wts <- log(wts)
head(log_wts)

x <- weight_draws(x, weights = log_wts, log = TRUE)
# extract weights
head(weights(x))
head(weights(x, log=TRUE, normalize = FALSE)) # recover original log_wts

---

**weight_draws**  
*Weight draws objects*

**Description**

Add weights to draws objects, with one weight per draw, for use in subsequent weighting operations. For reasons of numerical accuracy, weights are stored in the form of unnormalized log-weights (in a variable called `.log_weight`). See `weights.draws()` for details how to extract weights from draws objects.

**Usage**

weight_draws(x, weights, ...)

## S3 method for class 'draws_matrix'
weight_draws(x, weights, log = FALSE, ...)

## S3 method for class 'draws_array'
weight_draws(x, weights, log = FALSE, ...)

## S3 method for class 'draws_df'
weight_draws(x, weights, log = FALSE, ...)

## S3 method for class 'draws_list'
weight_draws(x, weights, log = FALSE, ...)

## S3 method for class 'draws_rvars'
weight_draws(x, weights, log = FALSE, ...)

**Arguments**

x  
(draws) A draws object or another R object for which the method is defined.
weights (numeric vector) A vector of weights of length `ndraws(x)`. Weights will be internally stored on the log scale (in a variable called `.log_weight`) and will not be normalized, but normalized (non-log) weights can be returned via the `weights.draws()` method later.

Arguments passed to individual methods (if applicable).

log (logical) Are the weights passed already on the log scale? The default is `FALSE`, that is, expecting weights to be on the standard (non-log) scale.

Value

A `draws` object of the same class as `x`.

See Also

`weights.draws()`, `resample_draws()`

Examples

```r
x <- example_draws()

# sample some random weights for illustration
wts <- rexp(ndraws(x))
head(wts)

# add weights
x <- weight_draws(x, weights = wts)

# extract weights
head(weights(x)) # defaults to normalized weights
head(weights(x, normalize=FALSE)) # recover original weights
head(weights(x, log=TRUE)) # get normalized log-weights

# add weights which are already on the log scale
log_wts <- log(wts)
head(log_wts)

x <- weight_draws(x, weights = log_wts, log = TRUE)
# extract weights
head(weights(x))
head(weights(x, log=TRUE, normalize = FALSE)) # recover original log_wts
```
Index

* diagnostics
  ess_basic, 29
  ess_bulk, 30
  ess_quantile, 33
  ess_sd, 34
  ess_tail, 35
  mcse_mean, 43
  mcse_quantile, 45
  mcse_sd, 46
  rhat, 68
  rhat_basic, 69
  rstar, 71
* formats
  draws, 12
  draws_array, 15
  draws_df, 16
  draws_list, 18
  draws_matrix, 20
  draws_rvars, 22
* rfun
  rdo, 61
  rfun, 67
  rvar_rng, 88
* rvar-summaries
  rvar-summaries-over-draws, 78
  rvar-summaries-within-draws, 81
  rvar_is_finite, 86
  %%%(rvar-matmult), 76
  %in%(match), 42
  all.rvar(rvar-summaries-over-draws), 78
  any.rvar(rvar-summaries-over-draws), 78
  as.character, 6, 85
  as.draws, 12
  as_draws_array(draws_array), 15
  as_draws_df(draws_df), 16
  as_draws_list(draws_list), 18
  as_draws_matrix(draws_matrix), 20
  as_draws_rvars(draws_rvars), 22
  as_function(), 25
  as_rvar, 4
  as_rvar(), 41, 74, 83
  as_rvar_factor, 6
  as_rvar_factor(), 42, 86
  as_rvar_ordered(as_rvar), 4
  as_rvar_ordered(as_rvar_factor), 6
  as_rvar_ordered(), 42
  base::diag(), 9
  base::drop(), 26
  base::factor, 6, 85
  base::match, 43
  base::match(), 42, 43
  bind_draws, 7
  cdf.rvar(rvar-dist), 75
  cdf.rvar_factor(rvar-dist), 75
  cdf.rvar_ordered(rvar-dist), 75
  chain_ids(draws-index), 13
  chol.rvar, 8
  convergence(diagnostics), 10
  default_convergence_measures(draws_summary), 24
  default_mcse_measures(draws_summary), 24
  default_summary_measures(draws_summary), 24
  density.rvar(rvar-dist), 75
  density.rvar_factor(rvar-dist), 75
  diag, rvar-method, 9
  diagnostic, 25
  diagnostics, 10, 26
  dim(), 5, 6, 61, 73, 85
  dimnames(), 5, 6, 73, 85
  dissent, 11
  dissent(), 56, 59
  draw_ids(draws-index), 13
  draws, 4, 7, 8, 12, 16, 17, 19, 21, 24, 47, 50, 51, 62, 64, 66, 90, 92–94
INDEX

