

Package ‘rvec’

December 10, 2025

Type Package

Title Vectors Representing Random Variables

Version 1.0.0

Description Random vectors, called rvecs. An rvec holds multiple draws, but tries to behave like a standard R vector, including working well in data frames. Rvecs are useful for analysing output from a simulation or a Bayesian analysis.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

RoxygenNote 7.3.3

Depends R (>= 4.3.0)

Imports cli, glue, graphics, grDevices, lifecycle, Matrix, matrixStats, methods, rlang, stats, tibble, tidyselect, utils, vctrs

Suggests bookdown, covr, dplyr, ggdist, ggplot2, knitr, posterior, rmarkdown, testthat (>= 3.0.0), tidy, withr

Config/testthat/edition 3

VignetteBuilder knitr

URL <https://bayesiandemography.github.io/rvec/>,
<https://github.com/bayesiandemography/rvec>

BugReports <https://github.com/bayesiandemography/rvec/issues>

NeedsCompilation no

Author John Bryant [aut, cre],
Bayesian Demography Limited [cph]

Maintainer John Bryant <john@bayesiandemography.com>

Repository CRAN

Date/Publication 2025-12-10 14:10:02 UTC

Contents

rvec-package	3
as_list_col	5
collapse_to_rvec	6
dbeta_rvec	9
dbinom_rvec	10
dcauchy_rvec	12
dchisq_rvec	13
dexp_rvec	15
df_rvec	16
dgamma_rvec	17
dgeom_rvec	19
dhyper_rvec	20
divorce	22
dlnorm_rvec	23
dmultinom_rvec	24
dnbinom_rvec	25
dnorm_rvec	27
dpois_rvec	28
draws_all	30
draws_ci	31
draws_fun	33
draws_median	34
draws_min	36
draws_quantile	37
draws_sd	39
dt_rvec	41
dunif_rvec	42
dweibull_rvec	43
extract_draw	45
if_else_rvec	46
is_rvec	47
map_rvec	47
matrixOps.rvec	48
missing	49
new_rvec_blank	51
new_rvec_deprecated	52
n_draw	53
pool_draws	53
prob	55
rank	56
reg_post	57
rvec	57
sd	59
var	60
weighted_mean	61

rvec-package

Package 'rvec'

Description

Tools for working with random draws from a distribution, eg draws from a posterior distribution in a Bayesian analysis.

Details

An rvec holds multiple draws, but wherever possible behaves like an ordinary R vector. For instance, if `x` is an rvec holding 1000 draws from a distribution, then `2 * x` returns a new rvec where each draw has been multiplied by 2.

To summarise across draws, use a function starting with `draws`. For instance, to calculate a credible interval, use `draws_ci()`.

Functions

Creating rvecs

- `rvec()` Class depends on input
- `rvec_dbl()` Doubles
- `rvec_int()` Integers
- `rvec_lgl()` Logical
- `rvec_chr()` Character
- `new_rvec_dbl()` Empty doubles
- `new_rvec_int()` Empty integers
- `new_rvec_lgl()` Empty logical
- `new_rvec_chr()` Empty character
- `collapse_to_rvec()` Data in data frame

Manipulating rvecs

- `if_else_rvec()` `if_else()` where condition is rvec
- `map_rvec()` `map()` for rvecs
- `extract_draw()` Single draw from rvec
- `pool_draws()` Combine samples

Probability distributions

- `dbeta_rvec()` Beta
- `dbinom_rvec()` Binomial
- `dcauchy_rvec()` Cauchy
- `dchisq_rvec()` Chi-square

- `dexp_rvec()` Exponential
- `df_rvec()` F
- `dgamma_rvec()` Gamma
- `dgeom_rvec()` Geometric
- `dhyper_rvec()` Hypergeometric
- `dlnorm_rvec()` Lognormal
- `dmultinom()` Multinomial
- `dnbinom_rvec()` Negative binomial
- `dnorm_rvec()` Normal
- `dpois_rvec()` Poisson
- `dt_rvec()` Student's T
- `dunif_rvec()` Uniform
- `dweibull_rvec()` Weibull

Summarizing across draws

- `draws_all()` All
- `draws_any()` Any
- `draws_min()` Minimum
- `draws_max()` Maximum
- `draws_median()` Median
- `draws_mean()` Mean
- `draws_mode()` Modal
- `draws_sd()` Standard deviation
- `draws_var()` Variances
- `draws_cv()` Coefficients of variation
- `draws_ci()` Credible intervals
- `draws_quantile()` Quantiles
- `draws_fun()` Arbitrary function
- `n_draw()` Number

Coercion, classes

- `as_list_col()` Rvec or matrix to list
- `expand_from_rvec()` Inverse of `collapse_to_rvec()`
- `is_rvec()` Object an rvec?

Weighted summaries

- `weighted_mad()` Weighted mean absolute deviation
- `weighted_mean()` Weighted mean
- `weighted_median()` Weighted median

- [weighted_sd\(\)](#) Weighted standard deviation
- [weighted_var\(\)](#) Weighted variances

Datasets

- [divorce\(\)](#) Divorce rates
- [reg_post\(\)](#) Regression coefficients

Packages with similar functionality

- [rv](#)
- [posterior](#)

Author(s)

Maintainer: John Bryant <john@bayesiandemography.com>

Other contributors:

- Bayesian Demography Limited [copyright holder]

See Also

Useful links:

- <https://bayesiandemography.github.io/rvec/>
- <https://github.com/bayesiandemography/rvec>
- Report bugs at <https://github.com/bayesiandemography/rvec/issues>

as_list_col

Convert to List Column

Description

Convert an [rvec](#) or matrix to a list that can be used as a list column in a data frame.

Usage

```
as_list_col(x)

## S3 method for class 'rvec'
as_list_col(x)

## S3 method for class 'matrix'
as_list_col(x)
```

Arguments

x An [rvecs](#) or matrix.

Value

A list:

- If `x` is an `rvec`, then the list contains `length(x)` vectors, each of which has `n_draw(x)` elements.
- If `x` is a matrix, then the list contains `nrow(x)` vectors, each of which has `ncol(x)` elements.

See Also

- `rvec()` Construct an `rvec`.
- `expand_from_rvec()` Convert a data frame from 'rvec' format to 'draw-and-value' format
- Functions for summarising and plotting distributions in package `ggdist` use list columns (among other formats).

Examples

```
l <- list(1:3,
         4:6)
r <- rvec(l)
as_list_col(r)
```

collapse_to_rvec	<i>Convert a Data Frame Between 'Draws-and-Values' Format and 'Rvec' Format</i>
------------------	---

Description

`collapse_to_rvec()` converts a data frame from a 'draws-and-values' format to an 'rvec' format. `expand_from_rvec()`, does the opposite, converting a data frame from an `rvecs` format to a draws-and-values format.

Usage

```
collapse_to_rvec(data, draw = draw, values = value, by = NULL, type = NULL)

## S3 method for class 'data.frame'
collapse_to_rvec(data, draw = draw, values = value, by = NULL, type = NULL)

## S3 method for class 'grouped_df'
collapse_to_rvec(data, draw = draw, values = value, by = NULL, type = NULL)

expand_from_rvec(data, draw = "draw")

## S3 method for class 'data.frame'
expand_from_rvec(data, draw = "draw")

## S3 method for class 'grouped_df'
expand_from_rvec(data, draw = "draw")
```

Arguments

data	A data frame, possibly grouped .
draw	<tidyselect> The variable that uniquely identifies random draws within each combination of values for the 'by' variables. Must be quoted for <code>expand_from_rvec()</code> .
values	<tidyselect> One or more variables in data that hold measurements.
by	<tidyselect> Variables used to stratify or cross-classify the data. See Details.
type	String specifying the class of rvec to use for each variable. Optional. See Details.

Details

In a draws-and-values format, each row represents one random draw. The data frame contains a 'draw' variable that distinguishes different draws within the same combination of 'by' variables. In rvec format, each row represents one combination of 'by' variables, and multiple draws are stored in an **rvec**. See below for examples.

Value

A data frame.

- `collapse_to_rvec()` **reduces** the number of rows by a factor of `n_draw()`.
- `expand_from_rvec()` **increases** the number of rows by a factor of `n_draw()`.
- `collapse_to_rvec()` silently drops all variables that are not draw, value, or grouping variables if data is a **grouped** data frame.

by argument

The by argument is used to specify stratifying variables. For instance if by includes sex and age, then data frame produced by `collapse_to_rvec()` has separate rows for each combination of sex and age.

If data is a **grouped** data frame, then the grouping variables take precedence over by.

If no value for by is provided, and data is not a grouped data frame, then `collapse_to_rvec()` assumes that all variables in data that are not included in value and draw should be included in by.

type argument

By default, `collapse_to_rvec()` calls function `rvec()` on each values variable in data. `rvec()` chooses the class of the output (ie `rvec_chr`, `rvec_dbl`, `rvec_int`, or `rvec_lgl`) depending on the input. Types can instead be specified in advance, using the type argument. type is a string, each character of which specifies the class of the corresponding values variable. The characters have the following meanings:

- "c": `rvec_chr`
- "d": `rvec_dbl`
- "i": `rvec_int`
- "l": `rvec_lgl`

- `"?"`: Depends on inputs.

The codes for type are modified from ones used by the `readr` package.

See Also

- `rvec()` Construct a single rvec
- `as_list_col()` Convert an rvec to a list variable
- `dplyr::group_vars()` Names of grouping variables

`collapse_to_rvec()` and `expand_from_rvec()` are analogous to `tidyr::nest()` and `tidyr::unnest()` though `collapse_to_rvec()` and `expand_from_rvec()` move values into and out of rvecs, while `tidyr::nest()` and `tidyr::unnest()` move them in and out of data frames. (`tidyr::nest()` and `tidyr::unnest()` are also a lot more flexible.)

Examples

```
library(dplyr)
data_db <- tribble(
  ~occupation, ~sim, ~pay,
  "Statistician", 1, 100,
  "Statistician", 2, 80,
  "Statistician", 3, 105,
  "Banker", 1, 400,
  "Banker", 2, 350,
  "Banker", 3, 420
)

## draws-and-values format to rvec format
data_rv <- data_db |>
  collapse_to_rvec(draw = sim,
                   values = pay)
data_rv

## rvec format to draws-and-values format
data_rv |>
  expand_from_rvec()

## provide a name for the draw variable
data_rv |>
  expand_from_rvec(draw = "sim")

## specify that rvec variable
## must be rvec_int
data_rv <- data_db |>
  collapse_to_rvec(draw = sim,
                   values = pay,
                   type = "i")

## specify stratifying variable explicitly,
## using 'by' argument
data_db |>
```

```

collapse_to_rvec(draw = sim,
                 values = pay,
                 by = occupation)

## specify stratifying variable explicitly,
## using 'group_by'
library(dplyr)
data_db |>
  group_by(occupation) |>
  collapse_to_rvec(draw = sim,
                  values = pay)

```

dbeta_rvec

*Beta Distribution, Using Multiple Draws***Description**

Density, distribution function, quantile function and random generation for the Beta distribution, modified to work with rvecs.

Usage

```

dbeta_rvec(x, shape1, shape2, ncp = 0, log = FALSE)

pbeta_rvec(q, shape1, shape2, ncp = 0, lower.tail = TRUE, log.p = FALSE)

qbeta_rvec(p, shape1, shape2, ncp = 0, lower.tail = TRUE, log.p = FALSE)

rbeta_rvec(n, shape1, shape2, ncp = 0, n_draw = NULL)

```

Arguments

x	Quantiles. Can be an rvec.
shape1, shape2	Parameters for beta distribution. Non-negative. See dbeta() . Can be an rvec.
ncp	Non-centrality parameter. Default is 0. Cannot be an rvec.
log, log.p	Whether to return results on a log scale. Default is FALSE. Cannot be an rvec.
q	Quantiles. Can be an rvec.
lower.tail	Whether to return $P[X \leq x]$, as opposed to $P[X > x]$. Default is TRUE. Cannot be an rvec.
p	Probabilities. Can be an rvec.
n	The length of random vector being created. Cannot be an rvec.
n_draw	Number of random draws in the random vector being created. Cannot be an rvec.

Details

Functions `dbeta_rvec()`, `pbeta_rvec()`, `qbeta_rvec()` and `rbeta_rvec()` work like base R functions `dbeta()`, `pbeta()`, `qbeta()`, and `rbeta()`, except that they accept rvecs as inputs. If any input is an rvec, then the output will be too. Function `rbeta_rvec()` also returns an rvec if a value for `n_draw` is supplied.

