Package ‘boostmath’

September 15, 2025

Title 'R' Bindings for the 'Boost' Math Functions

Version 1.2.0

Description 'R' bindings for the various functions and statistical distributions provided by the 'Boost' Math library <https://www.boost.org/doc/libs/latest/libs/math/doc/html/index.html>.

License MIT + file LICENSE

URL https://github.com/andrjohns/boostmath, https://www.boost.org/doc/libs/latest/libs/math/doc/html/index.html

BugReports https://github.com/andrjohns/boostmath/issues

Encoding UTF-8

RoxygenNote 7.3.3

Depends R (>= 3.0.2)

LinkingTo cpp11, BH (>= 1.87.0)

NeedsCompilation yes

Suggests knitr, rmarkdown, tinytest

VignetteBuilder knitr

Author Andrew R. Johnson [aut, cre] (ORCID: <https://orcid.org/0000-0001-7000-8065>)

Maintainer Andrew R. Johnson <andrew.johnson@arjohnsonau.com>

Repository CRAN

Date/Publication 2025-09-15 12:30:02 UTC

Contents

`airy_functions` *Airy Functions*

Description

Functions to compute the Airy functions Ai and Bi, their derivatives, and their zeros.

Usage

```
airy_ai(x)

airy_bi(x)

airy_ai_prime(x)

airy_bi_prime(x)

airy_ai_zero(m = NULL, start_index = NULL, number_of_zeros = NULL)

airy_bi_zero(m = NULL, start_index = NULL, number_of_zeros = NULL)
```

Arguments

`x`	Input numeric value
`m`	The index of the zero to find (1-based).
`start_index`	The starting index for the zeros (1-based).
`number_of_zeros`	The number of zeros to find.

Value

Single numeric value for the Airy functions and their derivatives, or a vector of length `number_of_zeros` for the multiple zero functions.

See Also

Boost Documentation for more details on the mathematical background.

Examples

```
airy_ai(2)
airy_bi(2)
airy_ai_prime(2)
airy_bi_prime(2)
airy_ai_zero(1)
airy_bi_zero(1)
airy_ai_zero(start_index = 1, number_of_zeros = 5)
airy_bi_zero(start_index = 1, number_of_zeros = 5)
```

anderson_darling_test *Anderson-Darling Test for Normality*

Description

Functions to perform the Anderson-Darling test for normality.

Usage

```
anderson_darling_normality_statistic(x, mu = 0, sd = 1)
```

Arguments

x	A numeric vector.
mu	A single numeric value.
sd	A single numeric value.

Value

A numeric value or vector with the computed statistic.

See Also

Boost Documentation for more details on the mathematical background.

Examples

```
# Anderson-Darling test for normality
anderson_darling_normality_statistic(c(1, 2, 3, 4, 5), 0, 1)
```

arcsine_distribution *Arcsine Distribution Functions*

Description

Functions to compute the probability density function, cumulative distribution function, and quantile function for the arcsine distribution.

Usage

```
arcsine_pdf(x, x_min = 0, x_max = 1)

arcsine_lpdf(x, x_min = 0, x_max = 1)

arcsine_cdf(x, x_min = 0, x_max = 1)

arcsine_lcdf(x, x_min = 0, x_max = 1)

arcsine_quantile(p, x_min = 0, x_max = 1)
```

Arguments

x	quantile
x_min	minimum value of the distribution (default is 0)
x_max	maximum value of the distribution (default is 1)
p	probability

Value

A single numeric value with the computed probability density, log-probability density, cumulative distribution, log-cumulative distribution, or quantile depending on the function called.

See Also

Boost Documentation for more details on the mathematical background.

Examples

```
arcsine_pdf(0.5)
arcsine_lpdf(0.5)
arcsine_cdf(0.5)
arcsine_lcdf(0.5)
arcsine_quantile(0.5)
```

---

barycentric_rational *Barycentric Rational Interpolation*

Description

Constructs a barycentric rational interpolator given data points.

Usage

```
barycentric_rational(x, y, order = 3)
```

Arguments

`x`	Numeric vector of data points (abscissas).
`y`	Numeric vector of data values (ordinates).
`order`	Integer representing the approximation order of the interpolator, defaults to 3.

Value

An object of class `barycentric_rational_interpolator` with methods:

- `interpolate(xi)`: Evaluate the interpolator at point `xi`.
- `prime(xi)`: Evaluate the derivative of the interpolator at point `xi`.

Examples

```
x <- c(0, 1, 2, 3)
y <- c(1, 2, 0, 2)
order <- 3
interpolator <- barycentric_rational(x, y, order)
xi <- 1.5
interpolator$interpolate(xi)
interpolator$prime(xi)
```

`basic_functions` *Basic Mathematical Functions*

Description

Functions to compute sine, cosine, logarithm, exponential, cube root, square root, power, hypotenuse, and inverse square root.

Usage

```
sin_pi(x)

cos_pi(x)

log1p_boost(x)

expm1_boost(x)

cbrt(x)

sqrt1pm1(x)

powm1(x, y)

hypot(x, y)

rsqrt(x)
```

Arguments

x	Input numeric value
y	Second input numeric value (for power and hypotenuse functions)

Value

A single numeric value with the computed result of the function.

See Also

Boost Documentation) for more details on the mathematical background.

Examples

```
# sin(pi * 0.5)
sin_pi(0.5)
# cos(pi * 0.5)
cos_pi(0.5)
# log(1 + 0.5)
log1p_boost(0.5)
# exp(0.5) - 1
expm1_boost(0.5)
cbrt(8)
# sqrt(1 + 0.5) - 1
sqrt1pm1(0.5)
# 2^3 - 1
powm1(2, 3)
hypot(3, 4)
rsqrt(4)
```

---

bernoulli_distribution

Bernoulli Distribution Functions

Description

Functions to compute the probability density function, cumulative distribution function, and quantile function for the Bernoulli distribution.

