Package ‘posterior’

July 4, 2024

Title Tools for Working with Posterior Distributions
Version 1.6.0
Date 2024-06-28

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.

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, ggdist, rmarkdown

License BSD_3_clause + file LICENSE

Encoding UTF-8


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

RoxygenNote 7.3.1

VignetteBuilder knitr

NeedsCompilation no

Author Paul-Christian Bürkner [aut, cre],
Jonah Gabry [aut],
Matthew Kay [aut],
Aki Vehtari [aut],
Måns Magnusson [ctb],
Rok Češnovar [ctb],
Ben Lambert [ctb],
Ozan Adıgüzel [ctb],
Jacob Socolar [ctb],
Noa Kallioinen [ctb]

Maintainer  Paul-Christian Bürkner <paul.buerkner@gmail.com>

Repository  CRAN

Date/Publication  2024-07-03 23:00:02 UTC

Contents

posterior-package ......................................................... 4
as_rvar ................................................................. 5
as_rvar_factor ......................................................... 7
bind_draws .............................................................. 8
chol.rvar ............................................................... 9
diag,rvar-method ..................................................... 10
diagnostics ............................................................ 11
dissent ................................................................. 12
draws ................................................................. 13
draws-index ............................................................ 14
draws_array ............................................................. 15
draws_df ............................................................... 17
draws_list .............................................................. 19
draws_matrix ........................................................... 20
draws_of ............................................................. 22
draws_rvars ............................................................ 23
draws_summary ........................................................... 25
drop,rvar-method .......................................................... 27
entropy ................................................................. 28
ess_basic ............................................................... 29
ess_bulk ............................................................... 31
ess_mean ............................................................... 32
ess_quantile ............................................................ 33
ess_sd ................................................................. 34
ess_tail .............................................................. 36
example_draws .......................................................... 37
extract_variable ......................................................... 38
extract_variable_array .................................................. 39
extract_variable_matrix ................................................ 40
for_each_draw ........................................................... 41
is_rvar ............................................................... 43
is_rvar_factor .......................................................... 43
match ................................................................. 44
mcse_mean ............................................................ 45
<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>mcse_quantile</td>
<td>46</td>
</tr>
<tr>
<td>mcse_sd</td>
<td>47</td>
</tr>
<tr>
<td>merge_chains</td>
<td>49</td>
</tr>
<tr>
<td>modal_category</td>
<td>50</td>
</tr>
<tr>
<td>mutate_variables</td>
<td>51</td>
</tr>
<tr>
<td>order_draws</td>
<td>52</td>
</tr>
<tr>
<td>pareto_diags</td>
<td>53</td>
</tr>
<tr>
<td>pareto_khat</td>
<td>56</td>
</tr>
<tr>
<td>pareto_smooth</td>
<td>58</td>
</tr>
<tr>
<td>print.draws_array</td>
<td>60</td>
</tr>
<tr>
<td>print.draws_df</td>
<td>61</td>
</tr>
<tr>
<td>print.draws_list</td>
<td>62</td>
</tr>
<tr>
<td>print.draws_matrix</td>
<td>63</td>
</tr>
<tr>
<td>print.draws_rvars</td>
<td>64</td>
</tr>
<tr>
<td>print.draws_summary</td>
<td>65</td>
</tr>
<tr>
<td>print.rvar</td>
<td>66</td>
</tr>
<tr>
<td>ps_convergence_rate</td>
<td>68</td>
</tr>
<tr>
<td>ps_khat_threshold</td>
<td>68</td>
</tr>
<tr>
<td>ps_min_ss</td>
<td>69</td>
</tr>
<tr>
<td>ps_tail_length</td>
<td>70</td>
</tr>
<tr>
<td>quantile2</td>
<td>70</td>
</tr>
<tr>
<td>rdo</td>
<td>71</td>
</tr>
<tr>
<td>rename_variables</td>
<td>72</td>
</tr>
<tr>
<td>repair_draws</td>
<td>73</td>
</tr>
<tr>
<td>resample_draws</td>
<td>74</td>
</tr>
<tr>
<td>reserved_variables</td>
<td>76</td>
</tr>
<tr>
<td>rfun</td>
<td>77</td>
</tr>
<tr>
<td>rhat</td>
<td>79</td>
</tr>
<tr>
<td>rhat_basic</td>
<td>80</td>
</tr>
<tr>
<td>rhat_nested</td>
<td>81</td>
</tr>
<tr>
<td>rstar</td>
<td>83</td>
</tr>
<tr>
<td>rvar</td>
<td>85</td>
</tr>
<tr>
<td>rvar_dist</td>
<td>87</td>
</tr>
<tr>
<td>rvar_matmult</td>
<td>89</td>
</tr>
<tr>
<td>rvar_slice</td>
<td>90</td>
</tr>
<tr>
<td>rvar_summaries_over_draws</td>
<td>92</td>
</tr>
<tr>
<td>rvar_summaries_within_draws</td>
<td>96</td>
</tr>
<tr>
<td>rvar_apply</td>
<td>97</td>
</tr>
<tr>
<td>rvar_factor</td>
<td>99</td>
</tr>
<tr>
<td>rvar_ifelse</td>
<td>101</td>
</tr>
<tr>
<td>rvar_is_finite</td>
<td>102</td>
</tr>
<tr>
<td>rvar_rng</td>
<td>103</td>
</tr>
<tr>
<td>split_chains</td>
<td>104</td>
</tr>
<tr>
<td>subset_draws</td>
<td>105</td>
</tr>
<tr>
<td>thin_draws</td>
<td>108</td>
</tr>
<tr>
<td>variables</td>
<td>109</td>
</tr>
<tr>
<td>variables&lt;</td>
<td>110</td>
</tr>
<tr>
<td>weights.draws</td>
<td>111</td>
</tr>
</tbody>
</table>
Description

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 for formatting the output of `summarize_draws`:

- `posterior.num_args`: Arguments passed to `num()` for pretty printing of summaries.

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) than `getOption("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

**Author(s)**

**Maintainer**: Paul-Christian Bürkner <paul.buerkner@gmail.com>

Authors:

- Jonah Gabry <jsg201@columbia.edu>
- Matthew Kay <mjskay@northwestern.edu>
- Aki Vehtari <Aki.Vehtari@aalto.fi>

Other contributors:

- Måns Magnusson [contributor]
- Rok Češnovar [contributor]
- Ben Lambert [contributor]
- Ozan Adığüzel [contributor]
- Jacob Socolar [contributor]
- Noa Kallioinen [contributor]

**See Also**

Useful links:

- [https://mc-stan.org/posterior/](https://mc-stan.org/posterior/)
- [https://discourse.mc-stan.org/](https://discourse.mc-stan.org/)
- Report bugs at [https://github.com/stan-dev/posterior/issues](https://github.com/stan-dev/posterior/issues)

---

**as_rvar**

*Coerce to a random variable*

**Description**

Convert x to an rvar object.

**Usage**

```r
as_rvar(x, dim = NULL, dimnames = NULL, nchains = NULL)
```

```r
as_rvar_numeric(x, dim = NULL, dimnames = NULL, nchains = NULL)
```

```r
as_rvar_integer(x, dim = NULL, dimnames = NULL, nchains = NULL)
```

```r
as_rvar_logical(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 an rvar_factor or rvar_ordered), as_rvar_numeric(), as_rvar_integer(), and as_rvar_logical() always coerce the draws of the output rvar to be numeric, integer, or logical (respectively), and always return 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

as_rvar_numeric(y)

---

**as_rvar_factor**

*Coerce to a factor random variable*

**Description**

Convert `x` to an `rvar_factor` or `rvar_ordered` object.

**Usage**

```r
as_rvar_factor(x, dim = NULL, dimnames = NULL, nchains = NULL, ...)
```

```r
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` *(optional)*: 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` *(vector)*: 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)*: Logical flag to determine if the levels should be regarded as ordered (in the order given).
- `nmax` *(optional)*: An upper bound on the number of levels; see ‘Details’.
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")
## Arguments

**x**  
(draws) A draws object. The draws format of x will define the format of the returned draws object.

**...**  
(draws) Additional draws objects to bind to x.

**along**  
(string) The dimension along which draws objects should be bound together. Possible values are "variable" (the default), "chain", "iteration", and "draw". Not all options are supported for all input formats.