`dbeta_rvec()`, `pbeta_rvec()`, `qbeta_rvec()` and `rbeta_rvec()` use [tidyverse](#) vector recycling rules:

- Vectors of length 1 are recycled
- All other vectors must have the same size

Value

- If any of the arguments are rvecs, or if a value for `n_draw` is supplied, then an [rvec](#); otherwise an ordinary R vector.

See Also

- [dbeta\(\)](#), [pbeta\(\)](#), [qbeta\(\)](#), [rbeta\(\)](#) Base R equivalents
- [distributions](#) All base R distributions

Examples

```
x <- rvec(list(c(0, 0.25),
               c(0.5, 0.99)))
dbeta_rvec(x, shape1 = 1, shape2 = 1)
pbeta_rvec(x, shape1 = 1, shape2 = 1)

rbeta_rvec(n = 2,
           shape = 1:2,
           shape2 = 1,
           n_draw = 1000)
```

dbinom_rvec

Binomial Distribution, Using Multiple Draws

Description

Density, distribution function, quantile function and random generation for the binomial distribution, modified to work with rvecs.

Usage

```
dbinom_rvec(x, size, prob, log = FALSE)

pbinom_rvec(q, size, prob, lower.tail = TRUE, log.p = FALSE)

qbinom_rvec(p, size, prob, lower.tail = TRUE, log.p = FALSE)

rbinom_rvec(n, size, prob, n_draw = NULL)
```

Arguments

<code>x</code>	Quantiles. Can be an rvec.
<code>size</code>	Number of trials. See dbinom() . Can be an rvec.
<code>prob</code>	Probability of success in each trial. See dbinom() . Can be an rvec.
<code>log, log.p</code>	Whether to return results on a log scale. Default is FALSE. Cannot be an rvec.
<code>q</code>	Quantiles. Can be an rvec.
<code>lower.tail</code>	Whether to return $P[X \leq x]$, as opposed to $P[X > x]$. Default is TRUE. Cannot be an rvec.
<code>p</code>	Probabilities. Can be an rvec.
<code>n</code>	The length of random vector being created. Cannot be an rvec.
<code>n_draw</code>	Number of random draws in the random vector being created. Cannot be an rvec.

Details

Functions `dbinom_rvec()`, `pbinom_rvec()`, `qbinom_rvec()` and `rbinom_rvec()` work like base R functions [dbinom\(\)](#), [pbinom\(\)](#), [qbinom\(\)](#), and [rbinom\(\)](#), except that they accept rvecs as inputs. If any input is an rvec, then the output will be too. Function `rbinom_rvec()` also returns an rvec if a value for `n_draw` is supplied.

`dbinom_rvec()`, `pbinom_rvec()`, `qbinom_rvec()` and `rbinom_rvec()` use [tidyverse](#) vector recycling rules:

- Vectors of length 1 are recycled
- All other vectors must have the same size

Value

- If any of the arguments are rvecs, or if a value for `n_draw` is supplied, then an [rvec](#); otherwise an ordinary R vector.
- Unlike base `rbinom()`, `rbinom_rvec()` always returns doubles.

See Also

- [dbinom\(\)](#), [pbinom\(\)](#), [qbinom\(\)](#), [rbinom\(\)](#) Base R equivalents
- [distributions](#) All base R distributions

Examples

```
x <- rvec(list(c(3, 8),
               c(0, 2)))
dbinom_rvec(x, size = 8, prob = 0.3)
pbinom_rvec(x, size = 8, prob = 0.3)

rbinom_rvec(n = 2,
            size = 10,
            prob = c(0.7, 0.3),
            n_draw = 1000)
```

dcauchy_rvec

*Cauchy Distribution, Using Multiple Draws***Description**

Density, distribution function, quantile function and random generation for the Cauchy distribution, modified to work with rvecs.

Usage

```
dcauchy_rvec(x, location = 0, scale = 1, log = FALSE)

pcauchy_rvec(q, location = 0, scale = 1, lower.tail = TRUE, log.p = FALSE)

qcauchy_rvec(p, location = 0, scale = 1, lower.tail = TRUE, log.p = FALSE)

rcauchy_rvec(n, location = 0, scale = 1, n_draw = NULL)
```

Arguments

x	Quantiles. Can be an rvec.
location	Center of distribution. Default is 0. See dcauchy() . Can be an rvec.
scale	Scale parameter. Default is 1. See dcauchy() . Can be an rvec.
log, log.p	Whether to return results on a log scale. Default is FALSE. Cannot be an rvec.
q	Quantiles. Can be an rvec.
lower.tail	Whether to return $P[X \leq x]$, as opposed to $P[X > x]$. Default is TRUE. Cannot be an rvec.
p	Probabilities. Can be an rvec.
n	The length of random vector being created. Cannot be an rvec.
n_draw	Number of random draws in the random vector being created. Cannot be an rvec.

Details

Functions `dcauchy_rvec()`, `pcauchy_rvec()`, `qcauchy_rvec()` and `rcauchy_rvec()` work like base R functions [dcauchy\(\)](#), [pcauchy\(\)](#), [qcauchy\(\)](#), and [rcauchy\(\)](#), except that they accept rvecs as inputs. If any input is an rvec, then the output will be too. Function `rcauchy_rvec()` also returns an rvec if a value for `n_draw` is supplied.

`dcauchy_rvec()`, `pcauchy_rvec()`, `qcauchy_rvec()` and `rcauchy_rvec()` use [tidyverse](#) vector recycling rules:

- Vectors of length 1 are recycled
- All other vectors must have the same size

Value

- If any of the arguments are rvecs, or if a value for `n_draw` is supplied, then an [rvec](#); otherwise an ordinary R vector.

See Also

- [dcauchy\(\)](#), [pcauchy\(\)](#), [qcauchy\(\)](#), [rcauchy\(\)](#) Base R equivalents
- [distributions](#) All base R distributions

Examples

```
x <- rvec(list(c(3, -5.1),
               c(0, -2.3)))
dcauchy_rvec(x)
pcauchy_rvec(x)

rcauchy_rvec(n = 2,
             location = c(-5, 5),
             n_draw = 1000)
```

dchisq_rvec

Chi-Squared Distribution, Using Multiple Draws

Description

Density, distribution function, quantile function and random generation for the chi-squared distribution, modified to work with rvecs.

Usage

```
dchisq_rvec(x, df, ncp = 0, log = FALSE)

pchisq_rvec(q, df, ncp = 0, lower.tail = TRUE, log.p = FALSE)

qchisq_rvec(p, df, ncp = 0, lower.tail = TRUE, log.p = FALSE)

rchisq_rvec(n, df, ncp = 0, n_draw = NULL)
```

Arguments

<code>x</code>	Quantiles. Can be an rvec.
<code>df</code>	Degrees of freedom. See dchisq() . Can be an rvec.
<code>ncp</code>	Non-centrality parameter. Default is 0. Cannot be an rvec.
<code>log, log.p</code>	Whether to return results on a log scale. Default is FALSE. Cannot be an rvec.
<code>q</code>	Quantiles. Can be an rvec.
<code>lower.tail</code>	Whether to return $P[X \leq x]$, as opposed to $P[X > x]$. Default is TRUE. Cannot be an rvec.
<code>p</code>	Probabilities. Can be an rvec.
<code>n</code>	The length of random vector being created. Cannot be an rvec.
<code>n_draw</code>	Number of random draws in the random vector being created. Cannot be an rvec.

Details

Functions `dchisq_rvec()`, `pchisq_rvec()`, `qchisq_rvec()` and `rchisq_rvec()` work like base R functions [dchisq\(\)](#), [pchisq\(\)](#), [qchisq\(\)](#), and [rchisq\(\)](#), except that they accept rvecs as inputs. If any input is an rvec, then the output will be too. Function `rchisq_rvec()` also returns an rvec if a value for `n_draw` is supplied.

`dchisq_rvec()`, `pchisq_rvec()`, `qchisq_rvec()` and `rchisq_rvec()` use [tidyverse](#) vector recycling rules:

- Vectors of length 1 are recycled
- All other vectors must have the same size

Value

- If any of the arguments are rvecs, or if a value for `n_draw` is supplied, then an [rvec](#); otherwise an ordinary R vector.

See Also

- [dchisq\(\)](#), [pchisq\(\)](#), [qchisq\(\)](#), [rchisq\(\)](#) Base R equivalents
- [distributions](#) All base R distributions

Examples

```
x <- rvec(list(c(3, 5.1),
               c(0.1, 2.3)))
dchisq_rvec(x, df = 3)
pchisq_rvec(x, df = 3)

rchisq_rvec(n = 2,
            df = 3:4,
            n_draw = 1000)
```

dexp_rvec

*Exponential Distribution, Using Multiple Draws***Description**

Density, distribution function, quantile function and random generation for the exponential distribution, modified to work with rvecs.

Usage

```
dexp_rvec(x, rate = 1, log = FALSE)
```

```
pexp_rvec(q, rate = 1, lower.tail = TRUE, log.p = FALSE)
```

```
qexp_rvec(p, rate = 1, lower.tail = TRUE, log.p = FALSE)
```

```
rexp_rvec(n, rate = 1, n_draw = NULL)
```

Arguments

x	Quantiles. Can be an rvec.
rate	Vector of rates. See dexp() . Can be an rvec.
log, log.p	Whether to return results on a log scale. Default is FALSE. Cannot be an rvec.
q	Quantiles. Can be an rvec.
lower.tail	Whether to return $P[X \leq x]$, as opposed to $P[X > x]$. Default is TRUE. Cannot be an rvec.
p	Probabilities. Can be an rvec.
n	The length of random vector being created. Cannot be an rvec.
n_draw	Number of random draws in the random vector being created. Cannot be an rvec.

Details

Functions `dexp_rvec()`, `pexp_rvec()`, `qexp_rvec()` and `rexp_rvec()` work like base R functions [dexp\(\)](#), [pexp\(\)](#), [qexp\(\)](#), and [rexp\(\)](#), except that they accept rvecs as inputs. If any input is an rvec, then the output will be too. Function `rexp_rvec()` also returns an rvec if a value for `n_draw` is supplied.

`dexp_rvec()`, `pexp_rvec()`, `qexp_rvec()` and `rexp_rvec()` use [tidyverse](#) vector recycling rules:

- Vectors of length 1 are recycled
- All other vectors must have the same size

Value

- If any of the arguments are rvecs, or if a value for `n_draw` is supplied, then an [rvec](#); otherwise an ordinary R vector.

See Also

- [dexp\(\)](#), [pexp\(\)](#), [qexp\(\)](#), [rexp\(\)](#) Base R equivalents
- [distributions](#) All base R distributions

Examples

```
x <- rvec(list(c(3, 5.1),
               c(0.1, 2.3)))
dexp_rvec(x, rate = 1.5)
pexp_rvec(x, rate = 1.5)

rexp_rvec(n = 2,
          rate = c(1.5, 4),
          n_draw = 1000)
```

df_rvec

*F Distribution, Using Multiple Draws***Description**

Density, distribution function, quantile function and random generation for the F distribution, modified to work with rvecs.

Usage

```
df_rvec(x, df1, df2, ncp = 0, log = FALSE)

pf_rvec(q, df1, df2, ncp = 0, lower.tail = TRUE, log.p = FALSE)

qf_rvec(p, df1, df2, ncp = 0, lower.tail = TRUE, log.p = FALSE)

rf_rvec(n, df1, df2, ncp = 0, n_draw = NULL)
```

Arguments

x	Quantiles. Can be an rvec.
df1, df2	Degrees of freedom. See df() . Can be rvecs.
ncp	Non-centrality parameter. Default is 0. Cannot be an rvec.
log, log.p	Whether to return results on a log scale. Default is FALSE. Cannot be an rvec.
q	Quantiles. Can be an rvec.
lower.tail	Whether to return $P[X \leq x]$, as opposed to $P[X > x]$. Default is TRUE. Cannot be an rvec.
p	Probabilities. Can be an rvec.
n	The length of random vector being created. Cannot be an rvec.
n_draw	Number of random draws in the random vector being created. Cannot be an rvec.

