

# Package: rvec (via r-universe)

October 27, 2024

**Type** Package

**Title** Vector Representing a Random Variable

**Version** 0.0.7

**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 working with output from a simulation or a Bayesian analysis.

**License** MIT + file LICENSE

**Encoding** UTF-8

**LazyData** true

**Roxygen** list(markdown = TRUE)

**RoxygenNote** 7.3.2

**Depends** R (>= 4.3.0)

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

**Suggests** bookdown, covr, dplyr, ggdist, ggplot2, knitr, posterior, rmarkdown, testthat (>= 3.0.0), tidyr, vdiff

**Config/testthat/edition** 3

**VignetteBuilder** knitr

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

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

**Repository** <https://bayesiandemography.r-universe.dev>

**RemoteUrl** <https://github.com/bayesiandemography/rvec>

**RemoteRef** HEAD

**RemoteSha** 3287860a531a62e16883a3197be5fc1098e1b46b

## 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	18
dgeom_rvec	20
dhyper_rvec	21
divorce	23
dlnorm_rvec	24
dmultinom_rvec	25
dnbinom_rvec	26
dnorm_rvec	28
dpois_rvec	30
draws_all	31
draws_ci	32
draws_fun	34
draws_median	35
draws_min	37
draws_quantile	38
dt_rvec	40
dunif_rvec	41
dweibull_rvec	43
extract_draw	44
if_else_rvec	45
is_rvec	46
map_rvec	47
matrixOps.rvec	48
missing	48
new_rvec	50
n_draw	51
rank	52
reg_post	53
rvec	53
sd	55
var	56
weighted_mean	57

## Index

60

---

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
- `collapse_to_rvec()` Data in data frame
- `new_rvec()` Blanks

### Manipulating rvecs

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

### 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
- `draws_any()` Any draws
- `draws_min()` Minimum draw
- `draws_max()` Maximum draw
- `draws_median()` Median draw
- `draws_mean()` Mean draw
- `draws_mode()` Modal draw
- `draws_ci()` Credible intervals
- `draws_quantile()` Quantiles
- `draws_fun()` Arbitrary function
- `n_draw()` Number of draws

### 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](mailto: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()` to construct an rvec.
- `expand_from_rvec()` to convert a data frame from using rvecs to using draw and value columns.
- `as_rvar???`
- converting rvecs to
- Functions for summarising and plotting distributions in package `ggdist` understand list columns.

**Examples**

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

---

collapse\_to\_rvec      *Convert a Data Frame Between 'Database' and 'Rvec' Formats*

---

**Description**

`collapse_to_rvec()` converts a data frame from a 'database' format to an 'rvec' format. `expand_from_rvec()`, does the opposite, converting a data frame from an rvecs format to a database 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 <b>grouped</b> .
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 database 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()` to construct a single rvec.
- `as_list_col()` to convert an rvec to a list variable.
- `dplyr::group_vars()` gives the names of the grouping variables in a grouped data frame.

`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
)

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

## rvec format to database 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

*The 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 <a href="#">stats::dbeta()</a> . Can be an 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 `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\(\)](#)
- [stats::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

*The 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**

x	Quantiles. Can be an rvec.
size	Number of trials. See <a href="#">stats::dbinom()</a> . Can be an rvec.
prob	Probability of success in each trial. See <a href="#">stats::dbinom()</a> . 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 `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.

**See Also**

- [dbinom\(\)](#)
- [pbinom\(\)](#)
- [qbinom\(\)](#)
- [rbinom\(\)](#)
- [stats::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

*The 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 <code>stats::dcauchy()</code> . Can be an rvec.
scale	Scale parameter. Default is 1. See <code>stats::dcauchy()</code> . 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\(\)](#)
- [stats::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)
```

**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**

x	Quantiles. Can be an rvec.
df	Degrees of freedom. See <a href="#">stats::dchisq()</a> . 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 `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\(\)](#)
- [stats::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

*The 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 <code>stats::dexp()</code> . 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\(\)](#)
- [stats::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)
```

## 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 <a href="#">stats::df()</a> . 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()`, `qf_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()`, `qf_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\(\)](#)
- [stats::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