## Value

A draws object of the same class as x.

## Examples

```r
x1 <- draws_matrix(alpha = rnorm(5), beta = rnorm(5))
x2 <- draws_matrix(alpha = rnorm(5), beta = rnorm(5))
ndraws(x1)
ndraws(x2)
x3 <- bind_draws(x1, x2, along = "draw")
ndraws(x3)

x4 <- draws_matrix(theta = rexp(5))
x5 <- bind_draws(x1, x4, along = "variable")
variables(x5)
```

---

### chol.rvar

**Cholesky decomposition of random matrix**

**Description**

Cholesky decomposition of an rvar containing a matrix.

**Usage**

```r
## 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

```r
# 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><code>ess_basic()</code></td>
<td>Basic version of effective sample size</td>
</tr>
<tr>
<td><code>ess_bulk()</code></td>
<td>Bulk effective sample size</td>
</tr>
<tr>
<td><code>ess_tail()</code></td>
<td>Tail effective sample size</td>
</tr>
<tr>
<td><code>ess_quantile()</code></td>
<td>Effective sample sizes for quantiles</td>
</tr>
<tr>
<td><code>ess_sd()</code></td>
<td>Effective sample sizes for standard deviations</td>
</tr>
<tr>
<td><code>mcse_mean()</code></td>
<td>Monte Carlo standard error for the mean</td>
</tr>
<tr>
<td><code>mcse_quantile()</code></td>
<td>Monte Carlo standard error for quantiles</td>
</tr>
<tr>
<td><code>mcse_sd()</code></td>
<td>Monte Carlo standard error for standard deviations</td>
</tr>
<tr>
<td><code>pareto_khat()</code></td>
<td>Pareto khat diagnostic for tail(s)</td>
</tr>
<tr>
<td><code>pareto_diags()</code></td>
<td>Additional diagnostics related to Pareto khat</td>
</tr>
<tr>
<td><code>rhat_basic()</code></td>
<td>Basic version of Rhat</td>
</tr>
<tr>
<td><code>rhat()</code></td>
<td>Improved, rank-based version of Rhat</td>
</tr>
<tr>
<td><code>rhat_nested()</code></td>
<td>Rhat for use with many short chains</td>
</tr>
<tr>
<td><code>rstar()</code></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

```r
ds <- 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)

# 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)

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

# dissent measures
dissent(x)
dissent(y)
dissent(xy)
```

---

**draws**

*Transform to draws objects*

---

**Description**

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

**Usage**

```r
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

# 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 iterations, chains, and draws of draws objects.

Usage

iteration_ids(x)

chain_ids(x)

draw_ids(x)

niterations(x)

nchains(x)

ndraws(x)

Arguments

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

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

Value

For `iteration_ids()`, `chain_ids()`, and `draw_ids()`, an integer vector.
For `niterations()`, `nchains()`, and `ndraws()`, a scalar integer.

See Also

`variables`, `rename_variables`

Examples

```r
x <- example_draws()
iteration_ids(x)
niterations(x)
chain_ids(x)
nchains(x)
draw_ids(x)
ndraws(x)
```

---

draws_array

The draws_array format

Description

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
as_draws_array(x, ...)
```

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

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

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

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

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

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

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

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

draws_array(..., .nchains = 1)

is_draws_array(x)

Arguments

x      An object to convert to a draws_array object.
...
.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, draws_df(), draws_list(), draws_matrix(), draws_rvars()

Examples

x1 <- as_draws_array(example_draws())
class(x1)
print(x1)
str(x1)
```r
dx2 <- draws_array(a = rnorm(10), b = rnorm(10), c = 1)
class(x2)
print(x2)
str(x2)
```

---

### 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

```r
as_draws_df(x, ...)
```

`## Default S3 method:`

```r
as_draws_df(x, ...)
```

`## S3 method for class 'data.frame'`

```r
as_draws_df(x, ...)
```

`## S3 method for class 'draws_df'`

```r
as_draws_df(x, ...)
```

`## S3 method for class 'draws_matrix'`

```r
as_draws_df(x, ...)
```

`## S3 method for class 'draws_array'`

```r
as_draws_df(x, ...)
```

`## S3 method for class 'draws_list'`

```r
as_draws_df(x, ...)
```

`## S3 method for class 'draws_rvars'`

```r
as_draws_df(x, ...)
```

`## S3 method for class 'mcmc'`

```r
as_draws_df(x, ...)
```

`## S3 method for class 'mcmc.list'`

```r
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, draws_array(), draws_list(), draws_matrix(), draws_rvars()

Examples

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)
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, draws_array(), draws_df(), draws_matrix(), draws_rvars()

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

The draws_matrix format

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'
as_draws_matrix(x, ...)

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.

.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, draws_array(), draws_df(), draws_list(), draws_rvars()

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

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.

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`, `draws_array()`, `draws_df()`, `draws_list()`, `draws_matrix()`

### Examples

```
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 summaries

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, ...)
```

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

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

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

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

```r
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.
draws_summary

% .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

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_*_measures()
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)
```

**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]

<table>
<thead>
<tr>
<th>entropy</th>
<th>Normalized entropy</th>
</tr>
</thead>
</table>

**Description**

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

**Usage**

```r
entropy(x)
```

## Default S3 method:

```r
entropy(x)
```

## S3 method for class 'rvar'

```r
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:

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

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)
```

---

**ess_basic**

*Basic version of the effective sample size*

**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()`, `pareto_diags()`, `pareto_khat()`, `rhat()`, `rhat_basic()`, `rhat_nested()`, `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)
```
**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**

```
ess_bulk(x, 
## Default S3 method:
ess_bulk(x, 
## S3 method for class 'rvar'
ess_bulk(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_quantile(), ess_sd(), ess_tail(), mcse_mean(), mcse_quantile(), mcse_sd(), pareto_diags(), pareto_khat(), rhat(), rhat_basic(), rhat_nested(), rstar()

Examples
mu <- extract_variable_matrix(example_draws(), "mu")
ess_bulk(mu)

d <- as_draws_rvars(example_draws("multi_normal"))
ess_bulk(d$Sigma)

---

**ess_mean**

*Effective sample size for the mean*

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

**Usage**
```r
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)
```

**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

Rank-normalization, folding, and localization: An improved R-hat for assessing convergence of
MCMC (with discussion). *Bayesian Data Analysis*. 16(2), 667—718. doi:10.1214/20-BA1221

See Also

Other diagnostics: ess_basic(), ess_bulk(), ess_sd(), ess_tail(), mcse_mean(), mcse_quantile(),
mcse_sd(), pareto_diags(), pareto_khat(), rhat(), rhat_basic(), rhat_nested(), 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:
```
```
## S3 method for class 'rvar'

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()`, `pareto_diags()`, `pareto_khat()`, `rhat()`, `rhat_basic()`, `rhat_nested()`, `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

```r
ess_tail(x, ...)
```

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

## S3 method for class 'rvar'
```r
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()`, `pareto_diags()`, `pareto_khat()`, `rhat()`, `rhat_basic()`, `rhat_nested()`, `rstar()

Examples

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

d <- as_draws_rvars(example_draws("multi_normal"))
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
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_df'
extract_variable(x, variable, ...)