Details

Functions `df_rvec()`, `pf_rvec()`, `pf_rvec()` and `rf_rvec()` work like base R functions `df()`, `pf()`, `qf()`, and `rf()`, except that they accept `rvecs` as inputs. If any input is an `rvec`, then the output will be too. Function `rf_rvec()` also returns an `rvec` if a value for `n_draw` is supplied.

`df_rvec()`, `pf_rvec()`, `pf_rvec()` and `rf_rvec()` use [tidyverse](#) vector recycling rules:

- Vectors of length 1 are recycled
- All other vectors must have the same size

Value

- If any of the arguments are `rvecs`, or if a value for `n_draw` is supplied, then an [rvec](#)
- Otherwise an ordinary R vector.

See Also

- [df\(\)](#), [pf\(\)](#), [qf\(\)](#), [rf\(\)](#) Base R equivalents
- [distributions](#) All base R distributions

Examples

```
x <- rvec(list(c(3, 5.1),
               c(0.1, 2.3)))
df_rvec(x, df1 = 1, df2 = 3)
pf_rvec(x, df1 = 1, df2 = 3)

rf_rvec(n = 2, df1 = 1, df2 = 2:3, n_draw = 1000)
```

dgamma_rvec

Gamma Distribution, Using Multiple Draws

Description

Density, distribution function, quantile function and random generation for the gamma distribution, modified to work with `rvecs`.

Usage

```
dgamma_rvec(x, shape, rate = 1, scale = 1/rate, log = FALSE)
```

```
pgamma_rvec(
  q,
  shape,
  rate = 1,
  scale = 1/rate,
  lower.tail = TRUE,
  log.p = FALSE
```

```

)

qgamma_rvec(
  p,
  shape,
  rate = 1,
  scale = 1/rate,
  lower.tail = TRUE,
  log.p = FALSE
)

rgamma_rvec(n, shape, rate = 1, scale = 1/rate, n_draw = NULL)

```

Arguments

x	Quantiles. Can be an rvec.
shape	Shape parameter. See dgamma() . Can be an rvec.
rate	Rate parameter. See stats::dgamma() . Can be an rvec.
scale	Scale parameter. An alternative to rate. See dgamma() . Can be an rvec.
log, log.p	Whether to return results on a log scale. Default is FALSE. Cannot be an rvec.
q	Quantiles. Can be an rvec.
lower.tail	Whether to return $P[X \leq x]$, as opposed to $P[X > x]$. Default is TRUE. Cannot be an rvec.
p	Probabilities. Can be an rvec.
n	The length of random vector being created. Cannot be an rvec.
n_draw	Number of random draws in the random vector being created. Cannot be an rvec.

Details

Functions [dgamma_rvec\(\)](#), [pgamma_rvec\(\)](#), [pgamma_rvec\(\)](#) and [rgamma_rvec\(\)](#) work like base R functions [dgamma\(\)](#), [pgamma\(\)](#), [qgamma\(\)](#), and [rgamma\(\)](#), except that they accept rvecs as inputs. If any input is an rvec, then the output will be too. Function [rgamma_rvec\(\)](#) also returns an rvec if a value for `n_draw` is supplied.

[dgamma_rvec\(\)](#), [pgamma_rvec\(\)](#), [pgamma_rvec\(\)](#) and [rgamma_rvec\(\)](#) use [tidyverse](#) vector recycling rules:

- Vectors of length 1 are recycled
- All other vectors must have the same size

Value

- If any of the arguments are rvecs, or if a value for `n_draw` is supplied, then an [rvec](#); otherwise an ordinary R vector.

See Also

- [dgamma\(\)](#), [pgamma\(\)](#), [qgamma\(\)](#), [rgamma\(\)](#) Base R equivalents
- [distributions](#) All base R distributions

Examples

```
x <- rvec(list(c(3, 5.1),
               c(0.1, 2.3)))
dgamma_rvec(x, shape = 1)
pgamma_rvec(x, shape = 1)

rgamma_rvec(n = 2,
            shape = 1,
            rate = c(0.5, 1),
            n_draw = 1000)
```

dgeom_rvec

*Geometric Distribution, Using Multiple Draws***Description**

Density, distribution function, quantile function and random generation for the geometric distribution, modified to work with rvecs.

Usage

```
dgeom_rvec(x, prob, log = FALSE)

pgeom_rvec(q, prob, lower.tail = TRUE, log.p = FALSE)

qgeom_rvec(p, prob, lower.tail = TRUE, log.p = FALSE)

rgeom_rvec(n, prob, n_draw = NULL)
```

Arguments

x	Quantiles. Can be an rvec.
prob	Probability of success in each trial. See dgeom() . Can be an rvec.
log, log.p	Whether to return results on a log scale. Default is FALSE. Cannot be an rvec.
q	Quantiles. Can be an rvec.
lower.tail	Whether to return $P[X \leq x]$, as opposed to $P[X > x]$. Default is TRUE. Cannot be an rvec.
p	Probabilities. Can be an rvec.
n	The length of random vector being created. Cannot be an rvec.
n_draw	Number of random draws in the random vector being created. Cannot be an rvec.

Details

Functions `dgeom_rvec()`, `pgeom_rvec()`, `qgeom_rvec()` and `rgeom_rvec()` work like base R functions `dgeom()`, `pgeom()`, `qgeom()`, and `rgeom()`, except that they accept rvecs as inputs. If any input is an rvec, then the output will be too. Function `rgeom_rvec()` also returns an rvec if a value for `n_draw` is supplied.

`dgeom_rvec()`, `pgeom_rvec()`, `qgeom_rvec()` and `rgeom_rvec()` use [tidyverse](#) vector recycling rules:

- Vectors of length 1 are recycled
- All other vectors must have the same size

Value

- If any of the arguments are rvecs, or if a value for `n_draw` is supplied, then an [rvec](#); otherwise an ordinary R vector.
- Unlike base `rgeom()`, `rgeom_rvec()` always returns doubles.

See Also

- [dgeom\(\)](#), [pgeom\(\)](#), [qgeom\(\)](#), [rgeom\(\)](#) Base R equivalents
- [distributions](#) All base R distributions

Examples

```
x <- rvec(list(c(3, 5),
               c(0, 2)))
dgeom_rvec(x, prob = 0.3)
pgeom_rvec(x, prob = 0.3)

rgeom_rvec(n = 2,
           prob = c(0.5, 0.8),
           n_draw = 1000)
```

dhyper_rvec

Hypergeometric Distribution, Using Multiple Draws

Description

Density, distribution function, quantile function and random generation for the hypergeometric distribution, modified to work with rvecs.

Usage

```

dhyper_rvec(x, m, n, k, log = FALSE)

phyper_rvec(q, m, n, k, lower.tail = TRUE, log.p = FALSE)

qhyper_rvec(p, m, n, k, lower.tail = TRUE, log.p = FALSE)

rhyper_rvec(nn, m, n, k, n_draw = NULL)

```

Arguments

x	Quantiles. Can be an rvec.
m	Number of white balls in the urn. See stats::dhyper() . Can be an rvec.
n	Number of black balls in the urn. See stats::rhyper() . Can be an rvec.
k	Number of balls drawn from urn. See stats::dhyper() . Can be an rvec.
log, log.p	Whether to return results on a log scale. Default is FALSE. Cannot be an rvec.
q	Quantiles. Can be an rvec.
lower.tail	Whether to return $P[X \leq x]$, as opposed to $P[X > x]$. Default is TRUE. Cannot be an rvec.
p	Probabilities. Can be an rvec.
nn	The length of the random vector being created. The equivalent of n in other random variate functions. See stats::rhyper() . Cannot be an rvec.
n_draw	Number of random draws in the random vector being created. Cannot be an rvec.

Details

Functions `dhyper_rvec()`, `phyper_rvec()`, `qhyper_rvec()` and `rhyper_rvec()` work like base R functions [dhyper\(\)](#), [phyper\(\)](#), [qhyper\(\)](#), and [rhyper\(\)](#), except that they accept rvecs as inputs. If any input is an rvec, then the output will be too. Function `rhyper_rvec()` also returns an rvec if a value for `n_draw` is supplied.

`dhyper_rvec()`, `phyper_rvec()`, `qhyper_rvec()` and `rhyper_rvec()` use [tidyverse](#) vector recycling rules:

- Vectors of length 1 are recycled
- All other vectors must have the same size

Value

- If any of the arguments are rvecs, or if a value for `n_draw` is supplied, then an [rvec](#); otherwise an ordinary R vector.
- Unlike base `rhyper()`, `rhyper_rvec()` always returns doubles.

See Also

- [dhyper\(\)](#), [phyper\(\)](#), [qhyper\(\)](#), [rhyper\(\)](#) Base R equivalents
- [distributions](#) All base R distributions

Examples

```
x <- rvec(list(c(3, 5),
               c(0, 2)))
dhyper_rvec(x, m = 6, n = 6, k = 5)
phyper_rvec(x, m = 6, n = 6, k = 5)

rhyper_rvec(nn = 2,
            k = c(3, 5),
            m = 6,
            n = 6,
            n_draw = 1000)
```

divorce

*Divorce Rates in New Zealand***Description**

Posterior sample from a model of divorce rates in New Zealand.

Usage

```
divorce
```

Format

A tibble with 30,000 rows and the following variables:

- age: Age, in 5-year age groups, 15-19 to 65+.
- sex: "Female" or "Male".
- draw: Index for random draw.
- rate: Divorce rate, per 1000.

Source

Derived from data in tables "Age at divorces by sex (marriages and civil unions) (Annual-Dec)" and "Estimated Resident Population by Age and Sex (1991+) (Annual-Dec)" in the online database Infoshare on the Statistics New Zealand website, downloaded on 22 March 2023.

dlnorm_rvec

*Log-Normal Distribution, Using Multiple Draws***Description**

Density, distribution function, quantile function and random generation for the log-normal distribution, modified to work with rvecs.

Usage

```
dlnorm_rvec(x, meanlog = 0, sdlog = 1, log = FALSE)
```

```
plnorm_rvec(q, meanlog = 0, sdlog = 1, lower.tail = TRUE, log.p = FALSE)
```

```
qlnorm_rvec(p, meanlog = 0, sdlog = 1, lower.tail = TRUE, log.p = FALSE)
```

```
rlnorm_rvec(n, meanlog = 0, sdlog = 1, n_draw = NULL)
```

Arguments

x	Quantiles. Can be an rvec.
meanlog	Mean of distribution, on log scale. Default is 0. See dlnorm() . Can be an rvec.
sdlog	Standard deviation of distribution, on log scale. Default is 1. See dlnorm() . Can be an rvec.
log, log.p	Whether to return results on a log scale. Default is FALSE. Cannot be an rvec.
q	Quantiles. Can be an rvec.
lower.tail	Whether to return $P[X \leq x]$, as opposed to $P[X > x]$. Default is TRUE. Cannot be an rvec.
p	Probabilities. Can be an rvec.
n	The length of random vector being created. Cannot be an rvec.
n_draw	Number of random draws in the random vector being created. Cannot be an rvec.

Details

Functions `dlnorm_rvec()`, `plnorm_rvec()`, `qlnorm_rvec()` and `rlnorm_rvec()` work like base R functions [dlnorm\(\)](#), [plnorm\(\)](#), [qlnorm\(\)](#), and [rlnorm\(\)](#), except that they accept rvecs as inputs. If any input is an rvec, then the output will be too. Function `rlnorm_rvec()` also returns an rvec if a value for `n_draw` is supplied.

`dlnorm_rvec()`, `plnorm_rvec()`, `qlnorm_rvec()` and `rlnorm_rvec()` use [tidyverse](#) vector recycling rules:

- Vectors of length 1 are recycled
- All other vectors must have the same size

Value

- If any of the arguments are rvecs, or if a value for `n_draw` is supplied, then an [rvec](#); otherwise an ordinary R vector.

See Also

- [dlnorm\(\)](#), [plnorm\(\)](#), [qlnorm\(\)](#), [rlnorm\(\)](#) Base R equivalents
- [distributions](#) All base R distributions

Examples

```
x <- rvec(list(c(3.1, 5.7),
               c(0.2, 2.3)))
dlnorm_rvec(x)
plnorm_rvec(x)

rlnorm_rvec(n = 2,
            meanlog = c(1, 3),
            n_draw = 1000)
```

dmultinom_rvec

*Multinomial Distribution, Using Multiple Draws***Description**

Density function random generation for the multinomial distribution, modified to work with rvecs.