Usage

```
bernoulli_pdf(x, p_success)

bernoulli_lpdf(x, p_success)

bernoulli_cdf(x, p_success)

bernoulli_lcdf(x, p_success)

bernoulli_quantile(p, p_success)
```

Arguments

x	quantile (0 or 1)
p_success	probability of success (0 <= p_success <= 1)
p	probability (0 <= p <= 1)

Value

A single numeric value with the computed probability density, log-probability density, cumulative distribution, log-cumulative distribution, or quantile depending on the function called.

See Also

Boost Documentation for more details on the mathematical background.

Examples

```
bernoulli_pdf(1, 0.5)
bernoulli_lpdf(1, 0.5)
bernoulli_cdf(1, 0.5)
bernoulli_lcdf(1, 0.5)
bernoulli_quantile(0.5, 0.5)
```

---

bessel_functions *Bessel Functions, Their Derivatives, and Zeros*

Description

Functions to compute Bessel functions of the first and second kind, their modified versions, spherical Bessel functions, and their derivatives and zeros.

Usage

```
cyl_bessel_j(v, x)

cyl_neumann(v, x)

cyl_bessel_j_zero(v, m = NULL, start_index = NULL, number_of_zeros = NULL)

cyl_neumann_zero(v, m = NULL, start_index = NULL, number_of_zeros = NULL)

cyl_bessel_i(v, x)

cyl_bessel_k(v, x)

sph_bessel(v, x)

sph_neumann(v, x)
```

```
cyl_bessel_j_prime(v, x)

cyl_neumann_prime(v, x)

cyl_bessel_i_prime(v, x)

cyl_bessel_k_prime(v, x)

sph_bessel_prime(v, x)

sph_neumann_prime(v, x)
```

Arguments

v	Order of the Bessel function
x	Argument of the Bessel function
m	The index of the zero to find (1-based).
start_index	The starting index for the zeros (1-based).
number_of_zeros	The number of zeros to find.

Value

Single numeric value for the Bessel functions and their derivatives, or a vector of length `number_of_zeros` for the multiple zero functions.

See Also

Boost Documentation for more details on the mathematical background.

Examples

```
# Bessel function of the first kind J_0(1)
cyl_bessel_j(0, 1)
# Bessel function of the second kind Y_0(1)
cyl_neumann(0, 1)
# Modified Bessel function of the first kind I_0(1)
cyl_bessel_i(0, 1)
# Modified Bessel function of the second kind K_0(1)
cyl_bessel_k(0, 1)
# Spherical Bessel function of the first kind j_0(1)
sph_bessel(0, 1)
# Spherical Bessel function of the second kind y_0(1)
sph_neumann(0, 1)
# Derivative of the Bessel function of the first kind J_0(1)
cyl_bessel_j_prime(0, 1)
# Derivative of the Bessel function of the second kind Y_0(1)
cyl_neumann_prime(0, 1)
# Derivative of the modified Bessel function of the first kind I_0(1)
```

```
cyl_bessel_i_prime(0, 1)
# Derivative of the modified Bessel function of the second kind K_0(1)
cyl_bessel_k_prime(0, 1)
# Derivative of the spherical Bessel function of the first kind j_0(1)
sph_bessel_prime(0, 1)
# Derivative of the spherical Bessel function of the second kind y_0(1)
sph_neumann_prime(0, 1)
# Finding the first zero of the Bessel function of the first kind J_0
cyl_bessel_j_zero(0, 1)
# Finding the first zero of the Bessel function of the second kind Y_0
cyl_neumann_zero(0, 1)
# Finding multiple zeros of the Bessel function of the first kind J_0 starting from index 1
cyl_bessel_j_zero(0, start_index = 1, number_of_zeros = 5)
# Finding multiple zeros of the Bessel function of the second kind Y_0 starting from index 1
cyl_neumann_zero(0, start_index = 1, number_of_zeros = 5)
```

---

beta_distribution *Beta Distribution Functions*

Description

Functions to compute the probability density function, cumulative distribution function, and quantile function for the Beta distribution.

Usage

```
beta_pdf(x, alpha, beta)

beta_lpdf(x, alpha, beta)

beta_cdf(x, alpha, beta)

beta_lcdf(x, alpha, beta)

beta_quantile(p, alpha, beta)
```

Arguments

x	quantile (0 <= x <= 1)
alpha	shape parameter (alpha > 0)
beta	shape parameter (beta > 0)
p	probability (0 <= p <= 1)

Value

A single numeric value with the computed probability density, log-probability density, cumulative distribution, log-cumulative distribution, or quantile depending on the function called.

See Also

Boost Documentation for more details on the mathematical background.

Examples

```
# Beta distribution with shape parameters alpha = 2, beta = 5
beta_pdf(0.5, 2, 5)
beta_lpdf(0.5, 2, 5)
beta_cdf(0.5, 2, 5)
beta_lcdf(0.5, 2, 5)
beta_quantile(0.5, 2, 5)
```

`beta_functions` *Beta Functions*

Description

Functions to compute the Euler beta function, normalised incomplete beta function, and their complements, as well as their inverses and derivatives.

Usage

```
beta_boost(a, b, x = NULL)

ibeta(a, b, x)

ibetac(a, b, x)

betac(a, b, x)

ibeta_inv(a, b, p)

ibetac_inv(a, b, q)

ibeta_inva(b, x, p)

ibetac_inva(b, x, q)

ibeta_invb(a, x, p)

ibetac_invb(a, x, q)

ibeta_derivative(a, b, x)
```

Arguments

a	First parameter of the beta function
b	Second parameter of the beta function
x	Upper limit of integration (0 <= x <= 1)
p	Probability value (0 <= p <= 1)
q	Probability value (0 <= q <= 1)

Value

A single numeric value with the computed beta function, normalised incomplete beta function, or their complements, depending on the function called.