*The 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 <code>stats::dgamma()</code> . Can be an rvec.
rate	Rate parameter. See <code>stats::dgamma()</code> . Can be an rvec.
scale	Scale parameter. An alternative to rate. See <code>stats::dgamma()</code> . 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()`, `qgamma_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()`, `qgamma_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\(\)](#)
- [stats::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)
```

**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 <code>stats::dgeom()</code> . 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.

**See Also**

- [dgeom\(\)](#)
- [pgeom\(\)](#)
- [qgeom\(\)](#)
- [rgeom\(\)](#)
- [stats::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

*The 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 <a href="#">stats::dhyper()</a> . Can be an rvec.
n	Number of black balls in the urn. See <a href="#">stats::rhyper()</a> . Can be an rvec.
k	Number of balls drawn from urn. See <a href="#">stats::dhyper()</a> . 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 <a href="#">stats::rhyper()</a> . 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()`, `phyper_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()`, `phyper_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.

## See Also

- [dhyper\(\)](#)
- [phyper\(\)](#)
- [qhyper\(\)](#)
- [rhyper\(\)](#)
- [stats::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

*The 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 <a href="#">stats::dlnorm()</a> . Can be an rvec.
sdlog	Standard deviation of distribution, on log scale. Default is 1. See <a href="#">stats::dlnorm()</a> . 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\(\)](#)
- [stats::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

*The 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 <a href="#">stats::dmultinom()</a> . Can be an rvec.
<code>prob</code>	Numeric non-negative vector, giving the probability of each outcome. Internally normalized to sum to 1. See <a href="#">stats::dmultinom()</a> . 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.
  - If `n` is greater than 1, a list of rvecs or ordinary R vectors

**See Also**

- [dmultinom\(\)](#)
- [rmultinom\(\)](#)
- [stats::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)
```

**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 <code>stats::dnbinom()</code> . Can be an rvec.
prob	Probability of success in each trial. See <code>stats::dnbinom()</code> . Can be an rvec.
mu	Mean value. See <code>stats::dnbinom()</code> . 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.

**See Also**

- [dnbinom\(\)](#)
- [pnbinom\(\)](#)
- [qnbinom\(\)](#)
- [rnbinom\(\)](#)
- [stats::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

*The 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 <a href="#">stats::dnorm()</a> . Can be an rvec.
sd	Standard deviation. Default is 1. See <a href="#">stats::dnorm()</a> . 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()`, `pnorm_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()`, `pnorm_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\(\)](#)
- [stats::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)
```

**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 <a href="#">stats::rpois()</a> . 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.

**See Also**

- [dpois\(\)](#)
- [ppois\(\)](#)
- [qpois\(\)](#)
- [rpois\(\)](#)
- [stats::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_ci()`
- `draws_quantile()`

Apply arbitrary function across draws:

- `draws_fun()`

For additional functions for summarising random draws, see `tidybayes` and `ggdist`. Function `as_list_col()` converts `rvecs` into a format that `tidybayes` and `ggdist` can work with.

**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 <a href="#">rvec</a> .
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\\_quantile\(\)](#)

Apply arbitrary function across draws:

- [draws\\_fun\(\)](#)

For additional functions for summarising random draws, see [tidybayes](#) and [ggdist](#). Function [as\\_list\\_col\(\)](#) converts rvecs into a format that [tidybayes](#) and [ggdist](#) can work with.

**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\\_ci\(\)](#)
- [draws\\_min\(\)](#)
- [draws\\_max\(\)](#)
- [draws\\_median\(\)](#)
- [draws\\_mean\(\)](#)
- [draws\\_mode\(\)](#)
- [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**

Summarise the distribution of random draws in an rvec, using means, medians, or modes.

**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 <a href="#">rvec</a> .
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\\_ci\(\)](#)
- [draws\\_quantile\(\)](#)

Apply arbitrary function across draws:

- [draws\\_fun\(\)](#)

For additional functions for summarising random draws, see [tidybayes](#) and [ggdist](#). Function [as\\_list\\_col\(\)](#) converts rvecs into a format that [tidybayes](#) and [ggdist](#) can work with.

### 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 <code>rvec</code> .
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\\_ci\(\)](#)
- [draws\\_quantile\(\)](#)

Apply arbitrary function across draws:

- [draws\\_fun\(\)](#)

For additional functions for summarising random draws, see [tidybayes](#) and [ggdist](#). Function [as\\_list\\_col\(\)](#) converts rvecs into a format that tidybayes and ggdist can work with.

### 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	<i>Quantiles Across Random Draws</i>
----------------	--------------------------------------

---

### 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 <a href="#">rvec</a> .
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\\_ci\(\)](#)
- [draws\\_min\(\)](#)
- [draws\\_max\(\)](#)
- [draws\\_median\(\)](#)
- [draws\\_mean\(\)](#)
- [draws\\_mode\(\)](#)

Apply arbitrary function across draws:

- [draws\\_fun\(\)](#)

For additional functions for summarising random draws, see [tidybayes](#) and [ggdist](#). Function [as\\_list\\_col\(\)](#) converts rvecs into a format that [tidybayes](#) and [ggdist](#) can work with.

**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))
```

