

Package ‘vws’

November 11, 2025

Type Package

Title Vertical Weighted Strips

Version 0.3.0

Maintainer Andrew M. Raim <andrew.raim@gmail.com>

Description A reference implementation of the Vertical Weighted Strips method explored by Raim, Livsey, and Irimata (2025) <[doi:10.48550/arXiv.2401.09696](https://doi.org/10.48550/arXiv.2401.09696)> for rejection sampling.

URL <https://github.com/andrewraim/vws>

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.3

Depends R (>= 4.1.0)

Imports Rcpp, fntI

LinkingTo Rcpp, fntI

Suggests knitr, rmarkdown, quarto, statmod, tidyverse

VignetteBuilder quarto

NeedsCompilation yes

Author Andrew M. Raim [aut, cre]

Repository CRAN

Date/Publication 2025-11-11 21:20:02 UTC

Contents

vws-package	2
Categorical	2
Gumbel	3
log_sum_exp	4
optimize_hybrid	5
Print	6
rect	7
seq_knots	8

vws-package	vws
-------------	-----

Description

Package documentation

Author(s)

Maintainer: Andrew M. Raim <andrew.raim@gmail.com>

See Also

Useful links:

- <https://github.com/andrewraim/vws>

Categorical	<i>Categorical Distribution</i>
-------------	---------------------------------

Description

Draw variates from a categorical distribution.

Usage

```
r_categ(n, p, log = FALSE, one_based = FALSE)
```

Arguments

n	Number of desired draws.
p	Vector of k probabilities for distribution.
log	logical; if TRUE, interpret p as being specified on the log-scale as $\log(p)$. Otherwise, interpret p as being specified on the original probability scale.
one_based	logical; if TRUE, assume a categorical distribution with support $\{1, \dots, k\}$. Otherwise, assume support $\{0, \dots, k - 1\}$.

Value

A vector of n draws.

Examples

```

p = c(0.1, 0.2, 0.3, 0.4)
lp = log(p)

set.seed(1234)
r_categ(50, p, log = FALSE, one_based = FALSE)
r_categ(50, p, log = FALSE, one_based = TRUE)

set.seed(1234)
r_categ(50, lp, log = TRUE, one_based = FALSE)
r_categ(50, lp, log = TRUE, one_based = TRUE)

```

Gumbel

*Gumbel Distribution***Description**

Functions for the Gumbel distribution with density

$$f(x \mid \mu, \sigma) = \frac{1}{\sigma} \exp\{-\{(x - \mu)/\sigma + e^{-(x - \mu)/\sigma}\}\}$$

Usage

```

r_gumbel(n, mu = 0, sigma = 1)

d_gumbel(x, mu = 0, sigma = 1, log = FALSE)

p_gumbel(q, mu = 0, sigma = 1, lower = TRUE, log = FALSE)

q_gumbel(p, mu = 0, sigma = 1, lower = TRUE, log = FALSE)

```

Arguments

n	Number of draws.
mu	Location parameter.
sigma	Scale parameter.
x	Vector; argument of density.
log	Logical; if TRUE, probabilities p are given as $\log(p)$.
q	Vector; argument of quantile function.
lower	Logical; if TRUE (default), probabilities are $P[X \leq x]$ otherwise, $P[X > x]$.
p	Vector; argument of cumulative distribution function.

Value

`d_gumbel` computes the density, `r_gumbel` generates random deviates, `p_gumbel` computes the CDF, and `q_gumbel` computes quantiles. A vector is returned by each.

Examples

```
mu = 1
sigma = 2
x = r_gumbel(100000, mu, sigma)
xx = seq(min(x), max(x), length.out = 100)

plot(density(x))
lines(xx, d_gumbel(xx, mu, sigma), lty = 2, col = "blue", lwd = 2)

plot(ecdf(x))
lines(xx, p_gumbel(xx, mu, sigma), lty = 2, col = "blue", lwd = 2)

pp = seq(0, 1, length.out = 102) |> head(-1) |> tail(-1)
qq = quantile(x, probs = pp)
plot(pp, qq)
lines(pp, q_gumbel(pp, mu, sigma), lty = 2, col = "blue", lwd = 2)
```

log_sum_exp

Log-Sum-Exp

Description

Compute arithmetic on the log-scale in a more stable way than directly taking logarithm and exponentiating.