See Also

Boost Documentation for more details on the mathematical background.

Examples

```
## Not run:
# Euler beta function B(2, 3)
beta_boost(2, 3)
# Normalised incomplete beta function I_x(2, 3) for x = 0.5
ibeta(2, 3, 0.5)
# Normalised complement of the incomplete beta function 1 - I_x(2, 3) for x = 0.5
ibetac(2, 3, 0.5)
# Full incomplete beta function B_x(2, 3) for x = 0.5
beta_boost(2, 3, 0.5)
# Full complement of the incomplete beta function 1 - B_x(2, 3) for x = 0.5
betac(2, 3, 0.5)
# Inverse of the normalised incomplete beta function I_x(2, 3) = 0.5
ibeta_inv(2, 3, 0.5)
# Inverse of the normalised complement of the incomplete beta function I_x(2, 3) = 0.5
ibetac_inv(2, 3, 0.5)
# Inverse of the normalised complement of the incomplete beta function I_x(a, b)
# with respect to a for x = 0.5 and q = 0.5
ibetac_inva(3, 0.5, 0.5)
# Inverse of the normalised incomplete beta function I_x(a, b)
# with respect to b for x = 0.5 and p = 0.5
ibeta_invb(0.8, 0.5, 0.5)
# Inverse of the normalised complement of the incomplete beta function I_x(a, b)
# with respect to b for x = 0.5 and q = 0.5
ibetac_invb(2, 0.5, 0.5)
# Derivative of the incomplete beta function with respect to x for a = 2, b = 3, x = 0.5
ibeta_derivative(2, 3, 0.5)

## End(Not run)
```

bezier_polynomial *Bezier Polynomial Interpolator*

Description

Constructs a Bezier polynomial interpolator given control points.

Usage

```
bezier_polynomial(control_points)
```

Arguments

`control_points` List of control points, where each element is a numeric vector of length 3.

Value

An object of class `bezier_polynomial` with methods:

- `interpolate(xi)`: Evaluate the interpolator at point `xi`.
- `prime(xi)`: Evaluate the derivative of the interpolator at point `xi`.
- `edit_control_point(new_control_point, index)`: Insert a new control point at the specified index.

Examples

```
control_points <- list(c(0, 0, 0), c(1, 2, 0), c(2, 0, 0), c(3, 3, 0))
interpolator <- bezier_polynomial(control_points)
xi <- 1.5
interpolator$interpolate(xi)
interpolator$prime(xi)
new_control_point <- c(1.5, 1, 0)
interpolator$edit_control_point(new_control_point, 2)
```

bilinear_uniform *Bilinear Uniform Interpolator*

Description

Constructs a bilinear uniform interpolator given a grid of data points.

Usage

```
bilinear_uniform(x, rows, cols, dx = 1, dy = 1, x0 = 0, y0 = 0)
```

Arguments

`x`	Numeric vector of all grid elements
`rows`	Integer representing the number of rows in the grid
`cols`	Integer representing the number of columns in the grid
`dx`	Numeric value representing the spacing between grid points in the x-direction, defaults to 1
`dy`	Numeric value representing the spacing between grid points in the y-direction, defaults to 1
`x0`	Numeric value representing the x-coordinate of the origin, defaults to 0
`y0`	Numeric value representing the y-coordinate of the origin, defaults to 0

Value

An object of class `bilinear_uniform` with methods:

- `interpolate(xi, yi)`: Evaluate the interpolator at point `(xi, yi)`.

Examples

```
x <- seq(0, 1, length.out = 10)
interpolator <- bilinear_uniform(x, rows = 2, cols = 5)
xi <- 0.5
yi <- 0.5
interpolator$interpolate(xi, yi)
```

`binomial_distribution` *Binomial Distribution Functions*

Description

Functions to compute the probability density function, cumulative distribution function, and quantile function for the Binomial distribution.

Usage

```
binomial_pdf(k, n, prob)

binomial_lpdf(k, n, prob)

binomial_cdf(k, n, prob)

binomial_lcdf(k, n, prob)

binomial_quantile(p, n, prob)
```

Arguments

k	number of successes (0 <= k <= n)
n	number of trials (n >= 0)
prob	probability of success on each trial (0 <= prob <= 1)
p	probability (0 <= p <= 1)

Value

A single numeric value with the computed probability density, log-probability density, cumulative distribution, log-cumulative distribution, or quantile depending on the function called.

See Also

Boost Documentation for more details on the mathematical background.

Examples

```
# Binomial dist ribution with n = 10, prob = 0.5
binomial_pdf(3, 10, 0.5)
binomial_lpdf(3, 10, 0.5)
binomial_cdf(3, 10, 0.5)
binomial_lcdf(3, 10, 0.5)
binomial_quantile(0.5, 10, 0.5)
```

---

bivariate_statistics *Bivariate Statistics Functions*

Description

Functions to compute various bivariate statistics.

Usage

```
covariance(x, y)

means_and_covariance(x, y)

correlation_coefficient(x, y)
```

Arguments

x	A numeric vector.
y	A numeric vector.

Value

A numeric value or vector with the computed statistic.

See Also

Boost Documentation for more details on the mathematical background.

Examples

```
# Covariance
covariance(c(1, 2, 3), c(4, 5, 6))
# Means and Covariance
means_and_covariance(c(1, 2, 3), c(4, 5, 6))
# Correlation Coefficient
correlation_coefficient(c(1, 2, 3), c(4, 5, 6))
```

---

cardinal_cubic_b_spline   *Cardinal Cubic B-Spline Interpolator*

Description

Constructs a cardinal cubic B-spline interpolator given data points.