## S3 method for class 'draws_list'
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. Must include indices for array variables (e.g. "x[1]", "y[1,2]"). To extract all dimensions from variables with indices, use `extract_variable_array()`.
- ... Arguments passed to individual methods (if applicable).

Value

A vector of length equal to the number of draws.

See Also

Other variable extraction methods: `extract_variable_array()`, `extract_variable_matrix()`

Examples

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

---

### `extract_variable_array`

**Extract array of a single (possibly indexed) variable**

Description

Extract an array of draws of a single variable, including any dimensions of variables with indices.

Usage

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

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

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

## S3 method for class 'draws'
`extract_variable_array(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. To extract all dimensions from variables with indices (e.g. "x[1]"), provide the base variable name (e.g. "x").
- ... Arguments passed to individual methods (if applicable).
**Value**

An array with dimension \( \text{dim}(\text{variable}) \) for each variable in the list.

**See Also**

Other variable extraction methods: `extract_variable()`, `extract_variable_matrix()`.

**Examples**

```r
x <- example_draws(example = "multi_normal")
mu <- extract_variable_array(x, variable = "mu")
str(mu)

mu1 <- extract_variable_array(x, variable = "mu[1]")
str(mu1)

Sigma <- extract_variable_array(x, variable = "Sigma")
str(Sigma)
```

---

**extract_variable_matrix**

*Extract matrix of a single variable*

**Description**

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

**Usage**

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

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

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

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

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

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

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. Must include indices for array variables (e.g. "x[1]", "y[1,2]"). To extract all dimensions from variables with indices, use `extract_variable_array()`.
- ... Arguments passed to individual methods (if applicable).

Value

A matrix with dimension iterations x chains.

See Also

Other variable extraction methods: `extract_variable()`, `extract_variable_array()`

Examples

```r
x <- example_draws()
mu <- extract_variable_matrix(x, variable = "mu")
dim(mu)
rhat(mu)
```

---

### 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**

```r
for_each_draw(x, expr)
```

**Arguments**

- **x** (draws) A `draws` object or another R object for which the method is defined.
- **expr** (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.
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 `ndraws(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

```r
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
is_rvar_factor(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.
**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:
```r
match(x, ...)
```

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

```r
x %in% table
```

**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`. 
mcse_mean

Examples

```r
x <- rvar(c("a","b","b","c","d"))
%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**

```r
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**

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.
Monte Carlo standard error for the standard deviation

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
mcse_sd(x, ...)

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

## S3 method for class 'rvar'
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()`, `pareto_diags()`, `pareto_khat()`, `rhat()`, `rhat_basic()`, `rhat_nested()`, `rstar()`

Examples

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

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

mcse_sd(d$Sigma)

merge_chains  Merge chains of draws objects

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

<table>
<thead>
<tr>
<th>modal_category</th>
<th>Modal category</th>
</tr>
</thead>
</table>

Description

Modal category of a vector.

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**

```r
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)
```

**Description**

Mutate variables in a `draws` object.

**Usage**

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

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

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

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

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

## S3 method for class 'draws_rvars'
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.
order_draws

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 ....

See Also

variables, rename_variables

Examples

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

Description

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

order_draws(x, ...)

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

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

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

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

## S3 method for class 'rvar'
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).

### Value

A draws object of the same class as x.

### See Also

- repair_draws()

### Examples

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

---

## pareto_diags

*Pareto smoothing diagnostics*

### Description

Compute diagnostics for Pareto smoothing the tail draws of x by replacing tail draws by order statistics of a generalized Pareto distribution fit to the tail(s).

### Usage

```r
pareto_diags(x, ...)
```

### Default S3 method:
```r
default_function(x, tail = c("both", "right", "left"), r_eff = NULL, ndraws_tail = NULL, verbose = FALSE)
```
pareto_diags

generate_corner_plot(x, ...)
verbose

(logical) Should diagnostic messages be printed? If TRUE, messages related to Pareto diagnostics will be printed. Default is FALSE.

are_log_weights

(logical) Are the draws log weights? Default is FALSE. If TRUE computation will take into account that the draws are log weights, and only right tail will be smoothed.

Details

When the fitted Generalized Pareto Distribution is used to smooth the tail values and these smoothed values are used to compute expectations, the following diagnostics can give further information about the reliability of these estimates.

- **min_ss**: Minimum sample size for reliable Pareto smoothed estimate. If the actual sample size is greater than \( \text{min}_\text{ss} \), then Pareto smoothed estimates can be considered reliable. If the actual sample size is lower than \( \text{min}_\text{ss} \), increasing the sample size might result in more reliable estimates. For further details, see Section 3.2.3, Equation 11 in Vehtari et al. (2024).

- **khat_threshold**: Threshold below which k-hat values result in reliable Pareto smoothed estimates. The threshold is lower for smaller effective sample sizes. If k-hat is larger than the threshold, increasing the total sample size may improve reliability of estimates. For further details, see Section 3.2.4, Equation 13 in Vehtari et al. (2024).

- **convergence_rate**: Relative convergence rate compared to the central limit theorem. Applicable only if the actual sample size is sufficiently large (greater than \( \text{min}_\text{ss} \)). The convergence rate tells the rate at which the variance of an estimate reduces when the sample size is increased, compared to the central limit theorem convergence rate. See Appendix B in Vehtari et al. (2024).

Value

List of Pareto smoothing diagnostics:

- **khat**: estimated Pareto k shape parameter,
- **min_ss**: minimum sample size for reliable Pareto smoothed estimate,
- **khat_threshold**: khat-threshold for reliable Pareto smoothed estimate,
- **convergence_rate**: Pareto smoothed estimate RMSE convergence rate.

References


See Also

pareto_khat, pareto_min_ss, pareto_khat_threshold, and pareto_convergence_rate for individual diagnostics; and pareto_smooth for Pareto smoothing draws.

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

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

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

---

**pareto_khat**

**Pareto khat diagnostic**

**Description**

Estimate Pareto k value by fitting a Generalized Pareto Distribution to one or two tails of x. This can be used to estimate the number of fractional moments that is useful for convergence diagnostics. For further details see Vehtari et al. (2024).

**Usage**

```r
pareto_khat(x, ...)
```

## Default S3 method:
```r
d# Default S3 method:
dpareto_khat(
  x,
  tail = c("both", "right", "left"),
  r_eff = NULL,
  ndraws_tail = NULL,
  verbose = FALSE,
  are_log_weights = FALSE,
  ...
)
```

## S3 method for class 'rvar'
```r
dpareto_khat(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).

- **tail**
  (string) The tail to diagnose/smooth:
  - "right": diagnose/smooth only the right (upper) tail
  - "left": diagnose/smooth only the left (lower) tail
  - "both": diagnose/smooth both tails and return the maximum k-hat value
  The default is "both".
**pareto_khat**

- **r_eff** (numeric) relative effective sample size estimate. If r_eff is NULL, it will be calculated assuming the draws are from MCMC. Default is NULL.
- **ndraws_tail** (numeric) number of draws for the tail. If ndraws_tail is not specified, it will be calculated as ceiling(3 * sqrt(length(x) / r_eff)) if length(x) > 225 and length(x) / 5 otherwise (see Appendix H in Vehtari et al. (2024)).
- **verbose** (logical) Should diagnostic messages be printed? If TRUE, messages related to Pareto diagnostics will be printed. Default is FALSE.
- **are_log_weights** (logical) Are the draws log weights? Default is FALSE. If TRUE computation will take into account that the draws are log weights, and only right tail will be smoothed.

