Package ‘rvec’

November 8, 2023

Type Package

Title Vector Representing a Random Variable

Version 0.0.6

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

RoxygenNote 7.2.3

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, vdiffr

Config/testthat/edition 3

VignetteBuilder knitr


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 2023-11-08 19:30:05 UTC
R topics documented:

- 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_quantile ........................................ 37
- dt_rvec ................................................. 38
- duniq_rvec ............................................. 40
- dweibull_rvec ........................................ 41
- if_else_rvec .......................................... 43
- is_rvec .................................................. 44
- map_rvec ................................................ 45
- matrixOps.rvec ....................................... 46
- missing .................................................. 46
- n_draw ................................................... 48
- rank ...................................................... 49
- reg_post ............................................... 50
- rvec ..................................................... 51
- sd ....................................................... 52
- var ....................................................... 53
- weighted_mean ........................................ 54

Index 57
Description

Tools for working with random draws from a distribution, e.g., 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

Manipulating rvecs

- `if_else_rvec()`
- `map_rvec()`

Probability distributions

- `dbeta_rvec()`
- `dbinom_rvec()`
- `dcauchy_rvec()`
- `dchisq_rvec()`
- `dexp_rvec()`
- `df_rvec()`
- `dgamma_rvec()`
- `dgeom_rvec()`
- `dhyper_rvec()`
- `dlnorm_rvec()`
• dmultinom()
• dnorm_rvec()
• dpois_rvec()
• dt_rvec()
• dunif_rvec()
• dweibull_rvec()

Summarizing across draws
• draws_all()
• draws_any()
• draws_median()
• draws_mean()
• draws_mode()
• draws_ci()
• draws_quantile()
• draws_fun()
• n_draw()

Coercion, classes
• as_list_col()
• expand_from_rvec()
• is_rvec()

Weighted summaries
• weighted_mad()
• weighted_mean()
• weighted_median()
• weighted_sd()
• weighted_var()

Datasets
• divorce()
• reg_post()

Packages with similar functionality
• rv
• posterior
as_list_col

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

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 rvec 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

```r
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

```r
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 `expand_from_rvec()`.
- **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** `<tidyselect>` Variables used to stratify or cross-classify the data. 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_db1, 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_db1
- "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**

```r
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 `stats::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()`
- `stats::distributions`

Examples

```r
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 `stats::dbinom()`. Can be an rvec.
- **prob**: Probability of success in each trial. See `stats::dbinom()`. 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

```r
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)
```

The Cauchy Distribution, Using Multiple Draws

Description

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

Usage

```r
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 `stats::dcauchy()`. Can be an rvec.
- `scale` Scale parameter. Default is 1. See `stats::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()`, `pcauchy_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()`, `pcauchy_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

```r
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

**The 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**

- **x**  
  Quantiles. Can be an rvec.
- **df**  
  Degrees of freedom. See `stats::dchisq()`. 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.

- 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

```r
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

```r
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 `stats::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()`
- `stats::distributions`.

Examples

```r
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 stats::df(). Can be rvecs.
cp 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()
- 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 stats::dgamma(). Can be an rvec.
rate
  Rate parameter. See stats::dgamma(). Can be an rvec.
scale
  Scale parameter. An alternative to rate. See stats::dgamma(). Can be an rvec.
\texttt{dgamma\_rvec}\hspace{1cm}19

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>log, (\text{log.p})</td>
<td>Whether to return results on a log scale. Default is (\text{FALSE}). Cannot be an (\text{rvec}).</td>
</tr>
<tr>
<td>q</td>
<td>Quantiles. Can be an (\text{rvec}).</td>
</tr>
<tr>
<td>lower.tail</td>
<td>Whether to return (P[X \leq x]), as opposed to (P[X &gt; x]). Default is (\text{TRUE}). Cannot be an (\text{rvec}).</td>
</tr>
<tr>
<td>p</td>
<td>Probabilities. Can be an (\text{rvec}).</td>
</tr>
<tr>
<td>n</td>
<td>The length of random vector being created. Cannot be an (\text{rvec}).</td>
</tr>
<tr>
<td>n_draw</td>
<td>Number of random draws in the random vector being created. Cannot be an (\text{rvec}).</td>
</tr>
</tbody>
</table>

**Details**

Functions \texttt{dgamma\_rvec()}, \texttt{pgamma\_rvec()}, \texttt{qgamma\_rvec()} and \texttt{rgamma\_rvec()} work like base \(\text{R}\) functions \texttt{dgamma(), pgamma(), qgamma(),} and \texttt{rgamma()}, except that they accept \(\text{rvec}\)s as inputs. If any input is an \(\text{rvec}\), then the output will be too. Function \texttt{rgamma\_rvec()} also returns an \(\text{rvec}\) if a value for \(\text{n\_draw}\) is supplied.

\texttt{dgamma\_rvec()}, \texttt{pgamma\_rvec()}, \texttt{qgamma\_rvec()} and \texttt{rgamma\_rvec()} use \texttt{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 \(\text{rvec}\)s, or if a value for \(\text{n\_draw}\) is supplied, then an \(\text{rvec}\)
- Otherwise an ordinary \(\text{R}\) vector.

**See Also**

- \texttt{dgamma()}  
- \texttt{pgamma()}  
- \texttt{qgamma()}  
- \texttt{rgamma()}  
- \texttt{stats::distributions}.

**Examples**