Usage

```
cardinal_cubic_b_spline(
  y,
  t0,
  h,
  left_endpoint_derivative = NULL,
  right_endpoint_derivative = NULL
)
```

Arguments

`y`	Numeric vector of data points to interpolate.
`t0`	Numeric scalar representing the starting point of the data.
`h`	Numeric scalar representing the spacing between data points.
`left_endpoint_derivative`	Optional numeric scalar for the derivative at the left endpoint.
`right_endpoint_derivative`	Optional numeric scalar for the derivative at the right endpoint.

Value

An object of class `cardinal_cubic_b_spline` with methods:

- `interpolate(x)`: Evaluate the spline at point `x`.
- `prime(x)`: Evaluate the first derivative of the spline at point `x`.
- `double_prime(x)`: Evaluate the second derivative of the spline at point `x`.

Examples

```
y <- c(1, 2, 0, 2, 1)
t0 <- 0
h <- 1
spline_obj <- cardinal_cubic_b_spline(y, t0, h)
x <- 0.5
spline_obj$interpolate(x)
spline_obj$prime(x)
spline_obj$double_prime(x)
```

---

cardinal_cubic_hermite

Cardinal Cubic Hermite Interpolator

Description

Constructs a cardinal cubic Hermite interpolator given the vectors of abscissas, ordinates, and derivatives.

Usage

```
cardinal_cubic_hermite(y, dydx, x0, dx)
```

Arguments

`y`	Numeric vector of ordinates (y-coordinates).
`dydx`	Numeric vector of derivatives (slopes) at each point.
`x0`	Numeric value of the first abscissa (x-coordinate).
`dx`	Numeric value of the spacing between abscissas.

Value

An object of class `cardinal_cubic_hermite` with methods:

- `interpolate(xi)`: Evaluate the interpolator at point `xi`.
- `prime(xi)`: Evaluate the derivative of the interpolator at point `xi`.
- `domain()`: Get the domain of the interpolator.

Examples

```
y <- c(0, 1, 0)
dydx <- c(1, 0, -1)
interpolator <- cardinal_cubic_hermite(y, dydx, 0, 1)
xi <- 0.5
interpolator$interpolate(xi)
interpolator$prime(xi)
interpolator$domain()
```

`cardinal_quadratic_b_spline` *Cardinal Quadratic B-Spline Interpolator*

Description

Constructs a cardinal quadratic B-spline interpolator given control points.

Usage

```
cardinal_quadratic_b_spline(
  y,
  t0,
  h,
  left_endpoint_derivative = NULL,
  right_endpoint_derivative = NULL
)
```

Arguments

`y`	Numeric vector of data points to interpolate.
`t0`	Numeric scalar representing the starting point of the data.
`h`	Numeric scalar representing the spacing between data points.
`left_endpoint_derivative`	Optional numeric scalar for the derivative at the left endpoint.
`right_endpoint_derivative`	Optional numeric scalar for the derivative at the right endpoint.

Value

An object of class `cardinal_quadratic_b_spline` with methods:

- `interpolate(xi)`: Evaluate the interpolator at point `xi`.
- `prime(xi)`: Evaluate the derivative of the interpolator at point `xi`.

Examples

```
y <- c(0, 1, 0, 1)
t0 <- 0
h <- 1
interpolator <- cardinal_quadratic_b_spline(y, t0, h)
xi <- 0.5
interpolator$interpolate(xi)
interpolator$prime(xi)
```

cardinal_quintic_b_spline
Cardinal Quintic B-Spline Interpolator

Description

Constructs a cardinal quintic B-spline interpolator given control points.

Usage

```
cardinal_quintic_b_spline(
  y,
  t0,
  h,
  left_endpoint_derivatives = NULL,
  right_endpoint_derivatives = NULL
)
```

Arguments

`y`	Numeric vector of data points to interpolate.
`t0`	Numeric scalar representing the starting point of the data.
`h`	Numeric scalar representing the spacing between data points.
`left_endpoint_derivatives`	Optional two-element numeric vector for the derivative at the left endpoint.
`right_endpoint_derivatives`	Optional two-element numeric vector for the derivative at the right endpoint.

Value

An object of class `cardinal_quintic_b_spline` with methods:

- `interpolate(xi)`: Evaluate the interpolator at point `xi`.
- `prime(xi)`: Evaluate the derivative of the interpolator at point `xi`.
- `double_prime(xi)`: Evaluate the second derivative of the interpolator at point `xi`.

Examples

```
y <- seq(0, 1, length.out = 20)
t0 <- 0
h <- 1
interpolator <- cardinal_quintic_b_spline(y, t0, h)
xi <- 0.5
interpolator$interpolate(xi)
interpolator$prime(xi)
interpolator$double_prime(xi)
```

`cardinal_quintic_hermite` *Cardinal Quintic Hermite Interpolator*

Description

Constructs a cardinal quintic Hermite interpolator given the vectors of ordinates, first derivatives, and second derivatives.

Usage

```
cardinal_quintic_hermite(y, dydx, d2ydx2, x0, dx)
```

Arguments

`y`	Numeric vector of ordinates (y-coordinates).
`dydx`	Numeric vector of first derivatives (slopes) at each point.
`d2ydx2`	Numeric vector of second derivatives at each point.
`x0`	Numeric value of the first abscissa (x-coordinate).
`dx`	Numeric value of the spacing between abscissas.

Value

An object of class `cardinal_quintic_hermite` with methods:

- `interpolate(xi)`: Evaluate the interpolator at point `xi`.
- `prime(xi)`: Evaluate the derivative of the interpolator at point `xi`.
- `double_prime(xi)`: Evaluate the second derivative of the interpolator at point `xi`.
- `domain()`: Get the domain of the interpolator.