---

dt\_rvec

*Student t Distribution, Using Multiple Draws*


---

## 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 <a href="#">stats::dt()</a> . Can be an rvec.
ncp	Non-centrality parameter. Default is 0. See <a href="#">stats::dt()</a> . 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\(\)](#)
- [stats::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 <code>stats::dunif()</code> . Can be an rvec.
max	Upper limited. Default is 1. See <code>stats::dunif()</code> . 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()`, `qunif_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()`, `qunif_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\(\)](#)
- [stats::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

x	Quantiles. Can be an rvec.
shape	Shape parameter. See <code>stats::dweibull()</code> . Can be an rvec.
scale	Scale parameter. See <code>stats::dweibull()</code> Default is 1. 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 `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\(\)](#)
- [stats::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

*Extract a Single Draw From an Rvec*

---

**Description**

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

**Usage**

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

**Arguments**

x                   An [rvec](#).

i                   Index for the draw to be extracted. A number between 1 and n\_draw(x). 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

*Is an Object an Rvec*

---

### 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`*Apply a Function and Put Results in an Rvec*

---

## 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

<code>.x</code>	A vector.
<code>.f</code>	A function.
<code>...</code>	Additional arguments passed to <code>.f</code> .

## 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 `vctrs::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	<i>Matrix Multiplication with Rvecs</i>
----------------	-----------------------------------------

---

**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	<i>Missing, Finite, and Infinite Values in Rvecs</i>
---------	------------------------------------------------------

---

**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 <i>rvec</i> .
<code>recursive</code>	Whether <code>anyNA()</code> should be applied recursively to lists. Ignored when <code>x</code> is an <i>rvec</i> .
<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()` for modifying individual values within draws.
- Base R functions `is.na()`, `is.nan()`, `is.finite()`, `is.infinite()`, `anyNA()`, `na.omit()`, `na.exclude()`
- `vctrs::vec_detect_missing()` to test whether all draws for an observation are missing.
- `vctrs::vec_detect_complete()` to test whether any draws for an observation are missing.
- `draws_any()`, `draws_all()` to 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

*Create a Blank Rvec*


---

**Description**

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

**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**

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**

- `rvec()` `rvec_chr()`, `rvec_dbl()`, `rvec_int()`, `rvec_lgl()` Create an rvec from data.
- `n_draw()` Query number of draws.

**Examples**

```
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)
```

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**

x	An ordinary vector or an <a href="#">rvec()</a> .
na.last	Treatment of NAs. Options are TRUE, FALSE, or "keep". See <a href="#">base::rank()</a> for details.
ties.method	Treatment of ties. See <a href="#">base::rank()</a> 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 *Standard Deviation, Including Rvecs*

---

### 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

`x` A numeric vector or R object, including an `rvec()`.

`na.rm` 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

x	Quantity being summarised. An ordinary vector or an <a href="#">rvec</a> .
wt	Weights. An ordinary vector, an <a href="#">rvec</a> , or NULL (the default.) If NULL, an unweighted summary is returned.
na_rm	Whether to remove NAs in x or wt before calculating. Default is FALSE. See <a href="#">matrixStats::weightedMean()</a> 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, 23  
  reg\_post, 53

anyNA(), 49  
anyNA.rvec (missing), 48  
as.matrix(), 46  
as\_list\_col, 5  
as\_list\_col(), 4, 8, 32, 33, 36, 38, 39, 46

base::rank(), 52

c(), 47  
character(), 54  
collapse\_to\_rvec, 6  
collapse\_to\_rvec(), 3, 4, 54