Usage

```
dmultinom_rvec(x, size = NULL, prob, log = FALSE)
```

```
rmultinom_rvec(n, size, prob, n_draw = NULL)
```

Arguments

<code>x</code>	Quantiles. Can be an rvec.
<code>size</code>	Total number of trials. See stats::dmultinom() . Can be an rvec.
<code>prob</code>	Numeric non-negative vector, giving the probability of each outcome. Internally normalized to sum to 1. See stats::dmultinom() . Can be an rvec.
<code>log</code>	Whether to return $\log(p)$ rather than p . Default is FALSE. Cannot be an rvec.
<code>n</code>	The length of random vector being created. Cannot be an rvec.
<code>n_draw</code>	Number of random draws in the random vector being created. Cannot be an rvec.

Details

Functions `dmultinom_rvec()` and `rmultinom_rvec()` work like base R functions `dmultinom()` and `rmultinom()`, except that they accept rvecs as inputs. If any input is an rvec, then the output will be too. Function `rmultinom_rvec()` also returns an rvec if a value for `n_draw` is supplied.

Like the base R functions `dmultinom()` and `[rmultinom()`, `dmultinom_rvec()` and `rmultinom_rvec()` do not recycle their arguments.

Value

- `dmultinom()`
 - If any of the arguments are rvecs, or if a value for `n_draw` is supplied, then an **rvec**; otherwise an ordinary R vector.
- `rmultinom()`
 - If `n` is 1, an rvec or ordinary R vector; otherwise a list of rvecs or ordinary R vectors
- Unlike base `rmultinom()`, `rmultinom_rvec()` always returns doubles.

See Also

- `dmultinom()`, `rmultinom()` Base R equivalents
- [distributions](#) All base R distributions

Examples

```
x <- rvec(list(c(1, 4, 0),
               c(1, 0, 0),
               c(1, 0, 0),
               c(1, 0, 4)))
prob <- c(1/4, 1/4, 1/4, 1/4)
dmultinom_rvec(x = x, prob = prob)
rmultinom_rvec(n = 1,
               size = 100,
               prob = c(0.1, 0.4, 0.2, 0.3),
               n_draw = 1000)
```

dnbinom_rvec

Negative Binomial Distribution, Using Multiple Draws

Description

Density, distribution function, quantile function and random generation for the negative binomial distribution, modified to work with rvecs.

Usage

```
dnbinom_rvec(x, size, prob, mu, log = FALSE)

pnbinom_rvec(q, size, prob, mu, lower.tail = TRUE, log.p = FALSE)

qnbinom_rvec(p, size, prob, mu, lower.tail = TRUE, log.p = FALSE)

rnbinom_rvec(n, size, prob, mu, n_draw = NULL)
```

Arguments

x	Quantiles. Can be an rvec.
size	Number of trials. See dnbinom() . Can be an rvec.
prob	Probability of success in each trial. See dnbinom() . Can be an rvec.
mu	Mean value. See dnbinom() . Can be an rvec.
log, log.p	Whether to return results on a log scale. Default is FALSE. Cannot be an rvec.
q	Quantiles. Can be an rvec.
lower.tail	Whether to return $P[X \leq x]$, as opposed to $P[X > x]$. Default is TRUE. Cannot be an rvec.
p	Probabilities. Can be an rvec.
n	The length of random vector being created. Cannot be an rvec.
n_draw	Number of random draws in the random vector being created. Cannot be an rvec.

Details

Functions [dnbinom_rvec\(\)](#), [pnbinom_rvec\(\)](#), [qnbinom_rvec\(\)](#) and [rnbinom_rvec\(\)](#) work like base R functions [dnbinom\(\)](#), [pnbinom\(\)](#), [qnbinom\(\)](#), and [rnbinom\(\)](#), except that they accept rvecs as inputs. If any input is an rvec, then the output will be too. Function [rnbinom_rvec\(\)](#) also returns an rvec if a value for `n_draw` is supplied.

[dnbinom_rvec\(\)](#), [pnbinom_rvec\(\)](#), [qnbinom_rvec\(\)](#) and [rnbinom_rvec\(\)](#) use [tidyverse](#) vector recycling rules:

- Vectors of length 1 are recycled
- All other vectors must have the same size

Value

- If any of the arguments are rvecs, or if a value for `n_draw` is supplied, then an [rvec](#); otherwise an ordinary R vector.
- Unlike base [rbinom\(\)](#), [rnbinom_rvec\(\)](#) always returns doubles.

See Also

- [dnbinom\(\)](#), [pnbinom\(\)](#), [qnbinom\(\)](#), [rnbinom\(\)](#) Base R equivalents
- [distributions](#) All base R distributions

Examples

```
x <- rvec(list(c(3, 5),
               c(0, 2)))
dnbinom_rvec(x, size = 6, prob = 0.2)
pnbinom_rvec(x, size = 6, prob = 0.2)

rnbinom_rvec(n = 2,
             size = 2,
             mu = c(4, 8),
             n_draw = 1000)
```

dnorm_rvec

*Normal Distribution, Using Multiple Draws***Description**

Density, distribution function, quantile function and random generation for the normal distribution, modified to work with rvecs.

Usage

```
dnorm_rvec(x, mean = 0, sd = 1, log = FALSE)

pnorm_rvec(q, mean = 0, sd = 1, lower.tail = TRUE, log.p = FALSE)

qnorm_rvec(p, mean = 0, sd = 1, lower.tail = TRUE, log.p = FALSE)

rnorm_rvec(n, mean = 0, sd = 1, n_draw = NULL)
```

Arguments

x	Quantiles. Can be an rvec.
mean	Mean of distribution. Default is 0. See dnorm() . Can be an rvec.
sd	Standard deviation. Default is 1. See dnorm() . Can be an rvec.
log, log.p	Whether to return results on a log scale. Default is FALSE. Cannot be an rvec.
q	Quantiles. Can be an rvec.
lower.tail	Whether to return $P[X \leq x]$, as opposed to $P[X > x]$. Default is TRUE. Cannot be an rvec.
p	Probabilities. Can be an rvec.
n	The length of random vector being created. Cannot be an rvec.
n_draw	Number of random draws in the random vector being created. Cannot be an rvec.

Details

Functions `dnorm_rvec()`, `pnorm_rvec()`, `qnorm_rvec()` and `rnorm_rvec()` work like base R functions `dnorm()`, `pnorm()`, `qnorm()`, and `rnorm()`, except that they accept rvecs as inputs. If any input is an rvec, then the output will be too. Function `rnorm_rvec()` also returns an rvec if a value for `n_draw` is supplied.

`dnorm_rvec()`, `pnorm_rvec()`, `qnorm_rvec()` and `rnorm_rvec()` use [tidyverse](#) vector recycling rules:

- Vectors of length 1 are recycled
- All other vectors must have the same size

Value

- If any of the arguments are rvecs, or if a value for `n_draw` is supplied, then an [rvec](#); otherwise an ordinary R vector.

See Also

- [dnorm\(\)](#), [pnorm\(\)](#), [qnorm\(\)](#), [rnorm\(\)](#) Base R equivalents
- [distributions](#) All base R distributions

Examples

```
x <- rvec(list(c(3.1, -5.4),
               c(0.1, 2.3)))
dnorm_rvec(x)
pnorm_rvec(x)

rnorm_rvec(n = 2,
           mean = c(-3, 3),
           sd = c(2, 4),
           n_draw = 1000)
```

dpois_rvec

Poisson Distribution, Using Multiple Draws

Description

Density, distribution function, quantile function and random generation for the Poisson distribution, modified to work with rvecs.

Usage

```
dpois_rvec(x, lambda, log = FALSE)

ppois_rvec(q, lambda, lower.tail = TRUE, log.p = FALSE)

qpois_rvec(p, lambda, lower.tail = TRUE, log.p = FALSE)

rpois_rvec(n, lambda, n_draw = NULL)
```

Arguments

x	Quantiles. Can be an rvec.
lambda	Vector of means. See rpois() . Can be an rvec.
log, log.p	Whether to return results on a log scale. Default is FALSE. Cannot be an rvec.
q	Quantiles. Can be an rvec.
lower.tail	Whether to return $P[X \leq x]$, as opposed to $P[X > x]$. Default is TRUE. Cannot be an rvec.
p	Probabilities. Can be an rvec.
n	The length of random vector being created. Cannot be an rvec.
n_draw	Number of random draws in the random vector being created. Cannot be an rvec.

Details

Functions `dpois_rvec()`, `ppois_rvec()`, `qpois_rvec()` and `rpois_rvec()` work like base R functions [dpois\(\)](#), [ppois\(\)](#), [qpois\(\)](#), and [rpois\(\)](#), except that they accept rvecs as inputs. If any input is an rvec, then the output will be too. Function `rpois_rvec()` also returns an rvec if a value for `n_draw` is supplied.

`dpois_rvec()`, `ppois_rvec()`, `qpois_rvec()` and `rpois_rvec()` use [tidyverse](#) vector recycling rules:

- Vectors of length 1 are recycled
- All other vectors must have the same size

Value

- If any of the arguments are rvecs, or if a value for `n_draw` is supplied, then an [rvec](#); otherwise an ordinary R vector.
- Unlike base `rpois()`, `rpois_rvec()` always returns doubles.

See Also

- [dpois\(\)](#), [ppois\(\)](#), [qpois\(\)](#), [rpois\(\)](#) Base R equivalents
- [distributions](#) All base R distributions

Examples

```
x <- rvec(list(c(3, 5),
               c(1, 2)))
dpois_rvec(x, lambda = 3)
ppois_rvec(x, lambda = 3)

rpois_rvec(n = 2,
           lambda = c(5, 10),
           n_draw = 1000)
```

draws_all

*Logical Operations Across Random Draws***Description**

Apply all or any logical summaries across random draws.

Usage

```
draws_all(x, na_rm = FALSE)

## S3 method for class 'rvec_chr'
draws_all(x, na_rm = FALSE)

## S3 method for class 'rvec'
draws_all(x, na_rm = FALSE)

draws_any(x, na_rm = FALSE)

## S3 method for class 'rvec_chr'
draws_any(x, na_rm = FALSE)

## S3 method for class 'rvec'
draws_any(x, na_rm = FALSE)
```

Arguments

x	An object of class rvec .
na_rm	Whether to remove NAs before calculating summaries. Default is FALSE.

Value

A vector.

See Also

Apply pre-specified functions across draws:

- `draws_min()`
- `draws_max()`
- `draws_median()`
- `draws_mean()`
- `draws_mode()`
- `draws_sd()`
- `draws_var()`
- `draws_cv()`
- `draws_ci()`
- `draws_quantile()`

Apply arbitrary function across draws:

- `draws_fun()`

Examples

```
m <- rbind(a = c(TRUE, FALSE, TRUE),
           b = c(TRUE, TRUE, TRUE),
           c = c(FALSE, FALSE, FALSE))
x <- rvec(m)
x
draws_all(x)
draws_any(x)
```

draws_ci

*Credible Intervals from Random Draws***Description**

Summarise the distribution of random draws in an rvec, using credible intervals.

Usage

```
draws_ci(x, width = 0.95, prefix = NULL, na_rm = FALSE)
```

```
## S3 method for class 'rvec'
```

```
draws_ci(x, width = 0.95, prefix = NULL, na_rm = FALSE)
```

```
## S3 method for class 'rvec_chr'
```

```
draws_ci(x, width = 0.95, prefix = NULL, na_rm = FALSE)
```

Arguments

x	An object of class <code>rvec</code> .
width	Width(s) of credible interval(s). One or more numbers greater than 0 and less than or equal to 1. Default is 0.975.
prefix	String to be added to the names of columns in the result. Defaults to name of x.
na_rm	Whether to remove NAs before calculating summaries. Default is FALSE.

Value

A `tibble` with three columns.

Warning

It is tempting to assign the results of a call to `draws_ci()` to a column in a data frame, as in

```
my_df$ci <- draws_ci(my_rvec)
```

However, creating columns in this way can corrupt an ordinary data frames. For safer options, see the examples below.

See Also

`draws_quantile()` gives more options for forming quantiles.