Examples

```
y <- c(0, 1, 0)
dydx <- c(1, 0, -1)
d2ydx2 <- c(0, -1, 0)
x0 <- 0
dx <- 1
interpolator <- cardinal_quintic_hermite(y, dydx, d2ydx2, x0, dx)
xi <- 0.5
interpolator$interpolate(xi)
interpolator$prime(xi)
interpolator$double_prime(xi)
interpolator$domain()
```

`catmull_rom` *Catmull-Rom Interpolation*

Description

Constructs a Catmull-Rom spline interpolator given control points.

Usage

```
catmull_rom(control_points, closed = FALSE, alpha = 0.5)
```

Arguments

`control_points`	List of control points, where each element is a numeric vector of length 3.
`closed`	Logical indicating whether the spline is closed (i.e., the first and last control points are connected), defaults to false
`alpha`	Numeric scalar for the tension parameter, defaults to 0.5

Value

An object of class `catmull_rom` with methods:

- `interpolate(xi)`: Evaluate the interpolator at point `xi`.
- `prime(xi)`: Evaluate the derivative of the interpolator at point `xi`.
- `max_parameter()`: Get the maximum parameter value of the spline.
- `parameter_at_point(i)`: Get the parameter value at index `i`.

Examples

```
control_points <- list(c(0, 0, 0), c(1, 1, 0), c(2, 0, 0), c(3, 1, 0))
interpolator <- catmull_rom(control_points)
xi <- 1.5
interpolator$interpolate(xi)
interpolator$prime(xi)
interpolator$max_parameter()
interpolator$parameter_at_point(2)
```

`cauchy_distribution` *Cauchy Distribution Functions*

Description

Functions to compute the probability density function, cumulative distribution function, and quantile function for the Cauchy distribution.

Usage

```
cauchy_pdf(x, location = 0, scale = 1)

cauchy_lpdf(x, location = 0, scale = 1)

cauchy_cdf(x, location = 0, scale = 1)

cauchy_lcdf(x, location = 0, scale = 1)

cauchy_quantile(p, location = 0, scale = 1)
```

Arguments

`x`	quantile
`location`	location parameter (default is 0)
`scale`	scale parameter (default is 1)
`p`	probability (0 <= p <= 1)

Value

A single numeric value with the computed probability density, log-probability density, cumulative distribution, log-cumulative distribution, or quantile depending on the function called.

See Also

Boost Documentation for more details on the mathematical background.

Examples

```
# Cauchy distribution with location = 0, scale = 1
cauchy_pdf(0)
cauchy_lpdf(0)
cauchy_cdf(0)
cauchy_lcdf(0)
cauchy_quantile(0.5)
```

chatterjee_correlation *Chatterjee Correlation Function*

Description

Functions to compute the Chatterjee correlation.

Usage

```
chatterjee_correlation(x, y)
```

Arguments

x	A numeric vector.
y	A numeric vector.

Value

A two-element numeric vector containing the test statistic and the p-value.

See Also

Boost Documentation for more details on the mathematical background.

Examples

```
x <- c(1, 2, 3, 4, 5)
y <- c(2, 3, 5, 7, 11)
# Chatterjee correlation
chatterjee_correlation(x, y)
```

chebyshev_polynomials *Chebyshev Polynomials and Related Functions*

Description

Functions to compute Chebyshev polynomials of the first and second kind.

Usage

```
chebyshev_next(x, Tn, Tn_1)

chebyshev_t(n, x)

chebyshev_u(n, x)

chebyshev_t_prime(n, x)

chebyshev_clenshaw_recurrence(c, x)

chebyshev_clenshaw_recurrence_ab(c, a, b, x)
```

Arguments

x	Argument of the polynomial
Tn	Value of the Chebyshev polynomial $(T_n(x))$
Tn_1	Value of the Chebyshev polynomial $(T_{n-1}(x))$
n	Degree of the polynomial
c	Coefficients of the Chebyshev polynomial
a	Lower bound of the interval
b	Upper bound of the interval

Value

A single numeric value with the computed Chebyshev polynomial, its derivative, or related functions.

See Also

Boost Documentation for more details on the mathematical background.

Examples

```
# Chebyshev polynomial of the first kind T_2(0.5)
chebyshev_t(2, 0.5)
# Chebyshev polynomial of the second kind U_2(0.5)
chebyshev_u(2, 0.5)
# Derivative of the Chebyshev polynomial of the first kind T_2'(0.5)
chebyshev_t_prime(2, 0.5)
# Next Chebyshev polynomial of the first kind T_3(0.5) using T_2(0.5) and T_1(0.5)
chebyshev_next(0.5, chebyshev_t(2, 0.5), chebyshev_t(1, 0.5))
# Chebyshev polynomial of the first kind using Clenshaw's recurrence with coefficients
# c = c(1, 0, -1) at x = 0.5
chebyshev_clenshaw_recurrence(c(1, 0, -1), 0.5)
# Chebyshev polynomial of the first kind using Clenshaw's recurrence with interval [0, 1]
chebyshev_clenshaw_recurrence_ab(c(1, 0, -1), 0, 1, 0.5)
```

chi_squared_distribution *Chi-Squared Distribution Functions*

Description

Functions to compute the probability density function, cumulative distribution function, and quantile function for the Chi-Squared distribution.

Usage

```
chi_squared_pdf(x, df)

chi_squared_lpdf(x, df)

chi_squared_cdf(x, df)

chi_squared_lcdf(x, df)

chi_squared_quantile(p, df)
```

Arguments

x	quantile
df	degrees of freedom (df > 0)
p	probability (0 <= p <= 1)

Value

A single numeric value with the computed probability density, log-probability density, cumulative distribution, log-cumulative distribution, or quantile depending on the function called.