**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**

- pareto_diags for additional related diagnostics, and pareto_smooth for Pareto smoothed draws.
- Other diagnostics: ess_basic(), ess_bulk(), ess_quantile(), ess_sd(), ess_tail(), mcse_mean(), mcse_quantile(), mcse_sd(), pareto_diags(), rhat(), rhat_basic(), rhat_nested(), rstar()

**Examples**

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

d <- as_draws_rvars(example_draws("multi_normal"))
pareto_khat(d$Sigma)
```
Pareto smoothing

Description

Smooth the tail draws of \( x \) by replacing tail draws by order statistics of a generalized Pareto distribution fit to the tail(s). For further details see Vehtari et al. (2024).

Usage

\[
\text{pareto_smooth}(x, \ldots)
\]

## S3 method for class 'rvar'
\[
\text{pareto_smooth}(x, \text{return\_k = FALSE}, \text{extra\_diags = FALSE}, \ldots)
\]

## Default S3 method:
\[
\text{pareto_smooth}(x, \text{tail = c("both", "right", "left")},
\text{r\_eff = NULL},
\text{ndraws\_tail = NULL},
\text{return\_k = FALSE},
\text{extra\_diags = FALSE},
\text{verbose = TRUE},
\text{are\_log\_weights = FALSE},
\ldots
\)

Arguments

- \( x \) (multiple options) One of:
  - A matrix of draws for a single variable (iterations x chains). See \texttt{extract\_variable\_matrix()}.
  - An \texttt{rvar}.
- \ldots Arguments passed to individual methods (if applicable).
- \text{return\_k} (logical) Should the Pareto khat be included in output? If TRUE, output will be a list containing smoothed draws and diagnostics, otherwise it will be a numeric of the smoothed draws. Default is FALSE.
- \text{extra\_diags} (logical) Should extra Pareto khat diagnostics be included in output? If TRUE, \text{min\_ss}, \text{khat\_threshold} and \text{convergence\_rate} for the estimated k value will be returned. Default is FALSE.
- \text{tail} (string) The tail to diagnose/smooth:
  - "right": diagnose/smooth only the right (upper) tail
  - "left": diagnose/smooth only the left (lower) tail
  - "both": diagnose/smooth both tails and return the maximum k-hat value
The default is "both".

- **r_eff** (numeric) relative effective sample size estimate. If `r_eff` is NULL, it will be calculated assuming the draws are from MCMC. Default is NULL.

- **ndraws_tail** (numeric) number of draws for the tail. If `ndraws_tail` is not specified, it will be calculated as ceiling(3 * sqrt(length(x) / r_eff)) if length(x) > 225 and length(x) / 5 otherwise (see Appendix H in Vehtari et al. (2024)).

- **verbose** (logical) Should diagnostic messages be printed? If TRUE, messages related to Pareto diagnostics will be printed. Default is FALSE.

- **are_log_weights** (logical) Are the draws log weights? Default is FALSE. If TRUE computation will take into account that the draws are log weights, and only right tail will be smoothed.

**Value**

Either a vector `x` of smoothed values or a named list containing the vector `x` and a named list diagnostics containing numeric values:

- **khat**: estimated Pareto k shape parameter, and optionally
- **min_ss**: minimum sample size for reliable Pareto smoothed estimate
- **khat_threshold**: sample size specific khat threshold for reliable Pareto smoothed estimates
- **convergence_rate**: Relative convergence rate for Pareto smoothed estimates

If any of the draws is non-finite, that is, NA, NaN, Inf, or -Inf, Pareto smoothing will not be performed, and the original draws will be returned and and diagnostics will be NA (numeric).

**References**


**See Also**

- `pareto_khat` for only calculating khat, and `pareto_diags` for additional diagnostics.

**Examples**

```r
cmu <- extract_variable_matrix(example_draws(), "mu")
pareto_smooth(cmu)

d <- as_draws_rvars(example_draws("multi_normal"))
pareto_smooth(d$Sigma)
```
print.draws_array  

**print.draws_array**  

*Print draws_array objects*

### Description

Pretty printing for draws_array objects.

### Usage

```r
## 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.

### Value

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

### Examples

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

Examples

```r
x <- as_draws_df(example_draws())
print(x)
```
Description
Pretty printing for draws_list objects.

Usage

```r
## 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)
```
print.draws_matrix  Print draws_matrix objects

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)
```
### Description

Pretty printing for `draws_rvars` objects.

### Usage

```r
def print_draws_rvars(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.

### Value

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

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))
```
Print or format a random variable

Description

Printing and formatting methods for rvars.

Usage

## 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.
...
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)
print.rvar

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

```r
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")

str(x)
```
Description

Given number of draws and scalar or array of k’s, compute the relative convergence rate of PSIS estimate RMSE. See Appendix B of Vehtari et al. (2024). This function is exported to be usable by other packages. For user-facing diagnostic functions, see `pareto_convergence_rate` and `pareto_diags`.

Usage

```r
ps_convergence_rate(k, ndraws, ...)
```

Arguments

- `k`: pareto-k values
- `ndraws`: number of draws
- `...`: unused

Value

- convergence rate

See Also

Other helper-functions: `ps_khat_threshold()`, `ps_min_ss()`, `ps_tail_length()`

Descriptio

Given number of draws, computes khat threshold for reliable Pareto smoothed estimate (to have small probability of large error). See section 3.2.4, equation (13) of Vehtari et al. (2024). This function is exported to be usable by other packages. For user-facing diagnostic functions, see `pareto_khat_threshold` and `pareto_diags`.

Usage

```r
ps_khat_threshold(ndraws, ...)
```
**ps_min_ss**

**Arguments**
- `ndraws` number of draws
- ... unused

**Value**
- threshold

**See Also**
Other helper-functions: `ps_convergence_rate()`, `ps_min_ss()`, `ps_tail_length()`

---

**Description**
Given Pareto-k computes the minimum sample size for reliable Pareto smoothed estimate (to have small probability of large error). See section 3.2.3 in Vehtari et al. (2024). This function is exported to be usable by other packages. For user-facing diagnostic functions, see `pareto_min_ss` and `pareto_diags`.

**Usage**
`ps_min_ss(k, ...)`

**Arguments**
- `k` pareto k value
- ... unused

**Value**
minimum sample size

**See Also**
Other helper-functions: `ps_convergence_rate()`, `ps_khat_threshold()`, `ps_tail_length()`
**ps_tail_length**  
*Pareto tail length*

**Description**
Calculate the tail length from number of draws and relative efficiency `r_eff`. See Appendix H in Vehtari et al. (2024). This function is used internally and is exported to be available for other packages.

**Usage**

```r
ps_tail_length(ndraws, r_eff, ...)  
```

**Arguments**
- `ndraws`  
  number of draws
- `r_eff`  
  relative efficiency
- `...`  
  unused

**Value**

tail length

**See Also**

Other helper-functions: `ps_convergence_rate()`, `ps_khat_threshold()`, `ps_min_ss()`

---

**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, ...)  
```

**Usage**

```r
quantile2(x, probs = c(0.05, 0.95), na.rm = FALSE, names = TRUE, ...)  
```

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

```r
# 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 \( \mathbf{R} \) expressions that may include rvars, producing a new rvar.

**Usage**

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

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

rename_variables  

Rename variables in draws objects

Description
Rename variables in a draws object.

Usage
rename_variables(.x, ...)  

## S3 method for class 'draws'
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`, `variables<-`, `mutate_variables`

Examples

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

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

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

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

Description

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

Usage

`repair_draws(x, order = TRUE, ...)`

---

`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**

`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`.

### See Also

- `order_draws()`

### Examples