Other ways of applying pre-specified functions across draws are:

- `draws_all()`
- `draws_any`
- `draws_min()`
- `draws_max()`
- `draws_median()`
- `draws_mean()`
- `draws_mode()`
- `draws_sd()`
- `draws_var()`
- `draws_cv()`
- `draws_quantile()`

Apply arbitrary function across draws:

- `draws_fun()`

Examples

```

set.seed(0)
m <- rbind(a = rnorm(100, mean = 5, sd = 2),
           b = rnorm(100, mean = -3, sd = 3),
           c = rnorm(100, mean = 0, sd = 20))
x <- rvec(m)
x
draws_ci(x)
draws_ci(x, width = c(0.5, 0.99))
draws_ci(x, prefix = "results")

## results from 'draws_ci'
## assigned to a data frame
library(dplyr)
df <- data.frame(x)

## base R approach
cbind(df, draws_ci(x))

## a tidyverse alternative:
## mutate with no '='
df |> mutate(draws_ci(x))

```

draws_fun

*Apply Summary Function Across Random Draws***Description**

Summarise the distribution of random draws in an `rvec`, using a function.

Usage

```

draws_fun(x, fun, ...)

## S3 method for class 'rvec'
draws_fun(x, fun, ...)

```

Arguments

<code>x</code>	An object of class <code>rvec</code> .
<code>fun</code>	A function.
<code>...</code>	Additional arguments passed to <code>fun</code> .

Value

The results from calls to `fun`, combined using `vctrs::vec_c()`.

See Also

Apply pre-specified functions across draws:

- `draws_all()`
- `draws_any()`
- `draws_min()`
- `draws_max()`
- `draws_median()`
- `draws_mean()`
- `draws_mode()`
- `draws_sd()`
- `draws_var()`
- `draws_cv()`
- `draws_ci()`
- `draws_quantile()`

Examples

```
set.seed(0)
m <- rbind(a = rnorm(100, mean = 5, sd = 2),
           b = rnorm(100, mean = -3, sd = 3),
           c = rnorm(100, mean = 0, sd = 20))
x <- rvec(m)
x
draws_fun(x, fun = mad)
draws_fun(x, fun = range)
draws_fun(x, weighted.mean, wt = runif(100))
draws_fun(x, function(x) sd(x) / mean(x))
```

`draws_median`*Medians, Means, and Modes Across Random Draws*

Description

Use means, medians, or modes to summarise the distribution of random draws in an `rvec`.

Usage

```
draws_median(x, na_rm = FALSE)

## S3 method for class 'rvec_chr'
draws_median(x, na_rm = FALSE)

## S3 method for class 'rvec'
```

```
draws_median(x, na_rm = FALSE)

draws_mean(x, na_rm = FALSE)

## S3 method for class 'rvec'
draws_mean(x, na_rm = FALSE)

## S3 method for class 'rvec_chr'
draws_mean(x, na_rm = FALSE)

draws_mode(x, na_rm = FALSE)

## S3 method for class 'rvec'
draws_mode(x, na_rm = FALSE)
```

Arguments

x	An object of class rvec .
na_rm	Whether to remove NAs before calculating summaries. Default is FALSE.

Details

When method is "mode", `reduce_rvec()` returns the most common value for each observation. When there is a tie, it returns NA.

Value

A vector.

See Also

Apply pre-specified functions across draws:

- [draws_all\(\)](#)
- [draws_any\(\)](#)
- [draws_min\(\)](#)
- [draws_max\(\)](#)
- [draws_sd\(\)](#)
- [draws_var\(\)](#)
- [draws_cv\(\)](#)
- [draws_ci\(\)](#)
- [draws_quantile\(\)](#)

Apply arbitrary function across draws:

- [draws_fun\(\)](#)

Examples

```

m <- rbind(a = c(1, 1, 1, 2, 3),
           b = c(2, 4, 0, 2, 3),
           c = c(0, 0, 1, 0, 100))
x <- rvec(m)
x
draws_median(x)
draws_mean(x)
draws_mode(x)

```

draws_min

*Minima and Maxima Across Random Draws***Description**

Apply min or max across random draws.

Usage

```

draws_min(x, na_rm = FALSE)

draws_max(x, na_rm = FALSE)

## S3 method for class 'rvec_chr'
draws_min(x, na_rm = FALSE)

## S3 method for class 'rvec'
draws_min(x, na_rm = FALSE)

## S3 method for class 'rvec_chr'
draws_max(x, na_rm = FALSE)

## S3 method for class 'rvec'
draws_max(x, na_rm = FALSE)

```

Arguments

x	An object of class rvec .
na_rm	Whether to remove NAs before calculating minima and maxima. Default is FALSE.

Value

A vector.

See Also

Apply pre-specified functions across draws:

- `draws_all()`
- `draws_any()`
- `draws_median()`
- `draws_mean()`
- `draws_mode()`
- `draws_sd()`
- `draws_var()`
- `draws_cv()`
- `draws_ci()`
- `draws_quantile()`

Apply arbitrary function across draws:

- `draws_fun()`

Examples

```
m <- rbind(a = c(1, -3, 2),
           b = c(Inf, 0, -Inf),
           c = c(0.2, 0.3, 0.1))
x <- rvec(m)
x
draws_min(x)
draws_max(x)
```

draws_quantile

Quantiles Across Random Draws

Description

Summarise the distribution of random draws in an `rvec`, using quantiles.

Usage

```
draws_quantile(x, probs = c(0.025, 0.25, 0.5, 0.75, 0.975), na_rm = FALSE)
```

```
## S3 method for class 'rvec'
```

```
draws_quantile(x, probs = c(0.025, 0.25, 0.5, 0.75, 0.975), na_rm = FALSE)
```

```
## S3 method for class 'rvec_chr'
```

```
draws_quantile(x, probs = c(0.025, 0.25, 0.5, 0.75, 0.975), na_rm = FALSE)
```

Arguments

x	An object of class rvec .
probs	Vector of probabilities.
na_rm	Whether to remove NAs before calculating summaries. Default is FALSE.

Details

The probs argument defaults to `c(0.025, 0.25, 0.5, 0.75, 0.975)`, the values needed for a median, a 50% credible intervals, and a 95% credible interval.

Value

A [tibble](#).

Warning

It is tempting to assign the results of a call to `draws_quantile()` to a column in a data frame, as in `my_df$quantile <- draws_quantile(my_rvec)`

However, creating data frame columns in this way can corrupt data frames. For safer options, see the examples below.

See Also

[draws_ci\(\)](#) creates simple credible intervals.

Other functions for applying pre-specified functions across draws are:

- [draws_all\(\)](#)
- [draws_any\(\)](#)
- [draws_min\(\)](#)
- [draws_max\(\)](#)
- [draws_median\(\)](#)
- [draws_mean\(\)](#)
- [draws_mode\(\)](#)
- [draws_sd\(\)](#)
- [draws_var\(\)](#)
- [draws_cv\(\)](#)
- [draws_ci\(\)](#)

Apply arbitrary function across draws:

- [draws_fun\(\)](#)

Examples

```

set.seed(0)
m <- rbind(a = rnorm(100, mean = 5, sd = 2),
           b = rnorm(100, mean = -3, sd = 3),
           c = rnorm(100, mean = 0, sd = 20))
x <- rvec(m)
x
draws_quantile(x)

## results from 'draws_quantile'
## assigned to a data frame
library(dplyr)
df <- data.frame(x)

## base R approach
cbind(df, draws_quantile(x))

## a tidyverse alternative:
## mutate with no '='
df |>
  mutate(draws_quantile(x))

```

draws_sd	<i>Standard Deviations, Variances, and Coefficients of Variation Across Random Draws</i>
----------	--

Description

Use standard deviations, variances, or coefficients of variation to summarise the distribution of random draws in an rvec.

Usage

```

draws_sd(x, na_rm = FALSE)

## S3 method for class 'rvec_chr'
draws_sd(x, na_rm = FALSE)

## S3 method for class 'rvec'
draws_sd(x, na_rm = FALSE)

draws_var(x, na_rm = FALSE)

## S3 method for class 'rvec_chr'
draws_var(x, na_rm = FALSE)

## S3 method for class 'rvec'
draws_var(x, na_rm = FALSE)

```

```
draws_cv(x, na_rm = FALSE)

## S3 method for class 'rvec_chr'
draws_cv(x, na_rm = FALSE)

## S3 method for class 'rvec'
draws_cv(x, na_rm = FALSE)
```

Arguments

x	An object of class rvec .
na_rm	Whether to remove NAs before calculating summaries. Default is FALSE.

Details

The coefficient of variation is the standard deviation divided by the mean.

Value

A vector.

See Also

Apply pre-specified functions across draws:

- [draws_all\(\)](#)
- [draws_any\(\)](#)
- [draws_mean\(\)](#)
- [draws_median\(\)](#)
- [draws_mode\(\)](#)
- [draws_min\(\)](#)
- [draws_max\(\)](#)
- [draws_ci\(\)](#)
- [draws_quantile\(\)](#)

Apply arbitrary function across draws:

- [draws_fun\(\)](#)

Examples

```
m <- rbind(a = c(1, 1, 1, 2, 3),
           b = c(2, 4, 0, 2, 3),
           c = c(0, 0, 1, 0, 100))
x <- rvec(m)
x
draws_sd(x)
draws_var(x)
draws_cv(x)
```

dt_rvec	<i>Student t Distribution, Using Multiple Draws</i>
---------	---

Description

Density, distribution function, quantile function and random generation for the t distribution, modified to work with rvecs.

Usage

```
dt_rvec(x, df, ncp = 0, log = FALSE)

pt_rvec(q, df, ncp = 0, lower.tail = TRUE, log.p = FALSE)

qt_rvec(p, df, ncp = 0, lower.tail = TRUE, log.p = FALSE)

rt_rvec(n, df, ncp = 0, n_draw = NULL)
```

Arguments

x	Quantiles. Can be an rvec.
df	Degrees of freedom. See dt() . Can be an rvec.
ncp	Non-centrality parameter. Default is 0. See dt() . Cannot be an rvec.
log, log.p	Whether to return results on a log scale. Default is FALSE. Cannot be an rvec.
q	Quantiles. Can be an rvec.
lower.tail	Whether to return $P[X \leq x]$, as opposed to $P[X > x]$. Default is TRUE. Cannot be an rvec.
p	Probabilities. Can be an rvec.
n	The length of random vector being created. Cannot be an rvec.
n_draw	Number of random draws in the random vector being created. Cannot be an rvec.

Details

Functions `dt_rvec()`, `pt_rvec()`, `qt_rvec()` and `rt_rvec()` work like base R functions [dt\(\)](#), [pt\(\)](#), [qt\(\)](#), and [rt\(\)](#), except that they accept rvecs as inputs. If any input is an rvec, then the output will be too. Function `rt_rvec()` also returns an rvec if a value for `n_draw` is supplied.

`dt_rvec()`, `pt_rvec()`, `qt_rvec()` and `rt_rvec()` use [tidyverse](#) vector recycling rules:

- Vectors of length 1 are recycled
- All other vectors must have the same size

Value

- If any of the arguments are rvecs, or if a value for `n_draw` is supplied, then an [rvec](#); otherwise an ordinary R vector.

See Also

- [dt\(\)](#), [pt\(\)](#), [qt\(\)](#), [rt\(\)](#) Base R equivalents
- [distributions](#) All base R distributions

Examples

```
x <- rvec(list(c(-3.2, 5.3),
               c(-1.6, 2)))
dt_rvec(x, df = 4)
pt_rvec(x, df = 4)

rt_rvec(n = 2,
        df = c(3, 5),
        n_draw = 1000)
```

dunif_rvec

*Uniform Distribution, Using Multiple Draws***Description**

Density, distribution function, quantile function and random generation for the uniform distribution, modified to work with rvecs.

Usage

```
dunif_rvec(x, min = 0, max = 1, log = FALSE)

punif_rvec(q, min = 0, max = 1, lower.tail = TRUE, log.p = FALSE)

qunif_rvec(p, min = 0, max = 1, lower.tail = TRUE, log.p = FALSE)

runif_rvec(n, min = 0, max = 1, n_draw = NULL)
```

Arguments

x	Quantiles. Can be an rvec.
min	Lower limits. Default is 0. See dunif() . Can be an rvec.
max	Upper limited. Default is 1. See dunif() . Can be an rvec.
log, log.p	Whether to return results on a log scale. Default is FALSE. Cannot be an rvec.
q	Quantiles. Can be an rvec.
lower.tail	Whether to return $P[X \leq x]$, as opposed to $P[X > x]$. Default is TRUE. Cannot be an rvec.
p	Probabilities. Can be an rvec.
n	The length of random vector being created. Cannot be an rvec.
n_draw	Number of random draws in the random vector being created. Cannot be an rvec.