Usage

```
log_sum_exp(x)

log_add2_exp(x, y)

log_sub2_exp(x, y)
```

Arguments

x	A numeric vector.
y	A numeric vector; it should have the same length as x.

Details

The function `log_sum_exp` computes $\log(\sum(\exp(x)))$ using the method in StackExchange post <https://stats.stackexchange.com/a/381937>.

The functions `log_add2_exp` and `log_sub2_exp` compute $\log(\exp(x) + \exp(y))$ and $\log(\exp(x) - \exp(y))$, respectively. The function `log_sub2_exp` expects that each element of `x` is larger than or equal to its corresponding element in `y`. Otherwise, NaN will be returned with a warning.

Value

log_add2_exp and log_sub2_exp return a vector of pointwise results whose i th element is the result based on $x[i]$ and $y[i]$. log_sum_exp returns a single scalar.

Examples

```
pi = 1:6 / sum(1:6)
x = log(2*pi)
log(sum(exp(x)))
log_sum_exp(x)

# Result should be 0
x = c(-Inf -Inf, 0)
log_sum_exp(x)

# Result should be -Inf
x = c(-Inf -Inf, -Inf)
log_sum_exp(x)

# Result should be Inf
x = c(-Inf -Inf, Inf)
log_sum_exp(x)

# Result should be 5 on the original scale
out = log_add2_exp(log(3), log(2))
exp(out)

# Result should be 7 on the original scale
out = log_sub2_exp(log(12), log(5))
exp(out)
```

optimize_hybrid

*Hybrid Univariate Optimization***Description**

Use Brent's method if a bounded search interval is specified. Otherwise use BFGS method.

Usage

```
optimize_hybrid(f, init, lower, upper, maximize = FALSE, maxiter = 10000L)
```

Arguments

<code>f</code>	Objective function. Should take a scalar as an argument.
<code>init</code>	Initial value for optimization variable.
<code>lower</code>	Lower bound for search; may be $-\infty$.

upper	Upper bound for search; may be $+\infty$.
maximize	logical; if TRUE, optimization will be a maximization. Otherwise, it will be a minimization.
maxiter	Maximum number of iterations.

Value

A list with the following elements.

par	Value of optimization variable.
value	Value of optimization function.
method	Description of result.
status	Status code from BFGS or 0 otherwise.

Examples

```
f = function(x) { x^2 }
optimize_hybrid(f, init = 0, lower = -1, upper = 2, maximize = FALSE)
optimize_hybrid(f, init = 0, lower = -1, upper = Inf, maximize = FALSE)
optimize_hybrid(f, init = 0, lower = -Inf, upper = 1, maximize = FALSE)
optimize_hybrid(f, init = 0, lower = 0, upper = Inf, maximize = FALSE)
optimize_hybrid(f, init = 0, lower = -Inf, upper = 0, maximize = FALSE)

f = function(x) { 1 - x^2 }
optimize_hybrid(f, init = 0, lower = -1, upper = 1, maximize = TRUE)
optimize_hybrid(f, init = 0, lower = -1, upper = 0, maximize = TRUE)
optimize_hybrid(f, init = 0, lower = 0, upper = 1, maximize = TRUE)
```

Print

Printing

Description

Functions to print messages using an sprintf syntax.

Usage

```
printf(fmt, ...)

logger(fmt, ..., dt_fmt = "%Y-%m-%d %H:%M:%S", join = " - ")

fprintf(file, fmt, ...)
```

Arguments

fmt	Format string which can be processed by <code>sprintf</code>
...	Additional arguments
dt_fmt	Format string which can be processed by <code>format.POSIXct</code>
join	A string to place between the timestamp and the message.
file	A connection, or a character string naming the file to print to

Value

None (invisible NULL); functions are called for side effects.