```r
x <- as_draws_array(example_draws())
(x <- x[10:5, 3:4, ])
repair_draws(x)
```

---

### Description

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

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

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 just 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

Reserved variables

Description

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

Usage

```r
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, ...)

```
rfun

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

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

rfun(.f, rvar_args = NULL, rvar_dots = TRUE, 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, including arguments passed via ... (if rvar_dots is TRUE).

rvar_dots (logical) Should dots (…) arguments also be converted? Only applies if rvar_args is NULL (i.e., all arguments are allowed to be 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

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)
rhat

```r
x <- rvar_norm(10, mu, sigma)
x
```

---

**rhat**

<table>
<thead>
<tr>
<th><strong>Rhat convergence diagnostic</strong></th>
</tr>
</thead>
</table>

**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**
```
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()`, `pareto_diags()`, `pareto_khat()`, `rhat_basic()`, `rhat_nested()`, `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**

```r
rhat_basic(x, ...)
```

### Default S3 method:

```r
rhat_basic(x, split = TRUE, ...)
```

### S3 method for class 'rvar'

```r
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(), pareto_diags(), pareto_khat(), rhat(), rhat_nested(), rstar()

Examples

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

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

---

**rhat_nested**

*Nested Rhat convergence diagnostic*

Description

Compute the nested Rhat convergence diagnostic for a single variable as proposed in Margossian et al. (2023).
Usage

```r
rhat_nested(x, ...)
```

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

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

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).
- **superchain_ids**  
  (numeric) Vector of length `nchains` specifying which superchain each chain belongs to. There should be equal numbers of chains in each superchain. All chains within the same superchain are assumed to have been initialized at the same point.

Details

Nested Rhat is a convergence diagnostic useful when running many short chains. It is calculated on superchains, which are groups of chains that have been initialized at the same point.

Note that there is a slight difference in the calculation of Rhat and nested Rhat, as nested Rhat is lower bounded by 1. This means that nested Rhat with one chain per superchain will not be exactly equal to basic Rhat (see Footnote 1 in Margossian et al. (2023)).

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(), pareto_diags(), pareto_khat(), rhat(), rhat_basic(), rstar()

Examples

mu <- extract_variable_matrix(example_draws(), "mu")
rhat_nested(mu, superchain_ids = c(1, 1, 2, 2))

d <- as_draws_rvars(example_draws("multi_normal"))
rhat_nested(d$Sigma, superchain_ids = c(1, 1, 2, 2))

rstar

Calculate R* convergence diagnostic

Description

The 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.

Usage

rstar(
  x, 
  split = TRUE,
  uncertainty = FALSE,
  method = "rf",
  hyperparameters = NULL,
  training_proportion = 0.7,
  nsimulations = 1000,
  ...
)

Arguments

x (draws) A draws_df object or one coercible to a draws_df object.
split (logical) Should the estimate be computed on split chains? The default is TRUE.
uncertainty (logical). Indicates whether to provide a vector of R* values representing uncertainty in the calculated value (if TRUE) or a single value (if FALSE). The default is TRUE.
method (string) The machine learning classifier to use (must be available in the caret package). The default is "rf", which calls the random forest classifier.
hyperparameters
(named list) Hyperparameter settings passed to the classifier. The default for the random forest classifier (method = "rf") is list(mtry = floor(sqrt(nvariables(x))). The default for the gradient-based model (method = "gbm") is list(interaction.depth = 3, n.trees = 50, shrinkage = 0.1, n.minobsinnode = 10).

training_proportion
(positive real) The proportion (in (0,1)) of iterations in used to train the classifier. The default is 0.7.

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(), pareto_diags(), pareto_khat(), rhat(), rhat_basic(), rhat_nested()
Examples

```r
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 \texttt{rvar} to be created (see \texttt{dim()}). If \texttt{NULL} (the default), \texttt{dim} is determined by the input. \textbf{NOTE:} This argument controls the dimensions of the \texttt{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 \texttt{rvar} to be created (see \texttt{dimnames()}). If \texttt{NULL} (the default), this is determined by the input. \textbf{NOTE:} This argument controls the names of the dimensions of the \texttt{rvar}, not the underlying array.

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

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

Details

The "\texttt{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 \texttt{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 \texttt{[} is \texttt{FALSE}, not \texttt{TRUE}).
- Rather than base R-style recycling, \texttt{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 \texttt{rvars} cannot be used directly as arguments, you can use \texttt{rfun()} or \texttt{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 \texttt{rdo()} offers the most straightforward translation.

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

Value

An object of class "\texttt{rvar}" representing a random variable.

See Also

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

Examples

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
x <- rvar(array(rnorm(4000 * n, mean = rep(1:n, each = 4000), sd = 1), dim = c(4000, n)))
x

# Create a random matrix:
rows <- 4
cols <- 3
x <- rvar(array(rnorm(4000 * rows * cols, mean = 1, sd = 1), dim = c(4000, rows, cols)))
x

# If the input sample comes from multiple chains, we can indicate that using the
# nchains argument (here, 1000 draws each from 4 chains):
x <- rvar(rnorm(4000, mean = 1, sd = 1), nchains = 4)
x

# Or if the input sample has chain information as its second dimension, we can
# use with_chains to create the rvar
x <- rvar(array(rnorm(4000, mean = 1, sd = 1), dim = c(1000, 4)), with_chains = TRUE)
x

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

```r
## 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))
```
Matrix multiplication of random variables

Description
Matrix multiplication of random variables.

Usage
```r
def(x %**% y)
## S3 method for class 'rvar'
matrixOps(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)`.

In R >= 4.3, you can also use `%*%` in place of `%**%` for matrix multiplication of `rvars`. In R < 4.3, S3 classes cannot properly override `%*%`, so you must use `%**%` for matrix multiplication of `rvars`.

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-slice

Random variable slicing

Description

Operations for slicing rvars and replacing parts of rvars.

Usage

## S3 method for class 'rvar'
x[[i, ...]]

## S3 replacement method for class 'rvar'
x[[i, ...]] <- value

## S3 method for class 'rvar'
x[...drop = FALSE]

## S3 replacement method for class 'rvar'
x[i, ...] <- value

Arguments

x an rvar.
i, ... indices; see Details.
value (rvar or coercable to rvar) Value to insert into x at the location determined by the indices.
drop (logical) Should singular dimensions be dropped when slicing array rvars? Unlike base array slicing operations, defaults to FALSE.
Details

The `rvar` slicing operators ([ and ][]) attempt to implement the same semantics as the base array slicing operators. There are some exceptions; most notably, `rvar` slicing defaults to `drop = FALSE` instead of `drop = TRUE`.

Extracting or replacing single elements with [ [ ]]

The [ [ ] operator extracts (or replaces) single elements. It always returns (or replaces) a scalar (length-1) `rvar`.

The `x[i,...]` operator can be used as follows:

- `x[<numeric>]` for scalar numeric `i`: gives the `i`th element of `x`. If `x` is multidimensional (i.e. `length(dim(x)) > 1`), extra dimensions are ignored when indexing. For example, if `x` is a 6 × 2 `rvar` array, the 7th element, `x[[7]]`, will be the first element of the second column, `x[1,2].`
- `x[<numeric rvar>]` for scalar numeric `rvar `i`: a generalization of indexing when `i` is a scalar numeric. Within each draw of `x`, selects the element corresponding to the value of `i` within that same draw.
- `x[<character>]` for scalar character `i`: gives the element of `x` with name equal to `i`. Unlike with base arrays, does not work with multidimensional `rvars`.
- `x[[i_1,i_2,...,i_n]]` for scalar numeric or character `i_1, i_2,` etc. Must provide exactly the same number of indices as dimensions in `x`. Selects the element at the corresponding position in the `rvar` by number and/or dimname (as a string).

Extracting or replacing multiple elements with [ ]

The [ ] operator extracts (or replaces) multiple elements. It always returns (or replaces) a possibly-multidimensional `rvar`.

The `x[i,...]` operator can be used as follows:

- `x[<logical>]` for vector logical `i`: `i` is recycled to the same length as `x`, ignoring multiple dimensions in `x`, then an `rvar` vector is returned containing the elements in `x` where `i` is `TRUE`.
- `x[<logical rvar>]` for scalar logical `rvar `i`: returns an `rvar` the same shape as `x` containing only those draws where `i` is `TRUE`.
- `x[<numeric>]` for vector numeric `i`: an `rvar` vector is returned containing the `i`th elements of `x`, ignoring dimensions.
- `x[<matrix>]` for numeric matrix `i`, where `ncol(i) == length(dim(x))`: each row of `i` should give the multidimensional index for a single element in `x`. The result is an `rvar` vector of length `nrow(i)` containing elements of `x` selected by each row of `i`.
- `x[[i_1,i_2,...,i_n]]` for vector numeric, character, or logical `i_1, i_2,` etc. Returns a slice of `x` containing all elements from the dimensions specified in `i_1, i_2,` etc. If an argument is left empty, all elements from that dimension are included. Unlike base arrays, trailing dimensions can be omitted entirely and will still be selected; for example, if `x` has three dimensions, both `x[[1,]]` and `x[1,[]` can be used to create a slice that includes all elements from the last two dimensions. Unlike base arrays, [ ] defaults to `drop = FALSE`, so results retain the same number of dimensions as `x`. 
Examples