Details

Functions `dunif_rvec()`, `punif_rvec()`, `pnif_rvec()` and `runif_rvec()` work like base R functions `dt()`, `pt()`, `qt()`, and `rt()`, except that they accept rvecs as inputs. If any input is an rvec, then the output will be too. Function `runif_rvec()` also returns an rvec if a value for `n_draw` is supplied.

`dunif_rvec()`, `punif_rvec()`, `pnif_rvec()` and `runif_rvec()` use [tidyverse](#) vector recycling rules:

- Vectors of length 1 are recycled
- All other vectors must have the same size

Value

- If any of the arguments are rvecs, or if a value for `n_draw` is supplied, then an [rvec](#); otherwise an ordinary R vector.

See Also

- [dunif\(\)](#), [punif\(\)](#), [qunif\(\)](#), [runif\(\)](#) Base R equivalents
- [distributions](#) All base R distributions

Examples

```
x <- rvec(list(c(0.2, 0.5),
               c(0.6, 0.7)))
dunif_rvec(x)
punif_rvec(x)

runif_rvec(n = 2,
           min = c(0, 0.5),
           n_draw = 1000)
```

dweibull_rvec

Weibull Distribution, Using Multiple Draws

Description

Density, distribution function, quantile function and random generation for the Weibull distribution, modified to work with rvecs.

Usage

```
dweibull_rvec(x, shape, scale = 1, log = FALSE)

pweibull_rvec(q, shape, scale = 1, lower.tail = TRUE, log.p = FALSE)

qweibull_rvec(p, shape, scale = 1, lower.tail = TRUE, log.p = FALSE)

rweibull_rvec(n, shape, scale = 1, n_draw = NULL)
```

Arguments

<code>x</code>	Quantiles. Can be an rvec.
<code>shape</code>	Shape parameter. See dweibull() . Can be an rvec.
<code>scale</code>	Scale parameter. See dweibull() Default is 1. Can be an rvec.
<code>log, log.p</code>	Whether to return results on a log scale. Default is FALSE. Cannot be an rvec.
<code>q</code>	Quantiles. Can be an rvec.
<code>lower.tail</code>	Whether to return $P[X \leq x]$, as opposed to $P[X > x]$. Default is TRUE. Cannot be an rvec.
<code>p</code>	Probabilities. Can be an rvec.
<code>n</code>	The length of random vector being created. Cannot be an rvec.
<code>n_draw</code>	Number of random draws in the random vector being created. Cannot be an rvec.

Details

Functions `dweibull_rvec()`, `pweibull_rvec()`, `qweibull_rvec()` and `rweibull_rvec()` work like base R functions [dt\(\)](#), [pt\(\)](#), [qt\(\)](#), and [rt\(\)](#), except that they accept rvecs as inputs. If any input is an rvec, then the output will be too. Function `rweibull_rvec()` also returns an rvec if a value for `n_draw` is supplied.

`dweibull_rvec()`, `pweibull_rvec()`, `qweibull_rvec()` and `rweibull_rvec()` use [tidyverse](#) vector recycling rules:

- Vectors of length 1 are recycled
- All other vectors must have the same size

Value

- If any of the arguments are rvecs, or if a value for `n_draw` is supplied, then an [rvec](#); otherwise an ordinary R vector.

See Also

- [dweibull\(\)](#), [pweibull\(\)](#), [qweibull\(\)](#), [rweibull\(\)](#) Base R equivalents
- [distributions](#) All base R distributions

Examples

```
x <- rvec(list(c(3.2, 4.5),
               c(0.6, 0.7)))
dweibull_rvec(x, shape = 2)
pweibull_rvec(x, shape = 2)

rweibull_rvec(n = 2,
              shape = c(2, 3),
              n_draw = 1000)
```

extract_draw	<i>Extract a Single Draw From an Rvec</i>
--------------	---

Description

Extract a single draw from `x`. If a value is supplied for `i`, extract the `i`th draw; otherwise extract a random draw.

Usage

```
extract_draw(x, i = NULL)
```

Arguments

<code>x</code>	An rvec .
<code>i</code>	Index for the draw to be extracted. A number between 1 and <code>n_draw(x)</code> . If no value is supplied, a draw is chosen at random.

Value

A vector, with type

- double, if `x` has class `"rvec_dbl"`,
- integer, if `x` has class `"rvec_int"`,
- character, if `x` has class `"rvec_chr"`,
- logical, if `x` has class `"rvec_lgl"`.

See Also

[n_draw\(\)](#) Number of draws

Examples

```
x <- rvec(matrix(1:50, ncol = 5))
extract_draw(x, i = 1)
extract_draw(x)
```

if_else_rvec

*Vectorised If-Else, When Condition is an Rvec***Description**

A version of `if_else` for the situation where condition is an rvec.

Usage

```
if_else_rvec(condition, true, false, missing = NULL, size = NULL)
```

Arguments

condition	An object of class <code>rvec_lgl</code> .
true, false	Vectors (including rvecs) to use for TRUE and FALSE values of condition.
missing	Vectors to use for NA values of condition. Optional.
size	Length of output. Optional.

Value

An rvec with the same number of `draws` as condition.

See Also

- base R function `ifelse()` does not work correctly if any of the inputs are rvecs.
- `dplyr` function `if_else` works correctly if arguments `true`, `false` or `missing` are rvecs, but not if argument `condition` is an rvec.

Examples

```
x <- rvec(list(c(1, 11),
               c(2, 5),
               c(22, 6)))

x > 10 ## rvec_lgl

## if_else_rvec needed when
## 'condition' is an rvec
if_else_rvec(x > 10, 10, x)

## dplyr::if_else works when
## 'true', 'false', or 'missing'
## (but not 'condition') are rvecs
library(dplyr)
if_else(c(TRUE, FALSE, TRUE), x, 100)
```

is_rvec	<i>Is an Object an Rvec</i>
---------	-----------------------------

Description

Test whether x inherits from class "rvec".

Usage

```
is_rvec(x)
```

Arguments

x An object.

Value

TRUE or FALSE.

See Also

- [rvec\(\)](#) to create an rvec
- [as.matrix\(\)](#), [as_list_col\(\)](#), to convert an rvec into other formats

Examples

```
x <- rvec_dbl()
is_rvec(x)
```

map_rvec	<i>Apply a Function and Put Results in an Rvec</i>
----------	--

Description

Apply function .f to each element of .x, and then combine the results into an rvec with the same length as .x.

Usage

```
map_rvec(.x, .f, ...)
```

Arguments

.x A vector.
.f A function.
... Additional arguments passed to .f.

Details

Each call to function `.f` should produce an `rvec` with length 1.

Value

An `rvec` with the same length as `.x`.

See Also

`map_rvec()` is based on the map functions in package `purrr`, though the internal implementation is different.

Base R functions `sapply()` and `vapply()` do not work properly with rvecs. `lapply()` works, but to combine the results into a single rvec, functions such as `c()` or `vec_c()` are needed.

Examples

```
l <- list(a = rvec(matrix(1:2, 1)),
         b = rvec(matrix(1:4, 2)),
         c = rvec(matrix(1:6, 3)))
l
map_rvec(l, sum)

## sapply does not work with rvecs
sapply(l, sum)
```

matrixOps.rvec

Matrix Multiplication with Rvecs

Description

Matrix multiplication `%*%` can be used with `rvecs`. However, in contrast to standard R vectors, multiplying an rvec by a matrix does not produce a row or column vector. Instead it produces an ordinary rvec, with no dimensions.

Usage

```
## S3 method for class 'rvec'
matrixOps(x, y)
```

Arguments

`x, y` Vectors, matrices, or rvecs.

Value

An rvec, if `x` or `y` is an rvec.

Examples

```

A <- matrix(c(10, 10, 10,
             11, 11, 11),
           nrow = 2, byrow = TRUE)
x <- rvec(list(c(1, 2),
               c(3, 4),
               c(5, 6)))

A %*% x

## matrix multiplication with an
## ordinary R matrix produces
## a row or column vector
y <- c(1, 3, 5)
A %*% y

```

missing

*Missing, Finite, and Infinite Values in Rvecs***Description**

Detect or remove missing and infinite values in rvecs. Operations are done independently on each draw, though `na.omit()`, `na.exclude()`, and `na.fail()` also look across draws.

Usage

```

## S3 method for class 'rvec'
anyNA(x, recursive = FALSE)

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

## S3 method for class 'rvec'
na.exclude(object, ...)

## S3 method for class 'rvec'
na.omit(object, ...)

```

Arguments

<code>x, object</code>	An rvec .
<code>recursive</code>	Whether <code>anyNA()</code> should be applied recursively to lists. Ignored when <code>x</code> is an <code>rvec</code> .
<code>...</code>	Currently ignored.

Details

The behavior of the `rvec` methods for `is.na()`, `is.nan()`, `is.finite()`, and `is.infinite()` differs from the standard `vctrs` behavior, which is to return a logical vector with length equal to `length(x)`. With `rvecs`, the standard `vctrs` behavior would entail summarising across draws, which is the job of the `draws_*` functions.

Value

- `anyNA()` A logical `rvec` with length 1.
- `is.na()`, `is.nan()`, `is.finite()`, `is.infinite()` A logical `rvec` with the same length as the original `rvec`.
- `na.omit()`, `na.exclude()` An `rvec` with the same class as the original `rvec`, minus any elements that have NAs in any draws.
- `na.fail()` The original `rvec`, or an error.

See Also

- `if_else_rvec()` Modify individual values within draws
- `is.na()`, `is.nan()`, `is.finite()`, `is.infinite()`, `anyNA()`, `na.omit()`, `na.exclude()` Base R functions
- `vctrs::vec_detect_missing()` Test whether all draws for an observation are missing
- `vctrs::vec_detect_complete()` Test whether any draws for an observation are missing
- `draws_any()`, `draws_all()` Summarise across draws

Examples

```
x <- rvec(list(c(1.2, NA),
               c(Inf, 3),
               c(-1, NaN)))

## return a logical rvec
is.na(x)
is.nan(x)
is.finite(x)
is.infinite(x)

## return a logical rvec with length 1
anyNA(x)

## summarise across draws
draws_any(anyNA(x))

## return an NA-free version of 'x'
na.omit(x)
na.exclude(x)

## use 'if_else_rvec' to modify values
## within rvec
```

```

if_else_rvec(is.na(x), 999, x)

## vctrs functions
library(vctrs, warn.conflicts = FALSE)
## all draws missing
vec_detect_missing(x)
## any draws missing
vec_detect_complete(x)

```

new_rvec_blank	<i>Create an Empty Rvec</i>
----------------	-----------------------------

Description

Create an rvec, filled with 0, "", or FALSE, with a given length or number of draws.

Usage

```

new_rvec_chr(length = 0, n_draw = 1000)

new_rvec_dbl(length = 0, n_draw = 1000)

new_rvec_int(length = 0, n_draw = 1000)

new_rvec_lgl(length = 0, n_draw = 1000)

```

Arguments

length	Desired length of rvec. Default is 0.
n_draw	Number of draws of rvec. Default is 1000.

Value

An rvec.

See Also

- [rvec\(\)](#), [rvec_chr\(\)](#), [rvec_dbl\(\)](#), [rvec_int\(\)](#), [rvec_lgl\(\)](#) Create an rvec from data.
- [n_draw\(\)](#) Query number of draws.

Examples

```

new_rvec_int()
new_rvec_lgl(length = 1, n_draw = 5)

x <- new_rvec_dbl(length = 2)
x[1] <- rnorm_rvec(n = 1, n_draw = 1000)
x[2] <- runif_rvec(n = 1, n_draw = 1000)

```

new_rvec_deprecated	Create a Blank Rvec
---------------------	---------------------

Description

[Deprecated]

Usage

```
new_rvec(x = double(), length = 0, n_draw = 1000)
```

Arguments

x	Object with the intended type. Default is double().
length	Desired length of rvec. Default is 0.
n_draw	Number of draws of rvec. Default is 1000.

Details

Create an rvec, consisting entirely of NAs, with a given length and number of draws.

The type of the object is taken from x. If `typeof(x)` is "integer", for instance, then `new_rvec()` returns an object of class "rvec_int".

Value

An rvec.