dbeta(), 10  
dbeta\_rvec, 9  
dbeta\_rvec(), 3  
dbinom(), 11  
dbinom\_rvec, 10  
dbinom\_rvec(), 3  
dcauchy(), 13  
dcauchy\_rvec, 12  
dcauchy\_rvec(), 3  
dchisq(), 14  
dchisq\_rvec, 13  
dchisq\_rvec(), 3  
dexp(), 16  
dexp\_rvec, 15  
dexp\_rvec(), 3  
df(), 17  
df\_rvec, 16  
df\_rvec(), 3  
dgamma(), 19  
dgamma\_rvec, 18  
dgamma\_rvec(), 3  
dgeom(), 20, 21  
dgeom\_rvec, 20  
dgeom\_rvec(), 3  
dhyper(), 22  
dhyper\_rvec, 21  
dhyper\_rvec(), 4  
divorce, 23  
divorce(), 4  
dlnorm(), 24, 25  
dlnorm\_rvec, 24  
dlnorm\_rvec(), 4  
dmultinom(), 4, 26  
dmultinom\_rvec, 25  
dnbinom(), 27, 28  
dnbinom\_rvec, 26  
dnbinom\_rvec(), 4  
dnorm(), 29  
dnorm\_rvec, 28  
dnorm\_rvec(), 4  
double(), 54  
dpois(), 30, 31  
dpois\_rvec, 30  
dpois\_rvec(), 4  
draws, 45  
draws\_\*, 49  
draws\_all, 31  
draws\_all(), 4, 33, 35–37, 39, 49  
draws\_any, 33  
draws\_any (draws\_all), 31  
draws\_any(), 4, 35–37, 39, 49  
draws\_ci, 32  
draws\_ci(), 3, 4, 32, 35, 36, 38, 39  
draws\_fun, 34  
draws\_fun(), 4, 32, 33, 36, 38, 39, 58  
draws\_max (draws\_min), 37  
draws\_max(), 4, 32, 33, 35, 36, 39  
draws\_mean, 58  
draws\_mean (draws\_median), 35  
draws\_mean(), 4, 32, 33, 35, 37, 39  
draws\_median, 35  
draws\_median(), 4, 32, 33, 35, 37, 39, 58

- draws\_min, 37
- draws\_min(), 4, 32, 33, 35, 36, 39
- draws\_mode (draws\_median), 35
- draws\_mode(), 4, 32, 33, 35, 38, 39
- draws\_quantile, 38
- draws\_quantile(), 4, 32, 33, 35, 36, 38, 58
- dt(), 40–43
- dt\_rvec, 40
- dt\_rvec(), 4
- dunif(), 42
- dunif\_rvec, 41
- dunif\_rvec(), 4
- dweibull(), 44
- dweibull\_rvec, 43
- dweibull\_rvec(), 4
- expand\_from\_rvec (collapse\_to\_rvec), 6
- expand\_from\_rvec(), 4, 6
- extract\_draw, 44
- extract\_draw(), 3
- if\_else\_rvec, 45
- if\_else\_rvec(), 3, 49
- ifelse(), 45
- integer(), 54
- is.finite(), 49
- is.infinite(), 49
- is.na(), 49
- is.na.rvec (missing), 48
- is.nan(), 49
- is\_rvec, 46
- is\_rvec(), 4, 52
- logical(), 54
- mad(), 58
- map\_rvec, 47
- map\_rvec(), 3
- matrixOps.rvec, 48
- matrixStats::colWeightedMeans(), 58
- matrixStats::weightedMean(), 58
- mean(), 58
- median(), 58
- missing, 48
- n\_draw, 51
- n\_draw(), 4, 7, 45, 51
- na.exclude(), 49
- na.exclude.rvec (missing), 48
- na.omit(), 49
- na.omit.rvec (missing), 48
- new\_rvec, 50
- new\_rvec(), 3, 54
- 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(), 16
- pexp\_rvec (dexp\_rvec), 15
- pf(), 17
- pf\_rvec (df\_rvec), 16
- pgamma(), 19
- pgamma\_rvec (dgamma\_rvec), 18
- pgeom(), 20, 21
- pgeom\_rvec (dgeom\_rvec), 20
- phyper(), 22
- phyper\_rvec (dhyper\_rvec), 21
- plnorm(), 24, 25
- plnorm\_rvec (dlnorm\_rvec), 24
- pnbinom(), 27, 28
- pnbinom\_rvec (dnbinom\_rvec), 26
- pnorm(), 29
- pnorm\_rvec (dnorm\_rvec), 28
- ppois(), 30, 31
- ppois\_rvec (dpois\_rvec), 30
- pt(), 40–43
- pt\_rvec (dt\_rvec), 40
- punif(), 42
- punif\_rvec (dunif\_rvec), 41
- 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(), 16
- qexp\_rvec (dexp\_rvec), 15
- qf(), 17