See Also

Boost Documentation for more details on the mathematical background.

Examples

```
# Chi-Squared distribution with 3 degrees of freedom
chi_squared_pdf(2, 3)
chi_squared_lpdf(2, 3)
chi_squared_cdf(2, 3)
chi_squared_lcdf(2, 3)
chi_squared_quantile(0.5, 3)
```

`constants` *Boost Math Constants*

Description

Provides access to mathematical constants used in the Boost Math library.

Usage

```
constants(constant = NULL)
```

Arguments

`constant`	A string specifying the name of the constant to retrieve. If `NULL`, returns a list of all constants (see documentation below for full list).

Value

Requested constant value if `constant` is specified, otherwise a list of all available constants.

See Also

Boost Documentation for more details on the constants.

Examples

```
constants()
```

`cubic_hermite` *Cubic Hermite Interpolator*

Description

Constructs a cubic Hermite interpolator given the vectors of abscissas, ordinates, and derivatives.

Usage

```
cubic_hermite(x, y, dydx)
```

Arguments

`x`	Numeric vector of abscissas (x-coordinates).
`y`	Numeric vector of ordinates (y-coordinates).
`dydx`	Numeric vector of derivatives (slopes) at each point.

Value

An object of class `cubic_hermite` with methods:

- `interpolate(xi)`: Evaluate the interpolator at point `xi`.
- `prime(xi)`: Evaluate the derivative of the interpolator at point `xi`.
- `push_back(x, y, dydx)`: Add a new control point to the interpolator.
- `domain()`: Get the domain of the interpolator.

Examples

```
x <- c(0, 1, 2)
y <- c(0, 1, 0)
dydx <- c(1, 0, -1)
interpolator <- cubic_hermite(x, y, dydx)
xi <- 0.5
interpolator$interpolate(xi)
interpolator$prime(xi)
interpolator$push_back(3, 0, 1)
interpolator$domain()
```

---

double_exponential_quadrature

Double Exponential Quadrature

Description

Functions for numerical integration using double exponential quadrature methods such as tanh-sinh, sinh-sinh, and exp-sinh quadrature.

Usage

```
tanh_sinh(f, a, b, tol = sqrt(.Machine$double.eps), max_refinements = 15)

sinh_sinh(f, tol = sqrt(.Machine$double.eps), max_refinements = 9)

exp_sinh(f, a, b, tol = sqrt(.Machine$double.eps), max_refinements = 9)
```

Arguments

`f`	A function to integrate. It should accept a single numeric value and return a single numeric value.
`a`	The lower limit of integration.
`b`	The upper limit of integration.
`tol`	The tolerance for the approximation. Default is `sqrt(.Machine$double.eps)`.
`max_refinements`	The maximum number of refinements to apply. Default is 15 for tanh-sinh and 9 for sinh-sinh and exp-sinh.

Value

A single numeric value with the computed integral.

Examples

```
# Tanh-sinh quadrature of log(x) from 0 to 1
tanh_sinh(function(x) { log(x) * log1p(-x) }, a = 0, b = 1)
# Sinh-sinh quadrature of exp(-x^2)
sinh_sinh(function(x) { exp(-x * x) })
# Exp-sinh quadrature of exp(-3*x) from 0 to Inf
exp_sinh(function(x) { exp(-3 * x) }, a = 0, b = Inf)
```

---

elliptic_integrals *Elliptic Integrals*

Description

Functions to compute various elliptic integrals, including Carlson's elliptic integrals and incomplete elliptic integrals.

Usage

```
ellint_rf(x, y, z)

ellint_rd(x, y, z)

ellint_rj(x, y, z, p)

ellint_rc(x, y)

ellint_rg(x, y, z)

ellint_1(k, phi = NULL)

ellint_2(k, phi = NULL)

ellint_3(k, n, phi = NULL)

ellint_d(k, phi = NULL)

jacobi_zeta(k, phi)

heuman_lambda(k, phi)
```

Arguments

x	First parameter of the integral
y	Second parameter of the integral
z	Third parameter of the integral
p	Fourth parameter of the integral (for Rj)
k	Elliptic modulus (for incomplete elliptic integrals)
phi	Amplitude (for incomplete elliptic integrals)
n	Characteristic (for incomplete elliptic integrals of the third kind)

Value

A single numeric value with the computed elliptic integral.

See Also

Boost Documentation for more details on the mathematical background.

Examples

```
# Carlson's elliptic integral Rf with parameters x = 1, y = 2, z = 3
ellint_rf(1, 2, 3)
#' # Carlson's elliptic integral Rd with parameters x = 1, y = 2, z = 3
ellint_rd(1, 2, 3)
# Carlson's elliptic integral Rj with parameters x = 1, y = 2, z = 3, p = 4
ellint_rj(1, 2, 3, 4)
# Carlson's elliptic integral Rc with parameters x = 1, y = 2
ellint_rc(1, 2)
# Carlson's elliptic integral Rg with parameters x = 1, y = 2, z = 3
ellint_rg(1, 2, 3)
# Incomplete elliptic integral of the first kind with k = 0.5, phi = pi/4
ellint_1(0.5, pi / 4)
# Complete elliptic integral of the first kind
ellint_1(0.5)
# Incomplete elliptic integral of the second kind with k = 0.5, phi = pi/4
ellint_2(0.5, pi / 4)
# Complete elliptic integral of the second kind
ellint_2(0.5)
# Incomplete elliptic integral of the third kind with k = 0.5, n = 0.5, phi = pi/4
ellint_3(0.5, 0.5, pi / 4)
# Complete elliptic integral of the third kind with k = 0.5, n = 0.5
ellint_3(0.5, 0.5)
# Incomplete elliptic integral D with k = 0.5, phi = pi/4
ellint_d(0.5, pi / 4)
# Complete elliptic integral D
ellint_d(0.5)
# Jacobi zeta function with k = 0.5, phi = pi/4
jacobi_zeta(0.5, pi / 4)
# Heuman's lambda function with k = 0.5, phi = pi/4
heuman_lambda(0.5, pi / 4)
```

error_functions *Error Functions and Inverses*

Description

Functions to compute the error function, complementary error function, and their inverses.