See Also

- [new_rvec_chr\(\)](#), [new_rvec_dbl\(\)](#), [new_rvec_int\(\)](#), [new_rvec_lgl\(\)](#) Replacements for `rvec_new()`
- [rvec\(\)](#) [rvec_chr\(\)](#), [rvec_dbl\(\)](#), [rvec_int\(\)](#), [rvec_lgl\(\)](#) Create an rvec from data.
- [n_draw\(\)](#) Query number of draws.

Examples

```
suppressWarnings({
  new_rvec()
  new_rvec(TRUE, length = 3, n_draw = 100)

  x <- new_rvec(length = 2)
  x[1] <- rnorm_rvec(n = 1, n_draw = 1000)
  x[2] <- runif_rvec(n = 1, n_draw = 1000)
})
```

n_draw	<i>Query Number of Draws</i>
--------	------------------------------

Description

Get a count of the random draws held by `x`. If `x` does not hold random draws, then `n_draw()` throws an error.

Usage

```
n_draw(x)

## Default S3 method:
n_draw(x)

## S3 method for class 'rvec'
n_draw(x)
```

Arguments

`x` An object that holds random draws, eg an [rvec](#).

Value

An integer scalar.

See Also

- [is_rvec\(\)](#) to test if an object is an `rvec`.

Examples

```
m <- matrix(1:40, nrow = 4, ncol = 10)
x <- rvec(m)
n_draw(x)
```

pool_draws	<i>Pool Draws</i>
------------	-------------------

Description

Combine draws within each combination of grouping or 'by' variables in a data frame.

Usage

```
pool_draws(data, by = NULL)

## S3 method for class 'data.frame'
pool_draws(data, by = NULL)

## S3 method for class 'grouped_df'
pool_draws(data, by = NULL)
```

Arguments

<code>data</code>	A data frame with one or more rvecs. Can be grouped .
<code>by</code>	The variables distinguishing units after combining. Used if data is not grouped .

Details

Each combination of grouping or 'by' variables must have the same number of rows.

Value

A data frame.

See Also

- [collapse_to_rvec\(\)](#) Convert from 'draws-and-value' format to rvec format
- [expand_from_rvec\(\)](#) Convert from rvec format to 'draws-and-value' format

Examples

```
library(dplyr, warn.conflicts = FALSE)

df <- tibble(
  a = c(1, 1, 2, 2),
  x = rvec(list(1:2, 3:4, 5:6, 7:8))
)
df
df |> pool_draws(by = a)
df |> group_by(a) |> pool_draws()
df |> pool_draws()

df_big <- tibble(
  a = c(1, 1, 2, 2, 1, 1, 2, 2),
  b = c(1, 1, 1, 1, 2, 2, 2, 2),
  x = rvec(list(1:2, 3:4, 5:6, 7:8,
                9:10, 11:12, 13:14, 15:16)),
  y = rvec(list(1:3, 4:6, 7:9, 10:12,
                13:15, 16:18, 19:21, 22:24))
)
df_big |> pool_draws(by = c(a, b))
df_big |> group_by(a, b) |> pool_draws()
df_big |> pool_draws(by = a)
```

prob

Calculate Probabilities from Random Draws

Description

Convert an rvec of logical values (an [rvec_lgl](#)) into a vector of probabilities.

Usage

```
prob(x, na_rm = FALSE)

## S3 method for class 'rvec_lgl'
prob(x, na_rm = FALSE)

## S3 method for class 'logical'
prob(x, na_rm = FALSE)
```

Arguments

x	An object of class rvec_lgl .
na_rm	Whether to remove NAs before calculating summaries. Default is FALSE.

Details

`prob()` is essentially just [draws_mean\(\)](#) with a different name. The proportion of draws that are TRUE is used as an estimate of the underlying probability. The different name can make the intent of the code clearer.

Value

A logical vector with the same length as x.

See Also

- [draws_mean\(\)](#) Means across draws. Gives the same result as `prob` when applied to logical rvecs.

Examples

```
m <- rbind(c(FALSE, TRUE),
           c(TRUE, TRUE),
           c(FALSE, FALSE))
x <- rvec(m)
x
prob(x)

## logical rvec created on the fly
## through operations such as '>'
```

```

m <- rbind(c(-1, 1.3, 2),
           c(2, 0.1, -1),
           c(Inf, 0, -0.5))
y <- rvec(m)
y
prob(y > 0)
prob(y >= 0)
prob(y^2 > 0)

```

rank

*Sample Ranks, Including Rvecs***Description**

Calculate sample ranks for ordinary vectors or for rvecs. In the case of rvecs, ranks are calculated independently for each draw.

Usage

```

rank(
  x,
  na.last = TRUE,
  ties.method = c("average", "first", "last", "random", "max", "min")
)

```

Arguments

<code>x</code>	An ordinary vector or an <code>rvec()</code> .
<code>na.last</code>	Treatment of NAs. Options are TRUE, FALSE, or "keep". See <code>base::rank()</code> for details.
<code>ties.method</code>	Treatment of ties. See <code>base::rank()</code> for details.

Details

To enable different behavior for rvecs and for ordinary vectors, the base R function `base::rank()` is turned into a generic, with `base::rank()` as the default.

For details on the calculations, see the documentation for `base::rank()`.

Value

An object of class `rvec_int()` if `x` is an rvec. Otherwise an ordinary integer vector.

Examples

```

x <- rvec(list(c(3, 30),
               c(0, 100)))
rank(x)

```

reg_post	<i>Posterior Sample from Linear Regression</i>
----------	--

Description

Posterior sample for parameters from a linear regression model.

Usage

```
reg_post
```

Format

A matrix with 200 columns and the following rows:

- alpha: Intercept parameter
- beta: Slope parameter
- sigma: Standard deviation of error term

Source

reg_post contains values from the second half of the line dataset in package [coda](#). The line dataset draws on the BUGS manual: Spiegelhalter, D.J., Thomas, A., Best, N.G. and Gilks, W.R. (1995) BUGS: Bayesian inference using Gibbs Sampling, Version 0.5, MRC Biostatistics Unit, Cambridge.

rvec	<i>Create an Rvec from Data</i>
------	---------------------------------

Description

Create an object of class "rvec", based on input data.

Usage

```
rvec(x)

rvec_chr(x = NULL)

rvec_dbl(x = NULL)

rvec_int(x = NULL)

rvec_lgl(x = NULL)
```

Arguments

x A matrix, a list of vectors, an atomic vector, or an rvec.

Details

Class "rvec" has four subclasses, each dealing with a different type:

- "rvec_dbl" doubles
- "rvec_int" integers
- "rvec_lgl" logical
- "rvec_chr" character

These subclasses are analogous to [double\(\)](#), [integer\(\)](#), [logical\(\)](#), and [character\(\)](#) vectors.

Function [rvec\(\)](#) chooses the subclass, based on **x**. Functions [rvec_dbl\(\)](#), [rvec_int\(\)](#), [rvec_lgl\(\)](#), and [rvec_chr\(\)](#) each create objects of a particular subclass.

x can be

- a matrix, where each row is a set of draws for an unknown quantity;
- a list, where each element is a set of draws;
- an atomic vector, which is treated as a single-column matrix; or
- an rvec.

Value

An rvec with the following class:

- [rvec_dbl\(\)](#): "rvec_dbl"
- [rvec_int\(\)](#): "rvec_int"
- [rvec_lgl\(\)](#): "rvec_lgl"
- [rvec_chr\(\)](#): "rvec_chr"
- [rvec\(\)](#): "rvec_chr", "rvec_dbl", "rvec_int", or "rvec_lgl"

See Also

- [new_rvec\(\)](#) Create a blank rvec.
- [collapse_to_rvec\(\)](#) Create rvecs within a data frame.
- [rnorm_rvec\(\)](#), [rbinom_rvec\(\)](#), etc. Create rvecs representing probability distributions.

Examples

```
m <- rbind(c(-1.5, 2, 0.2),
           c(-2.3, 3, 1.2))
rvec_dbl(m)

l <- list(rpois(100, lambda = 10.2),
         rpois(100, lambda = 5.5))
```

```

rvec(1)

rvec(letters[1:5])

l <- list(a = c(TRUE, FALSE),
          b = c(FALSE, TRUE))
rvec(l)

```

sd	<i>Standard Deviation, Including Rvecs</i>
----	--

Description

Calculate standard deviation of `x`, where `x` can be an `rvec`. If `x` is an `rvec`, separate standard deviations are calculated for each draw.

Usage

```
sd(x, na.rm = FALSE)
```

Arguments

<code>x</code>	A numeric vector or R object, including an <code>rvec()</code> .
<code>na.rm</code>	Whether to remove NAs before calculating standard deviations.

Details

To enable different behavior for `rvecs` and for ordinary vectors, the base R function `stats::sd()` is turned into a generic, with `stats::sd()` as the default.

For details on the calculations, see the documentation for `stats::sd()`.

Value

An `rvec`, if `x` is an `rvec`. Otherwise typically a numeric vector.

See Also

[var\(\)](#)

Examples

```

x <- rvec(cbind(rnorm(10), rnorm(10, sd = 20)))
x
sd(x)

```

var

*Correlation, Variance and Covariance (Matrices), Including Rvecs***Description**

Calculate correlations and variances, including when `x` or `y` is an `rvec`.

Usage

```
var(x, y = NULL, na.rm = FALSE, use)
```

Arguments

<code>x</code>	A numeric vector, matrix, data frame, or <code>rvec()</code> .
<code>y</code>	NULL (default) or a vector, matrix, data frame, or <code>rvec</code> with compatible dimensions to <code>x</code> .
<code>na.rm</code>	Whether NAs removed before calculations.
<code>use</code>	Calculation method. See <code>stats::var()</code> .

Details

To enable different behavior for `rvecs` and for ordinary vectors, the base R function `stats::var()` is turned into a generic, with `stats::var()` as the default.

For details on the calculations, see the documentation for `stats::var()`.

Value

An `rvec`, if `x` or `y` is an `rvec`. Otherwise typically a numeric vector or matrix.

See Also

`sd()`

Examples

```
x <- rvec(cbind(rnorm(10), rnorm(10, sd = 20)))
x
var(x)
```

weighted_mean*Calculate Weighted Summaries*

Description

Calculate weighted

- means
- medians
- MADs (mean absolute deviations)
- variances
- standard deviations.

These functions all work with ordinary vectors and with [rvecs](#).

Usage

```
weighted_mean(x, wt = NULL, na_rm = FALSE)

## Default S3 method:
weighted_mean(x, wt = NULL, na_rm = FALSE)

## S3 method for class 'rvec'
weighted_mean(x, wt = NULL, na_rm = FALSE)

weighted_mad(x, wt = NULL, na_rm = FALSE)

## Default S3 method:
weighted_mad(x, wt = NULL, na_rm = FALSE)

## S3 method for class 'rvec'
weighted_mad(x, wt = NULL, na_rm = FALSE)

weighted_median(x, wt = NULL, na_rm = FALSE)

## Default S3 method:
weighted_median(x, wt = NULL, na_rm = FALSE)

## S3 method for class 'rvec'
weighted_median(x, wt = NULL, na_rm = FALSE)

weighted_sd(x, wt = NULL, na_rm = FALSE)

## Default S3 method:
weighted_sd(x, wt = NULL, na_rm = FALSE)
```

```
## S3 method for class 'rvec'
weighted_sd(x, wt = NULL, na_rm = FALSE)

weighted_var(x, wt = NULL, na_rm = FALSE)

## Default S3 method:
weighted_var(x, wt = NULL, na_rm = FALSE)

## S3 method for class 'rvec'
weighted_var(x, wt = NULL, na_rm = FALSE)
```

Arguments

<code>x</code>	Quantity being summarised. An ordinary vector or an rvec .
<code>wt</code>	Weights. An ordinary vector, an rvec , or NULL (the default.) If NULL, an un-weighted summary is returned.
<code>na_rm</code>	Whether to remove NAs in <code>x</code> or <code>wt</code> before calculating. Default is FALSE. See matrixStats::weightedMean() for a description of the algorithm used.

Details

`x` and `wt` must have the same length.

Internally the calculations are done by [matrixStats](#) functions such as [matrixStats::weightedMean\(\)](#) and [matrixStats::colWeightedMeans\(\)](#).

Value

If `x` or `wt` or is [rvec](#), then an [rvec](#) of length 1. Otherwise, a scalar.