draws_index, 13
draws_array, 13, 15, 17, 19, 21, 24, 37, 50, 52
draws_df, 13, 16, 19, 21, 24, 50, 53, 66, 71
draws_list, 13, 16, 17, 18, 21, 24, 50, 54
draws_matrix, 13, 16, 17, 19, 20, 24, 50, 55
draws_of, 21
draws_of(), 61, 68, 74
draws_of<- (draws_of), 21
draws_rvars, 13, 16, 17, 19, 21, 22, 40, 50, 56
draws_summary, 24
drop, rvar-method, 26

e(E(rvar_summaries_over_draws), 78
entropy(), 27
entropy(), 56, 58
ess_basic, 29, 31, 34–36, 44, 46, 47, 69, 70, 72
ess_basic(), 10
ess bulk, 30, 30, 34–36, 44, 46, 47, 69, 70, 72
ess bulk(), 10, 25, 29, 35
ess mean, 32
ess mean.default (ess_quantile), 33
ess median (ess_quantile), 33
ess quantile, 30, 31, 33, 35, 36, 44, 46, 47, 69, 70, 72
ess quantile(), 10
ess sd, 30, 31, 34, 34, 36, 44, 46, 47, 69, 70, 72
ess_sd(), 10
ess tail, 30, 31, 34, 35, 35, 44, 46, 47, 69, 70, 72
ess tail(), 10, 25, 29, 30
example_draws, 37
extract_variable, 38
extract_variable_matrix, 39
extract_variable_matrix(), 29, 31–34, 36, 44–46, 60, 69, 70

factor, 11, 27, 28, 49, 84, 85
for_each_draw, 40
format(), 59
format_rvar (print_rvar), 58

integer, 11, 27, 49
is finite_rvar
(rvar_summaries_over_draws), 78
is.infinite_rvar
(rvar_summaries_over_draws), 78

is.na_rvar (rvar_summaries_over_draws), 78
is.nan_rvar
(rvar_summaries_over_draws), 78
is_draws (draws), 12
is_draws_array (draws_array), 15
is_draws_df (draws_df), 16
is_draws_list (draws_list), 18
is_draws_matrix (draws_matrix), 20
is_draws_rvars (draws_rvars), 22
is_rvar, 41
is_rvar(), 61
is_rvar_factor, 42
is_rvar_ordered (is_rvar_factor), 42
iteration_ids (draws_index), 13

logical, 77

mad (rvar_summaries_over_draws), 78
mad(), 25
match, 42
matrix multiplication, 22
max_rvar (rvar_summaries_over_draws), 78
mcse_mean, 30, 31, 34–36, 43, 46, 47, 69, 70, 72
mcse_mean(), 10
mcse_quantile (mcse_quantile), 45
mcse_quantile, 30, 31, 34–36, 44, 45, 47, 69, 70, 72
mcse_quantile(), 10
mcse_sd, 30, 31, 34–36, 44, 46, 46, 69, 70, 72
mcse_sd(), 10
mean(), 25
mean_rvar (rvar_summaries_over_draws), 78
median(), 25
median_rvar
(rvar_summaries_over_draws), 78
merge_chains, 47, 91
min_rvar (rvar_summaries_over_draws), 78
modal_category, 48
mutate_variables, 50, 62

names, 9, 60
nchains (draws_index), 13
nndraws (draws_index), 13
niterations (draws_index), 13
num(), 25, 57
numeric, 5, 9, 11, 27, 28, 49, 77
nvariables (draws-index), 13