```r
x <- rvar(array(1:24, dim = c(4,2,3)))
dimnames(x) <- list(c("a","b"), c("d","e","f"))
x
```

```
## Slicing single elements
# x[[<numeric>]]
x[[2]]

# x[[<numeric rvar>]]
# notice the draws of x[1:4]...
draws_of(x[1:4])
x[[rvar(c(1,3,4,4))]]
# ... x[[rvar(c(1,3,4,4))]] creates a mixtures of those draws
draws_of(x[[rvar(c(1,3,4,4))]])

# x[[1,2,...]]
x[[2,"e"]]
```

```
## Slicing multiple elements
# x[[logical>]]
x[c(TRUE,TRUE,FALSE)]

# x[[logical rvar>]]
# select every other draw
x[rvar(c(TRUE,FALSE,TRUE,FALSE))]

# x[[numeric>]]
x[1:3]

# x[[matrix>]]
x[rbind(
  c(1,2),
  c(1,3),
  c(2,2)
)]

# x[i_1,i_2,...,i_n]
x[1,]
x[1:2:3]
x[,2:3]
```

---

**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'
Summary(...)

## 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)

### 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:
pnorm(1.5, mean = 1:4, sd = 2)
```
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.
- 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

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)
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). Op-
    tionally, 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 \texttt{x} is already an \texttt{rvar}, in which case the number of chains it has is used.

with_chains (logical) Does \texttt{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 \texttt{nchains} argument can be used to determine the number of chains. If TRUE, the \texttt{nchains} argument is ignored and the second dimension of \texttt{x} is used to index chains. Internally, the array will be converted to a format without the chain index. Ignored when \texttt{x} is already an \texttt{rvar}.

... Arguments passed on to \texttt{base::factor}

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

labels either an optional character vector of labels for the levels (in the same order as \texttt{levels} after removing those in \texttt{exclude}), or a character string of length 1. Duplicated values in \texttt{labels} can be used to map different values of \texttt{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 \texttt{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 \texttt{rvar()} that represents a (possibly multidimensional) sample of a \texttt{factor} or an \texttt{ordered} factor. It is otherwise very similar to the basic \texttt{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 \texttt{rvar_factor()} or c("ordered","factor") (for \texttt{rvar_ordered()}). If you pass a \texttt{factor} or \texttt{ordered} factor to \texttt{rvar()} it will automatically return an object with the classes "rvar_factor" or c("rvar_ordered","rvar_factor").

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

Examples
\begin{verbatim}
set.seed(1234)

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

# 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_array)

# 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_ifelse**

**Random variable ifelse**

**Description**

A version of ifelse() that returns an *rvar*.

**Usage**

rvar_ifelse(test, yes, no)

**Arguments**

- **test** (logical *rvar*, or castable to one) logical test determining whether the value in yes or no is assigned in the corresponding position of the result.
- **yes** (*rvar*, or castable to one) corresponding values assigned for entries in test that are TRUE.
- **no** (*rvar*, or castable to one) corresponding values assigned for entries in test that are FALSE.
Value
An `rvar` with the common type of `yes` and `no` (as determined by `vctrs::vec_cast_common()`) and a shape determined by broadcasting `test`, `yes`, and `no` to a common shape (see the section on broadcasting rules in vignette("rvar")). For every element of `draws_of(test)`, the corresponding element of `draws_of(yes)` or `draws_of(no)` is placed into the result, depending on whether the element of `test` is `TRUE` or `FALSE`.

Examples
```
x <- rvar(1:4)
y <- rvar(5:8)
i <- rvar(c(TRUE, FALSE, TRUE, FALSE))
z <- rvar_ifelse(i, x, y)
z
draws_of(z)
```

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.
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.

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)
Details

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

\texttt{rvar\_rng()} is a fast alternative to \texttt{rdo()} or \texttt{rfun()}, but you must ensure that \texttt{.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 \texttt{rdo()} or \texttt{rfun()} (which should be correct, but slower) if you are uncertain.

Value

A single-dimensional \texttt{rvar} of length \( n \).

See Also

Other \texttt{rfun}: \texttt{rdo()}, \texttt{rfun()}

Examples

```r
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
```

---

\textbf{split\_chains}

\textit{Split Chains}

Description

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

Usage

\texttt{split\_chains(x, ...)}

Arguments

\begin{itemize}
  \item \texttt{x} (draws) A \texttt{draws} object or another \texttt{R} object for which the method is defined.
  \item \texttt{...} Arguments passed to individual methods (if applicable).
\end{itemize}

Value

A \texttt{draws} object of the same class as \texttt{x}.
**Examples**

```r
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**

```r
subset_draws(x, ...)
```

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

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

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

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

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

## S3 method for class 'draws'
subset_draws

subset(x, ...)

Arguments

x      (draws) A draws object or another R object for which the method is defined.
...    Arguments passed to individual methods (if applicable).
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.
exclude  (logical) Should the selected subset be excluded? If FALSE (the default) only the selected subset will be returned. If TRUE everything but the selected subset will be returned.
scalar   (logical) Should only scalar variables be selected? If FALSE (the default), all variables with matching names and arbitrary indices will be selected (see examples).

Details

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

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")

# trying to extract only a scalar 'theta' will fail
Thin draws objects to reduce their size and autocorrelation in the chains.

Usage

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

## S3 method for class 'draws'
thin_draws(x, thin = NULL, ...)

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

Arguments

- **x** (draws) A draws object or another R object for which the method is defined.
- **thin** (positive numeric) The period for selecting draws. Must be between 1 and the number of iterations. If the value is not an integer, the draws will be selected such that the number of draws returned is equal to `round(ndraws(x) / thin)`. Intervals between selected draws will be either `ceiling(thin)` or `floor(thin)`, such that the average interval will be close to the thin value. If `NULL`, it will be automatically calculated based on bulk and tail effective sample size as suggested by Säilynoja et al. (2022).
- **...** Arguments passed to individual methods (if applicable).

Value

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

References

Variables

Examples

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

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

Variables

Get variable names from draws objects

Description

Get variable names from draws objects.

Usage