Usage

```
erf(x)

erfc(x)

erf_inv(p)

erfc_inv(p)
```

Arguments

x	Input numeric value
p	Probability value (0 <= p <= 1)

Value

A single numeric value with the computed error function, complementary error function, or their inverses.

See Also

Boost Documentation for more details

Examples

```
# Error function
erf(0.5)
# Complementary error function
erfc(0.5)
# Inverse error function
erf_inv(0.5)
# Inverse complementary error function
erfc_inv(0.5)
```

exponential_distribution *Exponential Distribution Functions*

Description

Functions to compute the probability density function, cumulative distribution function, and quantile function for the Exponential distribution.

Usage

```
exponential_pdf(x, lambda)

exponential_lpdf(x, lambda)

exponential_cdf(x, lambda)

exponential_lcdf(x, lambda)

exponential_quantile(p, lambda)
```

Arguments

x	quantile
lambda	rate parameter (lambda > 0)
p	probability (0 <= p <= 1)

Value

A single numeric value with the computed probability density, log-probability density, cumulative distribution, log-cumulative distribution, or quantile depending on the function called.

See Also

Boost Documentation for more details on the mathematical background.

Examples

```
# Exponential distribution with rate parameter lambda = 2
exponential_pdf(1, 2)
exponential_lpdf(1, 2)
exponential_cdf(1, 2)
exponential_lcdf(1, 2)
exponential_quantile(0.5, 2)
```

exponential_integrals *Exponential Integrals*

Description

Functions to compute various exponential integrals, including En and Ei.

Usage

```
expint_en(n, z)

expint_ei(z)
```

Arguments

n	Order of the integral (for En)
z	Argument of the integral (for En and Ei)

Value

A single numeric value with the computed exponential integral.

See Also

Boost Documentation for

Examples

```
# Exponential integral En with n = 1 and z = 2
expint_en(1, 2)
# Exponential integral Ei with z = 2
expint_ei(2)
```

extreme_value_distribution *Extreme Value Distribution Functions*

Description

Functions to compute the probability density function, cumulative distribution function, and quantile function for the Extreme Value distribution.

Usage

```
extreme_value_pdf(x, location = 0, scale = 1)

extreme_value_lpdf(x, location = 0, scale = 1)

extreme_value_cdf(x, location = 0, scale = 1)

extreme_value_lcdf(x, location = 0, scale = 1)

extreme_value_quantile(p, location = 0, scale = 1)
```

Arguments

x	quantile
location	location parameter (default is 0)
scale	scale parameter (default is 1)
p	probability (0 <= p <= 1)

Value

A single numeric value with the computed probability density, log-probability density, cumulative distribution, log-cumulative distribution, or quantile depending on the function called.

See Also

Boost Documentation for more details on the mathematical background.

Examples

```
# Extreme Value distribution with location = 0, scale = 1
extreme_value_pdf(0)
extreme_value_lpdf(0)
extreme_value_cdf(0)
extreme_value_lcdf(0)
extreme_value_quantile(0.5)
```

---

factorials_and_binomial_coefficients
Factorials and Binomial Coefficients

Description

Functions to compute factorials, double factorials, rising and falling factorials, and binomial coefficients.

Usage

```
factorial_boost(i)

unchecked_factorial(i)

max_factorial()

double_factorial(i)

rising_factorial(x, i)

falling_factorial(x, i)

binomial_coefficient(n, k)
```

Arguments

i	Non-negative integer input for factorials and double factorials.
x	Base value for rising and falling factorials.
n	Total number of elements for binomial coefficients.
k	Number of elements to choose for binomial coefficients.

Value

A single numeric value with the computed factorial, double factorial, rising factorial, falling factorial, or binomial coefficient.

See Also

Boost Documentation for more details on the mathematical background.

Examples

```
# Factorial of 5
factorial_boost(5)
# Unchecked factorial of 5 (using table lookup)
unchecked_factorial(5)
# Maximum factorial value that can be computed
max_factorial()
# Double factorial of 6
double_factorial(6)
# Rising factorial of 3 with exponent 2
rising_factorial(3, 2)
# Falling factorial of 3 with exponent 2
falling_factorial(3, 2)
# Binomial coefficient "5 choose 2"
binomial_coefficient(5, 2)
```

filters *Filters*

Description

Functions to compute Daubechies wavelet and scaling filters.

Usage

```
daubechies_scaling_filter(order)

daubechies_wavelet_filter(order)
```

Arguments

order	An integer specifying the order of the Daubechies filter (must be between 1 and 19).

Value

A numeric vector of size 2*order containing the filter coefficients.

See Also

Boost Documentation for more details on the mathematical background.

Examples

```
# Daubechies Scaling Filter of order 4
daubechies_scaling_filter(4)
# Daubechies Wavelet Filter of order 4
daubechies_wavelet_filter(4)
```

fisher_f_distribution *Fisher F Distribution Functions*

Description

Functions to compute the probability density function, cumulative distribution function, and quantile function for the Fisher F distribution.