option, 25, 52–57
order_draws(), 51
order_draws(), 63, 64
ordered, 85

pillar::style_num(), 59
posterior (posterior-package), 3
posterior-package, 3
Pr (rvar-summaries-over-draws), 78
print(), 52–56, 58
print.draws_array, 52
print.draws_df, 53
print.draws_list, 54
print.draws_matrix, 55
print.draws_rvars, 56
print.draws_summary, 57
print.rvar, 58
prod.rvar (rvar-summaries-over-draws), 78
quantile.rvar (rvar-dist), 75
quantile.rvar_factor (rvar-dist), 75
quantile.rvar_ordered (rvar-dist), 75
quantile2(), 60
quantile2(), 25
quasiquotation, 40, 61
range.rvar (rvar-summaries-over-draws), 78
rdo, 61, 68, 89
rdo(), 5, 7, 74, 83, 86
rename_variables, 51, 62
rename_variables(), 14
repair_draws, 63
repair_draws(), 52
repairs, 92
resample_draws, 64, 93
resample_draws(), 65, 95
reserved_variables, 52–56, 66
rfun, 61, 67, 89
rfun(), 5, 7, 74, 83, 86
rhat, 30, 31, 34–36, 44, 46, 47, 68, 70, 72
rhat(), 10, 25, 39, 69
rhat_basic, 30, 31, 34–36, 44, 46, 47, 69, 69, 72
rhat_basic(), 10
rlang::as_function(), 67
rstar, 30, 31, 34–36, 44, 46, 47, 69, 70, 71
rstar(), 10
rvar(), 5, 7, 85
rvar-dist, 75, 80, 82
rvar-matmult, 76
rvar-summaries-over-draws, 78, 82, 87
rvar-summaries-within-draws, 80, 81
rvar_all (rvar-summaries-within-draws), 81
rvar_any (rvar-summaries-within-draws), 81
rvar_apply, 82
rvar_factor, 5–7, 11, 27, 42, 49, 56, 58, 59, 84
rvar_factor(), 7, 85
rvar_is_finite, 80, 82, 86
rvar_is_infinite (rvar_is_finite), 86
rvar_is_na (rvar_is_finite), 86
rvar_is_nan (rvar_is_finite), 86
rvar_max (rvar-summaries-within-draws), 81
rvar_mean
(rvar-summaries-within-draws), 81
rvar_median
(rvar-summaries-within-draws), 81
rvar_min (rvar-summaries-within-draws), 81
rvar_ordered, 5–7, 11, 27, 42, 49, 56, 58
rvar_ordered (rvar_factor), 84
rvar_ordered(), 7, 85
rvar_prod
(rvar-summaries-within-draws), 81
rvar_quantile
(rvar-summaries-within-draws), 81
rvar_range
(rvar-summaries-within-draws), 81
rvar_rng, 61, 68, 88
rvar_rng(), 5, 7, 61, 68, 74, 83, 86
INDEX

rvar_sd(rvar-summaries-within-draws), 81
rvar_sum(rvar-summaries-within-draws), 81
rvar_var(rvar-summaries-within-draws), 81

sd(rvar-summaries-over-draws), 78
sd(), 25
split_chains, 89
stats::density(), 76
stats::quantile(), 60, 76
str.rvar(print.rvar), 58
subset.draws(subset_draws), 90
subset_draws, 65, 90
sum.rvar(rvar-summaries-over-draws), 78
summarise_draws(draws_summary), 24
summarise_draws(), 57
summarize_draws(draws_summary), 24
summary(), 24
summary.draws(draws_summary), 24
Summary.rvar
         (rvar-summaries-over-draws), 78
summary.rvar(draws_summary), 24

thin(thin_draws), 92
thin_draws, 92
tibble, 17, 26
tibble::print.tbl_df(), 57

var(rvar-summaries-over-draws), 78
variables, 51, 62
variables(draws-index), 13
variables<- (draws-index), 13
variance.rvar
         (rvar-summaries-over-draws), 78

weight_draws, 66, 93, 94
weight_draws(), 64
weights.draws, 93
weights.draws(), 94, 95