```r
x <- \text{rvec}((list(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)
```
The 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 `stats::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()`, `pgeom_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()`, `pgeom_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

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

```r
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()`, `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

```r
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)
```

```r
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**

```r
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.
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 `stats::dlnorm()`. Can be an rvec.
- `sdlog` Standard deviation of distribution, on log scale. Default is 1. See `stats::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()`, `plnorm_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()`, `plnorm_rvec()` and `rlnorm_rvec()` use tidyverse vector recycling rules:
- Vectors of length 1 are recycled
- All other vectors must have the same size
dmultinom_rvec

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

- x Quantiles. Can be an rvec.
- size Total number of trials. See stats::dmultinom(). Can be an rvec.
- prob Numeric non-negative vector, giving the probability of each outcome. Internally normalized to sum to 1. See stats::dmultinom(). Can be an rvec.
- log Whether to return log(p) rather than p. Default is FALSE. Cannot 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 \texttt{dmultinom\_rvec()} and \texttt{rmultinom\_rvec()} work like base R functions \texttt{dmultinom()} and \texttt{rmultinom()}, except that they accept \texttt{rvec}s as inputs. If any input is an \texttt{rvec}, then the output will be too. Function \texttt{rmultinom\_rvec()} also returns an \texttt{rvec} if a value for \texttt{n\_draw} is supplied.

Like the base R functions \texttt{dmultinom()} and \texttt{rmultinom()}, \texttt{dmultinom\_rvec()} and \texttt{rmultinom\_rvec()} do not recycle their arguments.

Value

- \texttt{dmultinom()}
  - If any of the arguments are \texttt{rvec}s, or if a value for \texttt{n\_draw} is supplied, then an \texttt{rvec}
  - Otherwise an ordinary R vector.
- \texttt{rmultinom()}
  - If \texttt{n} is 1, an \texttt{rvec} or ordinary R vector.
  - If \texttt{n} is greater than 1, a list of \texttt{rvec}s or ordinary R vectors

See Also

- \texttt{dmultinom()}
- \texttt{rmultinom()}
- \texttt{stats::distributions}.

Examples

\begin{verbatim}
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)
\end{verbatim}

\medskip

\textbf{The Negative Binomial Distribution, Using Multiple Draws}

Description

Density, distribution function, quantile function and random generation for the negative binomial distribution, modified to work with \texttt{rvec}s.
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 stats::dnbinom(). Can be an rvec.
prob  Probability of success in each trial. See stats::dnbinom(). Can be an rvec.
mu  Mean value. See stats::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(), pnbinom_rvec() and rnbinom_rvec() work like base R functions dnbinom(), pnbinom(), qbinom(), 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(), pnbinom_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.
dnorm_rvec

See Also
- `dnbinom()`
- `pnbinom()`
- `qnbinom()`
- `rnbinom()`
- `stats::distributions`

Examples

```r
x <- rvec(list(c(3, 5),
               c(0, 2)))
dbinom_rvec(x, size = 6, prob = 0.2)
pbinom_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

```r
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 `stats::dnorm()`. Can be an rvec.
- `sd`  
  Standard deviation. Default is 1. See `stats::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.

\( \text{n\_draw} \) Number of random draws in the random vector being created. Cannot be an rvec.

Details

Functions \text{dnorm\_rvec()}, \text{pnorm\_rvec()}, \text{qnorm\_rvec()} and \text{rnorm\_rvec()} work like base R functions \text{dnorm()}, \text{pnorm()}, \text{qnorm()}, and \text{rnorm()}, except that they accept rvecs as inputs. If any input is an rvec, then the output will be too. Function \text{rnorm\_rvec()} also returns an rvec if a value for \text{n\_draw} is supplied.

\text{dnorm\_rvec()}, \text{pnorm\_rvec()}, \text{qnorm\_rvec()} and \text{rnorm\_rvec()} use \text{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 \text{n\_draw} is supplied, then an rvec
- Otherwise an ordinary R vector.

See Also

- \text{dnorm()}
- \text{pnorm()}
- \text{qnorm()}
- \text{rnorm()}
- \text{stats::distributions.}

Examples

\begin{verbatim}
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)
\end{verbatim}
The Poisson Distribution, Using Multiple Draws

dpois_rvec

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)
pois_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 stats::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.