- qf\_rvec (df\_rvec), 16
- qgamma(), 19
- qgamma\_rvec (dgamma\_rvec), 18
- qgeom(), 20, 21
- qgeom\_rvec (dgeom\_rvec), 20
- qhyper(), 22
- qhyper\_rvec (dhyper\_rvec), 21
- qlnorm(), 24, 25
- qlnorm\_rvec (dlnorm\_rvec), 24
- qnbinom(), 27, 28
- qnbinom\_rvec (dnbinom\_rvec), 26
- qnorm(), 29
- qnorm\_rvec (dnorm\_rvec), 28
- qpois(), 30, 31
- qpois\_rvec (dpois\_rvec), 30
- qt(), 40–43
- qt\_rvec (dt\_rvec), 40
- qunif(), 42
- qunif\_rvec (dunif\_rvec), 41
- qweibull(), 44
- qweibull\_rvec (dweibull\_rvec), 43
  
- rank, 52
- rbeta(), 10
- rbeta\_rvec (dbeta\_rvec), 9
- rbinom(), 11
- rbinom\_rvec (dbinom\_rvec), 10
- rbinom\_rvec(), 54
- rcauchy(), 13
- rcauchy\_rvec (dcauchy\_rvec), 12
- rchisq(), 14
- rchisq\_rvec (dchisq\_rvec), 13
- reg\_post, 53
- reg\_post(), 4
- rexp(), 16
- rexp\_rvec (dexp\_rvec), 15
- rf(), 17
- rf\_rvec (df\_rvec), 16
- rgamma(), 19
- rgamma\_rvec (dgamma\_rvec), 18
- rgeom(), 20, 21
- rgeom\_rvec (dgeom\_rvec), 20
- rhyper(), 22
- rhyper\_rvec (dhyper\_rvec), 21
- rlnorm(), 24, 25
- rlnorm\_rvec (dlnorm\_rvec), 24
- rmultinom(), 26
- rmultinom\_rvec (dmultinom\_rvec), 25
- rnbinom(), 27, 28
- rnbinom\_rvec (dnbinom\_rvec), 26
- rnorm(), 29
- rnorm\_rvec (dnorm\_rvec), 28
- rnorm\_rvec(), 54
- rpois(), 30, 31
- rpois\_rvec (dpois\_rvec), 30
- rt(), 40–43
- rt\_rvec (dt\_rvec), 40
- runif(), 42
- runif\_rvec (dunif\_rvec), 41
- rvec, 5, 7, 10, 11, 13, 14, 16, 17, 19, 21, 22, 25–27, 29, 31–34, 36–38, 41, 42, 44, 47, 49, 51, 53, 58
- rvec(), 3, 6–8, 46, 51, 52, 55, 56
- rvec-package, 3
- rvec\_chr (rvec), 53
- rvec\_chr(), 3, 51
- rvec\_dbl (rvec), 53
- rvec\_dbl(), 3, 51
- rvec\_int (rvec), 53
- rvec\_int(), 3, 51, 52
- rvec\_lgl, 45
- rvec\_lgl (rvec), 53
- rvec\_lgl(), 3, 51
- rvecs, 5, 48, 57
- rweibull(), 44
- rweibull\_rvec (dweibull\_rvec), 43
  
- sapply(), 47
- sd, 55
- sd(), 56, 58
- stats::dbeta(), 9
- stats::dbinom(), 11
- stats::dcauchy(), 12
- stats::dchisq(), 14
- stats::dexp(), 15
- stats::df(), 17
- stats::dgamma(), 18
- stats::dgeom(), 20
- stats::dhyper(), 22
- stats::distributions, 10, 11, 13, 14, 16, 17, 19, 21, 22, 25, 26, 28, 29, 31, 41, 42, 44
- stats::dlnorm(), 24
- stats::dmultinom(), 25
- stats::dnbinom(), 27
- stats::dnorm(), 28
- stats::dt(), 40
- stats::dunif(), 42

`stats::dweibull()`, 43  
`stats::rhyper()`, 22  
`stats::rpois()`, 30  
`stats::sd()`, 55  
`stats::var()`, 56

`tibble`, 33, 39  
`tidyselect`, 7  
`tidyverse`, 10, 11, 13, 14, 16, 17, 19, 20, 22,  
24, 27, 29, 30, 40, 42, 43

`vapply()`, 47  
`var`, 56  
`var()`, 55, 58  
`vctrs::vec_c()`, 34, 47  
`vctrs::vec_detect_complete()`, 49  
`vctrs::vec_detect_missing()`, 49

`weighted.mean()`, 58  
`weighted_mad (weighted_mean)`, 57  
`weighted_mad()`, 4  
`weighted_mean`, 57  
`weighted_mean()`, 4  
`weighted_median (weighted_mean)`, 57  
`weighted_median()`, 4  
`weighted_sd (weighted_mean)`, 57  
`weighted_sd()`, 4  
`weighted_var (weighted_mean)`, 57  
`weighted_var()`, 4