See Also

- Functions [mean\(\)](#), [median\(\)](#), [mad\(\)](#), [var\(\)](#), [sd\(\)](#) for unweighted data all have methods for [rvecs](#)
- The original [matrixStats](#) weighted summary functions have additional options not implemented in the functions here.
- [weighted.mean\(\)](#) is a base R function for weighted data
- For numeric summaries of draws in an [rvec](#), use [draws_median\(\)](#), [draws_mean\(\)](#), [draws_quantile\(\)](#), [draws_fun\(\)](#).

Examples

```
## 'x' is rvec, 'wt' is ordinary vector
v <- rvec(list(c(1, 11),
               c(2, 12),
               c(7, 17)))
weights <- c(40, 80, 72)
weighted_mean(v, wt = weights)
```

```
## 'x' is ordinary vector, 'wt' is rvec
y <- c(1, 2, 3)
w <- rvec(list(c(100, 200),
               c(210, 889),
               c(200, 200)))
weighted_mean(y, wt = w)
weighted_mean(y, wt = w, na_rm = TRUE)
```

Index

* datasets

- divorce, [22](#)
- reg_post, [57](#)

anyNA(), [50](#)
anyNA.rvec(missing), [49](#)
as.matrix(), [47](#)
as_list_col, [5](#)
as_list_col(), [4](#), [8](#), [47](#)

base::rank(), [56](#)

c(), [48](#)
character(), [58](#)
collapse_to_rvec, [6](#)
collapse_to_rvec(), [3](#), [4](#), [54](#), [58](#)

dbeta(), [9](#), [10](#)
dbeta_rvec, [9](#)
dbeta_rvec(), [3](#)
dbinom(), [11](#)
dbinom_rvec, [10](#)
dbinom_rvec(), [3](#)
dcauchy(), [12](#), [13](#)
dcauchy_rvec, [12](#)
dcauchy_rvec(), [3](#)
dchisq(), [14](#)
dchisq_rvec, [13](#)
dchisq_rvec(), [3](#)
dexp(), [15](#), [16](#)
dexp_rvec, [15](#)
dexp_rvec(), [4](#)
df(), [16](#), [17](#)
df_rvec, [16](#)
df_rvec(), [4](#)
dgamma(), [18](#), [19](#)
dgamma_rvec, [17](#)
dgamma_rvec(), [4](#)
dgeom(), [19](#), [20](#)
dgeom_rvec, [19](#)
dgeom_rvec(), [4](#)
dhyper(), [21](#)
dhyper_rvec, [20](#)
dhyper_rvec(), [4](#)
distributions, [10](#), [11](#), [13](#), [14](#), [16](#), [17](#), [19–21](#),
[24–26](#), [28](#), [29](#), [42–44](#)
divorce, [22](#)
divorce(), [5](#)
dlnorm(), [23](#), [24](#)
dlnorm_rvec, [23](#)
dlnorm_rvec(), [4](#)
dmultinom(), [4](#), [25](#)
dmultinom_rvec, [24](#)
dnbinom(), [26](#)
dnbinom_rvec, [25](#)
dnbinom_rvec(), [4](#)
dnorm(), [27](#), [28](#)
dnorm_rvec, [27](#)
dnorm_rvec(), [4](#)
double(), [58](#)
dpois(), [29](#)
dpois_rvec, [28](#)
dpois_rvec(), [4](#)
draws, [46](#)
draws_*, [50](#)
draws_all, [30](#)
draws_all(), [4](#), [32](#), [34](#), [35](#), [37](#), [38](#), [40](#), [50](#)
draws_any, [32](#)
draws_any(draws_all), [30](#)
draws_any(), [4](#), [34](#), [35](#), [37](#), [38](#), [40](#), [50](#)
draws_ci, [31](#)
draws_ci(), [3](#), [4](#), [31](#), [34](#), [35](#), [37](#), [38](#), [40](#)
draws_cv(draws_sd), [39](#)
draws_cv(), [4](#), [31](#), [32](#), [34](#), [35](#), [37](#), [38](#)
draws_fun, [33](#)
draws_fun(), [4](#), [31](#), [32](#), [35](#), [37](#), [38](#), [40](#), [62](#)
draws_max(draws_min), [36](#)
draws_max(), [4](#), [31](#), [32](#), [34](#), [35](#), [38](#), [40](#)
draws_mean, [62](#)

draws_mean (draws_median), 34
 draws_mean(), 4, 31, 32, 34, 37, 38, 40, 55
 draws_median, 34
 draws_median(), 4, 31, 32, 34, 37, 38, 40, 62
 draws_min, 36
 draws_min(), 4, 31, 32, 34, 35, 38, 40
 draws_mode (draws_median), 34
 draws_mode(), 4, 31, 32, 34, 37, 38, 40
 draws_quantile, 37
 draws_quantile(), 4, 31, 32, 34, 35, 37, 40, 62
 draws_sd, 39
 draws_sd(), 4, 31, 32, 34, 35, 37, 38
 draws_var (draws_sd), 39
 draws_var(), 4, 31, 32, 34, 35, 37, 38
 dt(), 41–44
 dt_rvec, 41
 dt_rvec(), 4
 dunif(), 42, 43
 dunif_rvec, 42
 dunif_rvec(), 4
 dweibull(), 44
 dweibull_rvec, 43
 dweibull_rvec(), 4

 expand_from_rvec (collapse_to_rvec), 6
 expand_from_rvec(), 4, 6, 54
 extract_draw, 45
 extract_draw(), 3

 if_else_rvec, 46
 if_else_rvec(), 3, 50
 ifelse(), 46
 integer(), 58
 is.finite(), 50
 is.infinite(), 50
 is.na(), 50
 is.na.rvec (missing), 49
 is.nan(), 50
 is_rvec, 47
 is_rvec(), 4, 53

 lapply(), 48
 logical(), 58

 mad(), 62
 map_rvec, 47
 map_rvec(), 3
 matrixOps.rvec, 48

 matrixStats::colWeightedMeans(), 62
 matrixStats::weightedMean(), 62
 mean(), 62
 median(), 62
 missing, 49

 n_draw, 53
 n_draw(), 4, 7, 45, 51, 52
 na.exclude(), 50
 na.exclude.rvec (missing), 49
 na.omit(), 50
 na.omit.rvec (missing), 49
 new_rvec (new_rvec_deprecated), 52
 new_rvec(), 58
 new_rvec_blank, 51
 new_rvec_chr (new_rvec_blank), 51
 new_rvec_chr(), 3, 52
 new_rvec_dbl (new_rvec_blank), 51
 new_rvec_dbl(), 3, 52
 new_rvec_deprecated, 52
 new_rvec_int (new_rvec_blank), 51
 new_rvec_int(), 3, 52
 new_rvec_lgl (new_rvec_blank), 51
 new_rvec_lgl(), 3, 52

 pbeta(), 10
 pbeta_rvec (dbeta_rvec), 9
 pbinom(), 11
 pbinom_rvec (dbinom_rvec), 10
 pcauchy(), 13
 pcauchy_rvec (dcauchy_rvec), 12
 pchisq(), 14
 pchisq_rvec (dchisq_rvec), 13
 pexp(), 15, 16
 pexp_rvec (dexp_rvec), 15
 pf(), 17
 pf_rvec (df_rvec), 16
 pgamma(), 18, 19
 pgamma_rvec (dgamma_rvec), 17
 pgeom(), 20
 pgeom_rvec (dgeom_rvec), 19
 phyper(), 21
 phyper_rvec (dhyper_rvec), 20
 plnorm(), 23, 24
 plnorm_rvec (dlnorm_rvec), 23
 pnbinom(), 26
 pnbinom_rvec (dnbinom_rvec), 25
 pnorm(), 28
 pnorm_rvec (dnorm_rvec), 27

pool_draws, 53
 pool_draws(), 3
 ppois(), 29
 ppois_rvec (dpois_rvec), 28
 prob, 55
 pt(), 41–44
 pt_rvec (dt_rvec), 41
 punif(), 43
 punif_rvec (dunif_rvec), 42
 pweibull(), 44
 pweibull_rvec (dweibull_rvec), 43

 qbeta(), 10
 qbeta_rvec (dbeta_rvec), 9
 qbinom(), 11
 qbinom_rvec (dbinom_rvec), 10
 qcauchy(), 13
 qcauchy_rvec (dcauchy_rvec), 12
 qchisq(), 14
 qchisq_rvec (dchisq_rvec), 13
 qexp(), 15, 16
 qexp_rvec (dexp_rvec), 15
 qf(), 17
 qf_rvec (df_rvec), 16
 qgamma(), 18, 19
 qgamma_rvec (dgamma_rvec), 17
 qgeom(), 20
 qgeom_rvec (dgeom_rvec), 19
 qhyper(), 21
 qhyper_rvec (dhyper_rvec), 20
 qlnorm(), 23, 24
 qlnorm_rvec (dlnorm_rvec), 23
 qnbinom(), 26
 qnbinom_rvec (dnbinom_rvec), 25
 qnorm(), 28
 qnorm_rvec (dnorm_rvec), 27
 qpois(), 29
 qpois_rvec (dpois_rvec), 28
 qt(), 41–44
 qt_rvec (dt_rvec), 41
 qunif(), 43
 qunif_rvec (dunif_rvec), 42
 qweibull(), 44
 qweibull_rvec (dweibull_rvec), 43

 rank, 56
 rbeta(), 10
 rbeta_rvec (dbeta_rvec), 9
 rbinom(), 11
 rbinom_rvec (dbinom_rvec), 10
 rbinom_rvec(), 58
 rcauchy(), 13
 rcauchy_rvec (dcauchy_rvec), 12
 rchisq(), 14
 rchisq_rvec (dchisq_rvec), 13
 reg_post, 57
 reg_post(), 5
 rexp(), 15, 16
 rexp_rvec (dexp_rvec), 15
 rf(), 17
 rf_rvec (df_rvec), 16
 rgamma(), 18, 19
 rgamma_rvec (dgamma_rvec), 17
 rgeom(), 20
 rgeom_rvec (dgeom_rvec), 19
 rhyper(), 21
 rhyper_rvec (dhyper_rvec), 20
 rlnorm(), 23, 24
 rlnorm_rvec (dlnorm_rvec), 23
 rmultinom(), 25
 rmultinom_rvec (dmultinom_rvec), 24
 rnbinom(), 26
 rnbinom_rvec (dnbinom_rvec), 25
 rnorm(), 28
 rnorm_rvec (dnorm_rvec), 27
 rnorm_rvec(), 58
 rpois(), 29
 rpois_rvec (dpois_rvec), 28
 rt(), 41–44
 rt_rvec (dt_rvec), 41
 runif(), 43
 runif_rvec (dunif_rvec), 42
 rvec, 5, 7, 10, 11, 13–15, 17, 18, 20, 21,
 24–26, 28–30, 32, 33, 35, 36, 38, 40,
 41, 43–45, 48, 49, 53, 57, 62
 rvec(), 3, 6–8, 47, 51, 52, 56, 59, 60
 rvec-package, 3
 rvec_chr (rvec), 57
 rvec_chr(), 3, 51, 52
 rvec_dbl (rvec), 57
 rvec_dbl(), 3, 51, 52
 rvec_int (rvec), 57
 rvec_int(), 3, 51, 52, 56
 rvec_lgl, 46, 55
 rvec_lgl (rvec), 57
 rvec_lgl(), 3, 51, 52
 rvecs, 5, 48, 61

`rweibull()`, 44
`rweibull_rvec (dweibull_rvec)`, 43

`sapply()`, 48
`sd`, 59
`sd()`, 60, 62
`stats::dgamma()`, 18
`stats::dhyper()`, 21
`stats::dmultinom()`, 24
`stats::rhyper()`, 21
`stats::sd()`, 59
`stats::var()`, 60

`tibble`, 32, 38
`tidyselect`, 7
`tidyverse`, 10, 11, 13–15, 17, 18, 20, 21, 23,
26, 28, 29, 41, 43, 44

`vapply()`, 48
`var`, 60
`var()`, 59, 62
`vctrs::vec_c()`, 33
`vctrs::vec_detect_complete()`, 50
`vctrs::vec_detect_missing()`, 50
`vec_c()`, 48

`weighted.mean()`, 62
`weighted_mad (weighted_mean)`, 61
`weighted_mad()`, 4
`weighted_mean`, 61
`weighted_mean()`, 4
`weighted_median (weighted_mean)`, 61
`weighted_median()`, 4
`weighted_sd (weighted_mean)`, 61
`weighted_sd()`, 5
`weighted_var (weighted_mean)`, 61
`weighted_var()`, 5