```r
variables(x, ...)
```

## S3 method for class 'draws_matrix'
variables(x, reserved = FALSE, with_indices = TRUE, ...)

## S3 method for class 'draws_array'
variables(x, reserved = FALSE, with_indices = TRUE, ...)

## S3 method for class 'draws_df'
variables(x, reserved = FALSE, with_indices = TRUE, ...)

## S3 method for class 'draws_list'
variables(x, reserved = FALSE, with_indices = TRUE, ...)

## S3 method for class 'draws_rvars'
variables(x, reserved = FALSE, with_indices = FALSE, ...)

Arguments

- `x` (draws) A draws object or another R object for which the method is defined.
- `...` Arguments passed to individual methods (if applicable).
- `reserved` (logical) Should reserved variables be included in the output? Defaults to FALSE. See `reserved_variables` for an overview of currently reserved variable names.
- `with_indices` (logical) Should indices be included in variable names? For example, if the object includes variables named "x[1]" and "x[2]", if TRUE, c("x[1]", "x[2]") is returned; if FALSE, only "x" is returned. Defaults to TRUE for all formats except `draws_rvars()`.
variables() returns a vector of all variable names, and nvariables() returns the number of variables.

Value
For variables(), a character vector.
For nvariables(), a scalar integer.

See Also
variables<-, rename_variables, draws-index

Examples
x <- example_draws()

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

**Arguments**

- `x` (draws) A draws object or another R object for which the method is defined.
- `...` Arguments passed to individual methods (if applicable).
- `value`, `variables` (character vector) new variable names.
- `with_indices` (logical) Should indices be included in variable names? For example, if the object includes variables named "x[1]" and "x[2]"; if TRUE, `c("x[1]", "x[2]")` is returned; if FALSE, only "x" is returned. Defaults to TRUE for all formats except `draws_rvars()`.

**Details**

`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, `set_variables(x, value)` works equivalently, but is more intuitive when using the pipe operator.

For renaming specific variables, `rename_variables()` may offer a more convenient approach.

**Value**

Returns a `draws` object of the same format as `x`, with variables named as specified.

**See Also**

`variables`, `rename_variables`, `draws-index`

**Examples**

```r
x <- example_draws()
variables(x)
nvariables(x)
variables(x) <- letters[1:nvariables(x)]
# or equivalently...
x <- set_variables(x, letters[1:nvariables(x)])
```
weights.draws

Usage

```r
## 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

```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

# add weights on log scale and Pareto smooth them
x <- weight_draws(x, weights = log_wts, log = TRUE, pareto_smooth = TRUE)
```
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

```r
weight_draws(x, weights, ...)
```

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

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

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

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

## S3 method for class 'draws_rvars'
```r
weight_draws(x, weights, log = FALSE, pareto_smooth = 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.
- **pareto_smooth** (logical) Should the weights be Pareto-smoothed? The default is FALSE.

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)

# add weights on log scale and Pareto smooth them
x <- weight_draws(x, weights = log_wts, log = TRUE, pareto_smooth = TRUE)
```
Index

* diagnostics
  ess_basic, 29
  ess_bulk, 31
  ess_quantile, 33
  ess_sd, 34
  ess_tail, 36
  mcse_mean, 45
  mcse_quantile, 46
  mcse_sd, 47
  pareto_diags, 53
  pareto_khat, 56
  rhat, 79
  rhat_basic, 80
  rhat_nested, 81
  rstar, 83
* formats
  draws, 13
  draws_array, 15
  draws_df, 17
  draws_list, 19
  draws_matrix, 20
  draws_rvars, 23
* helper-functions
  ps_convergence_rate, 68
  ps_khat_threshold, 68
  ps_min_ss, 69
  ps_tail_length, 70
* rfun
  rdo, 71
  rfun, 77
  rvar_rng, 103
* rvar-summaries
  rvar-summaries-over-draws, 92
  rvar-summaries-within-draws, 96
  rvar_is_finite, 102
* variable extraction methods
  extract_variable, 38
  extract_variable_array, 39
  extract_variable_matrix, 40
  [.rvar (rvar-slice), 90
  [<=.rvar (rvar-slice), 90
  [[.rvar (rvar-slice), 90
  [<=-.rvar (rvar-slice), 90
  %**% (rvar-matmult), 89
  %in% (match), 44
  all.rvar (rvar-summaries-over-draws), 92
  any.rvar (rvar-summaries-over-draws), 92
  as.character, 7, 100
  as_draws (draws), 13
  as_draws_array (draws_array), 15
  as_draws_df (draws_df), 17
  as_draws_list (draws_list), 19
  as_draws_matrix (draws_matrix), 20
  as_draws_rvars (draws_rvars), 23
  as_function(), 25
  as_rvar, 5
  as_rvar(), 43, 86, 98
  as_rvar_factor, 7
  as_rvar_factor(), 43, 100
  as_rvar_integer (as_rvar), 5
  as_rvar_logical (as_rvar), 5
  as_rvar_numeric (as_rvar), 5
  as_rvar_ordered (as_rvar_factor), 7
  as_rvar_ordered(), 43
  base array slicing operators, 91
  base::diag(), 10
  base::drop(), 27
  base::factor, 7, 100
  base::match, 44
  base::match(), 44
  bind_draws, 8
  cdf.rvar (rvar-dist), 87
  cdf.rvar_factor (rvar-dist), 87
  cdf.rvar_ordered (rvar-dist), 87
  chain_ids (draws-index), 14
  chol.rvar, 9

115
INDEX

draw_ids, 11

default_convergence_measures (draws_summary), 25
default_mcse_measures (draws_summary), 25
default_summary_measures (draws_summary), 25
density.rvar (rvar-dist), 87
density.rvar_factor (rvar-dist), 87
diag, rvar-method, 10
diagnostic, 25
diagnostics, 11, 26
dim(), 6, 7, 71, 86, 99
dimnames(), 6, 7, 86, 99
dissent, 12
dissent(), 64, 67
draw.id() (draws-index), 14
draws, 4, 8, 9, 13, 14, 16, 18, 20, 21, 24, 49,
   51, 52, 72–74, 77, 105, 108–113
draws-index, 14
draws_array, 14, 15, 18, 20, 21, 24, 37, 52, 60
draws_df, 14, 16, 17, 20, 21, 24, 52, 61, 77, 83
draws_list, 14, 16, 18, 19, 21, 24, 52, 62
draws_matrix, 14, 16, 18, 20, 20, 24, 52, 63
draws_of, 22
draws_of(), 72, 78, 86
draws_of<- (draws_of), 22
draws_rvars, 14, 16, 18, 20, 21, 23, 42, 52, 64
draws_rvars(), 109, 111
draws_summary, 25
drop, rvar-method, 27