See Also

- dpois()
- ppois()
- qpois()
- rpois()
- stats::distributions.

Examples

```r
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**

```r
draws_all(x, na_rm = FALSE)
```

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

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

```r
draws_any(x, na_rm = FALSE)
```

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

```r
## 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_median()`
- `draws_mean()`
- `draws_mode()`
- `draws_ci()`
- `draws_quantile()`

Apply arbitrary function across draws:

- `draws_fun()` to apply arbitrary functions

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

```r
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 a simple credible interval.
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 `rvec`.
- **width**: A number, where \(0 < width \leq 1\). 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

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

However, creating columns in this way can corrupt 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_median()`
  - `draws_mean()`
  - `draws_mode()`
  - `draws_quantile()`

Apply arbitrary function across draws:

- `draws_fun()` to apply arbitrary functions

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

```r
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)
```

## 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**
```r
draws_fun(x, fun, ...)
```

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

**Arguments**
- `x`  
  An object of class `rvec`.
- `fun`  
  A function.
- `...`  
  Additional arguments passed to `fun`.

**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_median()
- draws_mean()
- draws_mode()
- draws_quantile()

Examples

```r
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'
draws_median(x, na_rm = FALSE)

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_ci()`
- `draws_quantile()`

Apply arbitrary function across draws:

- `draws_fun()` to apply arbitrary functions

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

```r
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_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()
dt_rvec

- `draws_ci()`
- `draws_median()`
- `draws_mean()`
- `draws_mode()`

Apply arbitrary function across draws:
- `draws_fun()` to apply arbitrary functions

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

```r
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**

```r
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 `stats::dt()`. Can be an rvec.
- `ncp`: Non-centrality parameter. Default is 0. See `stats::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()`
- `stats::distributions.`
Examples

```r
dx <- 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

```r
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 `stats::dunif()`. Can be an rvec.
- **max**: Upper limited. Default is 1. See `stats::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()`, `punif_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()`, `punif_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

```r
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)
```

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 stats::dweibull(). Can be an rvec.

scale
Scale parameter. See stats::dweibull() 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(), pweibull_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(), pweibull_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

```r
x <- rvec(list(c(3.2, 4.5),
              c(7.6, 0.7)))
dweibull_rvec(x, shape = 2)
pweibull_rvec(x, shape = 2)

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

Description

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

Usage

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

Arguments

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

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

```r
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**

```r
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**

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

Apply 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

.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 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)
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

```r
## 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

```r
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
```

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

x, object          An rvec.
recursive          Whether anyNA() should be applied recursively to lists. Ignored when x is an
                   rvec.
...                 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

\[
x <- \text{rvec}(\text{list}(\text{c}(1.2, \text{NA}), \\
\text{c}(\text{Inf}, 3), \\
\text{c}(-1, \text{NaN})))
\]

## 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)

---

n_draw Query Number of Draws

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)
**rank**

**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**

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

---

**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**

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

**Arguments**

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

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

---

**reg_post**  
**Posterior Sample from Linear Regression**

Description

Posterior sample for parameters from a linear regression model.

Usage

```r
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

Create an Object of Class "rvec".

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, or an atomic vector.

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.
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", depending on typeof(x).

See Also

- `collapse_to_rvec()` creates rvecs within a data frame.
- Random variate functions such as `rnorm_rvec()` and `rbinom_rvec()` can be used to create rvecs representing probability distributions.

Examples

```r
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(l)
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

```r
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.
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

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

Description

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

Usage

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

Arguments

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

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

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

---

### 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**

```r
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 `rvec`.
- **wt**: Weights. An ordinary vector, an `rvec`, 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 `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

```r
## '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, 50

anyNA(), 47
anyNA.rvec (missing), 46
as.matrix(), 44
as_list_col, 5
as_list_col(), 4, 8, 32, 36, 38, 44