Examples

```
printf("Hello world %f %d\n", 0.1, 5)
logger("Hello world\n")
logger("Hello world %f %d\n", 0.1, 5)
logger("Hello world %f %d\n", 0.1, 5, dt_fmt = "%H:%M:%S")
logger("Hello world %f %d\n", 0.1, 5, join = " >> ")
logger("Hello world %f %d\n", 0.1, 5, join = " ")
```

rect	<i>Rectangular transformation</i>
------	-----------------------------------

Description

A transformation from unconstrained \mathbb{R}^d to a rectangle in \mathbb{R}^d , and its inverse transformation.

Usage

```
rect(z, a, b)

inv_rect(x, a, b)
```

Arguments

z	A point in the rectangle $[a_1, b_1] \times \dots \times [a_d, b_d]$.
a	A vector (a_1, \dots, a_d) , Elements may be $-\text{Inf}$.
b	A vector (b_1, \dots, b_d) , Elements may be $+\text{Inf}$.
x	A point in \mathbb{R}^d .

Value

A vector of length d .

Examples

```

n = 20
x = seq(-5, 5, length.out = n)

# Transform x to the interval [-1, 1]
a = rep(-1, n)
b = rep(+1, n)
z = inv_rect(x, a, b)
print(z)
xx = rect(z, a, b)
stopifnot(all(abs(x - xx) < 1e-8))

# Transform x to the interval [-Inf, 1]
a = rep(-Inf, n)
b = rep(+1, n)
z = inv_rect(x, a, b)
print(z)
xx = rect(z, a, b)
stopifnot(all(abs(x - xx) < 1e-8))

# Transform x to the interval [-1, Inf]
a = rep(-1, n)
b = rep(+Inf, n)
z = inv_rect(x, a, b)
print(z)
xx = rect(z, a, b)
stopifnot(all(abs(x - xx) < 1e-8))

# Transform x to the interval [-Inf, Inf]
a = rep(-Inf, n)
b = rep(+Inf, n)
z = inv_rect(x, a, b)
print(z)
xx = rect(z, a, b)
stopifnot(all(abs(x - xx) < 1e-8))

```

seq_knots

Produce a sequence of knots

Description

Produce knots which define N equally-spaced intervals between (finite) endpoints lo and hi.

Usage

```
seq_knots(lo, hi, N, endpoints = FALSE)
```


Arguments

lo	Left endpoint; must be finite.
hi	Right endpoint; must be finite.
N	Number of desired intervals.
endpoints	logical; if TRUE, include the endpoints.

Value

A vector that represents a sequence of knots. If `endpoints = TRUE`, it contains $N + 1$ evenly-spaced knots that represent N regions with endpoints included. If `endpoints = FALSE`, the endpoints are excluded.

Examples

```
seq_knots(0, 1, N = 5)
seq_knots(0, 1, N = 5, endpoints = TRUE)

# Trivial case: make endpoints for just one interval
seq_knots(0, 1, N = 1)
seq_knots(0, 1, N = 1, endpoints = TRUE)

# The following calls throw errors
tryCatch({
  seq_knots(0, 1, N = 0)
}, error = function(e) { print(e) })
tryCatch({
  seq_knots(0, Inf, N = 5)
}, error = function(e) { print(e) })
tryCatch({
  seq_knots(-Inf, 1, N = 5)
}, error = function(e) { print(e) })
```

Index

Categorical, [2](#)
d_gumbel (Gumbel), [3](#)
fprintf (Print), [6](#)
Gumbel, [3](#)
inv_rect (rect), [7](#)
log_add2_exp (log_sum_exp), [4](#)
log_sub2_exp (log_sum_exp), [4](#)
log_sum_exp, [4](#)
logger (Print), [6](#)
optimize_hybrid, [5](#)
p_gumbel (Gumbel), [3](#)
Print, [6](#)
printf (Print), [6](#)
q_gumbel (Gumbel), [3](#)
r_categ (Categorical), [2](#)
r_gumbel (Gumbel), [3](#)
rect, [7](#)
seq_knots, [8](#)
vws (vws-package), [2](#)
vws-package, [2](#)