E (rvar-summaries-over-draws), 92
entropy, 28
entropy(), 64, 66
ess_basic, 29, 32, 34, 35, 37, 46–48, 55, 57,
   80, 81, 83, 84
ess_basic(), 11
ess_bulk, 30, 31, 34, 35, 37, 46–48, 55, 57,
   80, 81, 83, 84
ess_bulk(), 11, 26, 29, 36
ess_mean, 32
ess_mean.default (ess_quantile), 33
ess_median (ess_quantile), 33
ess_quantile, 30, 32, 33, 35, 37, 46–48, 55,
   57, 80, 81, 83, 84
ess_quantile(), 11

ess.sd, 30, 32, 34, 37, 46–48, 55, 57, 80,
   81, 83, 84
ess.sd(), 11
ess.tail, 30, 32, 34, 35, 36, 46–48, 55, 57,
   80, 81, 83, 84
ess.tail(), 11, 26, 29, 31
example.draws, 37
extract_variable, 38, 40, 41
extract_variable_array, 39, 40, 41
extract_variable_array(), 39, 41
extract_variable_matrix, 39, 40, 40
extract_variable_matrix(), 30–33, 35, 36,
   43, 46, 48, 54, 56, 58, 71, 79, 80, 82

factor, 12, 28, 50, 99
for_each_draw, 41
format(), 67
format.rvar (print.rvar), 66

integer, 6, 12, 28, 50
is.finite.rvar (rvar-summaries-over-draws), 92
is.infinite.rvar (rvar-summaries-over-draws), 92
is.na.rvar (rvar-summaries-over-draws), 92
is.nan.rvar (rvar-summaries-over-draws), 92
is.draws (draws), 13
is.draws_array (draws_array), 15
is.draws_df (draws_df), 17
is.draws_list (draws_list), 19
is.draws_matrix (draws_matrix), 20
is.draws_rvars (draws_rvars), 23
is.rvar, 43
is.rvar(), 72
is.rvar_factor, 43
is.rvar_ordered (is_rvar_factor), 43
iteration_ids (draws-index), 14

logical, 6, 89

mad (rvar-summaries-over-draws), 92
mad(), 26

match, 44
matrix multiplication, 22
matrixOps.rvar (rvar-matmult), 89
max.rvar (rvar-summaries-over-draws), 92
mcse_mean, 30, 32, 34, 37, 45, 47, 48, 55,
   57, 80, 81, 83, 84
mcse_mean(), 11
mcse_median (mcse_quantile), 46
mcse_quantile, 30, 32, 34, 35, 37, 46, 46, 48, 55, 57, 80, 81, 83, 84
mcse_quantile(), 11
mcse_sd, 30, 32, 34, 35, 37, 46, 47, 47, 55, 57, 80, 81, 83, 84
mcse_sd(), 11
mean(), 26
mean.rvar (rvar-summaries-over-draws), 92
median(), 26
median.rvar (rvar-summaries-over-draws), 92
merge_chains, 49, 107
min.rvar (rvar-summaries-over-draws), 92
modal_category, 50
mutate_variables, 51, 73
names, 10, 71
nchains (draws-index), 14
ndraws (draws-index), 14
niterations (draws-index), 14
num(), 4, 26, 65
numeric, 6, 10, 12, 27, 28, 50, 89
nvariables (variables), 109
option, 26, 60–65
order_draws, 52
order_draws(), 74
ordered, 100
pareto_convergence_rate, 55, 68
pareto_convergence_rate (pareto_diags), 53
pareto_diags, 30, 32, 34, 35, 37, 46–48, 53, 57, 59, 68, 69, 80, 81, 83, 84
pareto_diags(), 11
pareto_khat, 30, 32, 34, 35, 37, 46–48, 55, 56, 59, 80, 81, 83, 84
pareto_khat(), 11
pareto_khat_threshold, 55, 68
pareto_khat_threshold (pareto_diags), 53
pareto_min_ss, 55, 69
pareto_min_ss (pareto_diags), 53
pareto_smooth, 55, 57, 58
pillar::style_num(), 67
posterior (posterior-package), 4
posterior-package, 4
Pr (rvar-summaries-over-draws), 92
print(), 60–64, 66
print.draws_array, 60
print.draws_df, 61
print.draws_list, 62
print.draws_matrix, 63
print.draws_rvars, 64
print.draws_summary, 65
print.rvar, 66
prod.rvar (rvar-summaries-over-draws), 92
ps_convergence_rate, 68, 69, 70
ps_khat_threshold, 68, 69, 70
ps_min_ss, 68, 69, 70
ps_tail_length, 68, 69, 70
quantile.rvar (rvar-dist), 87
quantile.rvar_factor (rvar-dist), 87
quantile.rvar_ordered (rvar-dist), 87
quantile2, 70
quantile2(), 26
quasi quotation, 41, 71
range.rvar (rvar-summaries-over-draws), 92
rdo, 71, 78, 104
rdo(), 6, 8, 86, 98, 100
rename_variables, 15, 52, 72, 110, 111
rename_variables(), 111
repair_draws, 73
repair_draws(), 53
repairs, 107
resample_draws, 74, 112
resample_draws(), 76, 114
reserved_variables, 60–64, 76, 109
rfun, 72, 77, 104
rfun(), 6, 8, 86, 98, 100
rhat, 30, 32, 34, 35, 37, 46–48, 55, 57, 79, 81, 83, 84
rhat(), 11, 26, 40, 80
rhat_basic, 30, 32, 34, 35, 37, 46–48, 55, 57, 80, 80, 83, 84
rhat_basic(), 11
rhat_nested, 30, 32, 34, 35, 37, 46–48, 55, 57, 80, 81, 81, 84
rhat_nested(), 11
rlang::as_function(), 78
rstar, 30, 32, 34, 35, 37, 46–48, 55, 57, 80, 81, 83, 83
rstar(), 11
rvar(), 6, 8, 100
rvar-dist, 87, 95, 97
rvar-matmult, 89
rvar-slice, 90
rvar-summaries-over-draws, 92, 97, 103
rvar-summaries-within-draws, 95, 96
rvar_all (rvar-summaries-within-draws), 96
rvar_any (rvar-summaries-within-draws), 96
rvar_apply, 97
rvar_factor, 6–8, 12, 28, 43, 50, 64, 66, 67, 99
rvar_factor(), 8, 100
rvar_ifelse, 101
rvar_is_finite, 95, 97, 102
rvar_is_infinite (rvar_is_finite), 102
rvar_is_na (rvar_is_finite), 102
rvar_is_nan (rvar_is_finite), 102
rvar_mad (rvar-summaries-within-draws), 96
rvar_max (rvar-summaries-within-draws), 96
rvar_mean
(rvar-summaries-within-draws), 96
rvar_median
(rvar-summaries-within-draws), 96
rvar_min (rvar-summaries-within-draws), 96
rvar_ordered, 6–8, 12, 28, 43, 50, 64, 66
rvar_ordered (rvar_factor), 99
rvar_ordered(), 8, 100
rvar_prod
(rvar-summaries-within-draws), 96
rvar_quantile
(rvar-summaries-within-draws), 96
rvar_range
(rvar-summaries-within-draws), 96
rvar_rng, 72, 78, 103
rvar_rng(), 6, 8, 72, 78, 86, 98, 100
rvar_sd (rvar-summaries-within-draws), 96
rvar_sum (rvar-summaries-within-draws), 96
rvar_var (rvar-summaries-within-draws), 96
sd (rvar-summaries-over-draws), 92
sd(), 26
set_variables (variables<-), 110
split_chains, 104
stats::density(), 88
stats::quantile(), 71, 88
str.rvar (print.rvar), 66
subset.draews (subset_draews), 105
subset_draews, 75, 105
sum.rvar (rvar-summaries-over-draws), 92
summarise_draews (draews_summary), 25
summarise_draews(), 65
summarize_draews, 4
summarize_draews (draews_summary), 25
summary(), 25
summary.draews (draews_summary), 25
Summary.rvar
(rvar-summaries-over-draws), 92
summary.rvar (draews_summary), 25
thin (thin_draews), 108
thin_draews, 108
tibble, 18, 26
tibble::print.tbl_df(), 65
var (rvar-summaries-over-draws), 92
variables, 15, 52, 73, 109, 111
variables<-, 110
variance.rvar
(rvar-summaries-over-draws), 92
weight_draews, 77, 111, 112, 113
weight_draews(), 75
weights.draews, 111
weights.draews() 113, 114