base::rank(), 49

c(), 45
character(), 51
collapse_to_rvec, 6
collapse_to_rvec(), 3, 52
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(), 3
divorce, 23
divorce(), 4
dlnorm(), 24, 25
dlnorm_rvec, 24
dlnorm_rvec(), 3
dmultinom(), 4, 26
dmultinom_rvec, 25
dmultinom_rvec(), 3
dnbins(), 27, 28
dnbins_rvec, 26
dnbins_rvec(), 4
dnorm(), 29
dnorm_rvec, 28
dnorm_rvec(), 4
double(), 51
dpois(), 30, 31
dpois_rvec, 30
dpois_rvec(), 4
draws, 43
draws_*, 47
draws_all, 31
draws_all(), 4, 33, 35–37, 47
draws_any, 33
draws_any (draws_all), 31
draws_any(), 4, 35–37, 47
draws_ci, 32
draws_ci(), 3, 4, 32, 35–38
draws_fun, 34
draws_fun(), 4, 32, 33, 36, 38, 55
draws_mean, 55
draws_mean (draws_median), 35
draws_mean(), 4, 32, 33, 35, 38
draws_median, 35
draws_median(), 4, 32, 33, 35, 38, 55
draws_mode (draws_median), 35
draws_mode() 4, 32, 33, 35, 38
draws_quantile, 37
draws_quantile(), 37, 33, 35, 36, 39, 41, 42
dt(), 39, 41, 42
dt_rvec, 38
dt_rvec(), 4
dunif(), 41
dunif_rvec, 40
dunif_rvec(), 4
dweibull(), 42
dweibull_rvec, 41
dweibull_rvec(), 4
expand_from_rvec(collapse_to_rvec), 6
expand_from_rvec(), 4, 6
if_else_rvec, 43
if_else_rvec(), 3, 47
ifelse(), 43
integer(), 51
is.finite(), 47
is.infinite(), 47
is.na(), 47
is.na.rvec(missing), 46
is.nan(), 47
is_rvec, 44
is_rvec(), 4, 49
logical(), 51
mad(), 55
map_rvec, 45
map_rvec(), 3
matrixOps.rvec, 46
matrixStats::colWeightedMeans, 55
matrixStats::weightedMean(), 55
mean(), 55
median(), 55
missing, 46
n_draw, 48
n_draw(), 4, 7
na.exclude(), 47
na.exclude.rvec(missing), 46
na.omit(), 47
na.omit.rvec(missing), 46
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(), 39, 41, 42
pt_rvec(dt_rvec), 38
punif(), 41
punif_rvec(dunif_rvec), 40
pweibull(), 42
pweibull_rvec(dweibull_rvec), 41
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
qinorm(), 24, 25
qinorm_rvec(dlnorm_rvec), 24
Index

qnbinom(), 27, 28
qnbinom_rvec(dbinom_rvec), 26
qnorm(), 29
qnorm_rvec(dnorm_rvec), 28
qpois(), 30, 31
qpois_rvec(dpois_rvec), 30
qt(), 39, 41, 42
qt_rvec(dt_rvec), 38
qunif(), 41
qunif_rvec(dunif_rvec), 40
qweibull(), 42
qweibull_rvec(dweibull_rvec), 41
rank, 49
rbeta(), 10
rbeta_rvec(dbeta_rvec), 9
rbinom(), 11
rbinom_rvec(dbinom_rvec), 10
rbinom_rvec(), 52
rcauchy(), 13
rcauchy_rvec(dcauchy_rvec), 12
rchisq(), 14
rchisq_rvec(dchisq_rvec), 13
reg_post, 50
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(dbinom_rvec), 26
rnorm(), 29
rnorm_rvec(dnorm_rvec), 28
rnorm_rvec(), 52
rpois(), 30, 31
rpois_rvec(dpois_rvec), 30
rt(), 39, 41, 42
rt_rvec(dt_rvec), 38
runif(), 41
runif_rvec(dunif_rvec), 40
rvec, 5, 7, 10, 11, 13, 14, 16, 17, 19, 21, 22,
25–27, 29, 31–34, 36, 37, 39, 41, 42,
45, 47, 49, 51, 55
rvec(), 3, 6–8, 44, 49, 52, 53
rvec-package, 3
rvec_chr(rvec), 51
rvec_chr(), 3
rvec_dbl(rvec), 51
rvec_dbl(), 3
rvec_int(rvec), 51
rvec_int(), 3, 50
rvec_lgl, 43
rvec_lgl(rvec), 51
rvec_lgl(), 3
rvecs, 5, 46, 54
rweibull(), 42
rweibull_rvec(dweibull_rvec), 41
sapply(), 45
sd, 52
sd(), 54, 55
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, 39,
41, 42
stats::dlnorm(), 24
stats::dmultinom(), 25
stats::dnbinom(), 27
stats::dnorm(), 28
stats::dt(), 39
stats::dunif(), 40
stats::dweibull(), 42
stats::rhyper(), 22
stats::rpois(), 30
stats::sd(), 53
stats::var(), 53
tibble, 33, 37
tidyselect, 7
tidyverse, 10, 11, 13, 14, 16, 17, 19, 20, 22, 24, 27, 29, 30, 39, 41, 42

vapply(), 45
var, 53
var(), 53, 55
vctrs::vec_c(), 34, 45
vctrs::vec_detect_complete(), 47
vctrs::vec_detect_missing(), 47

weighted.mean(), 55
weighted_mad (weighted_mean), 54
weighted_mad(), 4
weighted_mean, 54
weighted_mean(), 4
weighted_median (weighted_mean), 54
weighted_median(), 4
weighted_sd (weighted_mean), 54
weighted_sd(), 4
weighted_var (weighted_mean), 54
weighted_var(), 4