Usage

```
fisher_f_pdf(x, df1, df2)

fisher_f_lpdf(x, df1, df2)

fisher_f_cdf(x, df1, df2)

fisher_f_lcdf(x, df1, df2)

fisher_f_quantile(p, df1, df2)
```

Arguments

x	quantile
df1	degrees of freedom for the numerator (df1 > 0)
df2	degrees of freedom for the denominator (df2 > 0)
p	probability (0 <= p <= 1)

Value

A single numeric value with the computed probability density, log-probability density, cumulative distribution, log-cumulative distribution, or quantile depending on the function called.

See Also

Boost Documentation for more details on the mathematical background.

Examples

```
# Fisher F distribution with df1 = 5, df2 = 2
fisher_f_pdf(1, 5, 2)
fisher_f_lpdf(1, 5, 2)
fisher_f_cdf(1, 5, 2)
fisher_f_lcdf(1, 5, 2)
fisher_f_quantile(0.5, 5, 2)
```

---

gamma_distribution *Gamma Distribution Functions*

Description

Functions to compute the probability density function, cumulative distribution function, and quantile function for the Gamma distribution.

Usage

```
gamma_pdf(x, shape, scale)

gamma_lpdf(x, shape, scale)

gamma_cdf(x, shape, scale)

gamma_lcdf(x, shape, scale)

gamma_quantile(p, shape, scale)
```

Arguments

x	quantile
shape	shape parameter (shape > 0)
scale	scale parameter (scale > 0)
p	probability (0 <= p <= 1)

Value

A single numeric value with the computed probability density, log-probability density, cumulative distribution, log-cumulative distribution, or quantile depending on the function called.

See Also

Boost Documentation for more details on the mathematical background.

Examples

```
# Gamma distribution with shape = 3, scale = 4
gamma_pdf(2, 3, 4)
gamma_lpdf(2, 3, 4)
gamma_cdf(2, 3, 4)
gamma_lcdf(2, 3, 4)
gamma_quantile(0.5, 3, 4)
```

---

gamma_functions *Gamma Functions*

Description

Functions to compute the gamma function, its logarithm, digamma, trigamma, polygamma, and various incomplete gamma functions.

Usage

```
tgamma(z)

tgamma1pm1(z)

lgamma_boost(z)

digamma_boost(z)

trigamma_boost(z)

polygamma(n, z)

tgamma_ratio(a, b)

tgamma_delta_ratio(a, delta)

gamma_p(a, z)

gamma_q(a, z)

tgamma_lower(a, z)

tgamma_upper(a, z)

gamma_q_inv(a, q)

gamma_p_inv(a, p)

gamma_q_inva(z, q)

gamma_p_inva(z, p)

gamma_p_derivative(a, z)
```

Arguments

z	Input numeric value for the gamma function
n	Order of the polygamma function (non-negative integer)
a	Argument for the incomplete gamma functions
b	Denominator argument for the ratio of gamma functions
delta	Increment for the ratio of gamma functions
q	Probability value for the incomplete gamma functions
p	Probability value for the incomplete gamma functions

Value

A single numeric value with the computed gamma function, logarithm, digamma, trigamma, polygamma, or incomplete gamma functions.

See Also

Boost Documentation for more details on the mathematical background.

Examples

```
## Not run:
# Gamma function for z = 5
tgamma(5)
# Gamma function for (1 + z) - 1, where z = 5
tgamma1pm1(5)
# Logarithm of the gamma function for z = 5
lgamma_boost(5)
# Digamma function for z = 5
digamma_boost(5)
# Trigamma function for z = 5
trigamma_boost(5)
# Polygamma function of order 1 for z = 5
polygamma(1, 5)
# Ratio of gamma functions for a = 5, b = 3
tgamma_ratio(5, 3)
# Ratio of gamma functions with delta for a = 5, delta = 2
tgamma_delta_ratio(5, 2)
# Normalised lower incomplete gamma function P(a, z) for a = 5, z = 2
gamma_p(5, 2)
# Normalised upper incomplete gamma function Q(a, z) for a = 5, z = 2
gamma_q(5, 2)
# Full lower incomplete gamma function for a = 5, z = 2
tgamma_lower(5, 2)
# Full upper incomplete gamma function for a = 5, z = 2
tgamma_upper(5, 2)
# Inverse of the normalised upper incomplete gamma function for a = 5, q = 0.5
gamma_q_inv(5, 0.5)
# Inverse of the normalised lower incomplete gamma function for a = 5, p = 0.5
gamma_p_inv(5, 0.5)
# Inverse of the normalised upper incomplete gamma function with respect to a for z = 2, q = 0.5
gamma_q_inva(2, 0.5)
# Inverse of the normalised lower incomplete gamma function with respect to a for z = 2, p = 0.5
gamma_p_inva(2, 0.5)
# Derivative of the normalised upper incomplete gamma function for a = 5, z = 2
gamma_p_derivative(5, 2)

## End(Not run)
```

`gegenbauer_polynomials` *Gegenbauer Polynomials and Related Functions*

Description

Functions to compute Gegenbauer polynomials, their derivatives, and related functions.

Usage

```
gegenbauer(n, lambda, x)

gegenbauer_prime(n, lambda, x)

gegenbauer_derivative(n, lambda, x, k)
```

Arguments

`n`	Degree of the polynomial
`lambda`	Parameter of the polynomial
`x`	Argument of the polynomial
`k`	Order of the derivative

Value

A single numeric value with the computed Gegenbauer polynomial, its derivative, or k-th derivative.

See Also

Boost Documentation for more details on the mathematical background.

Examples

```
# Gegenbauer polynomial C_2^(1)(0.5)
gegenbauer(2, 1, 0.5)
# Derivative of the Gegenbauer polynomial C_2^(1)'(0.5)
gegenbauer_prime(2, 1, 0.5)
# k-th derivative of the Gegenbauer polynomial C_2^(1)''(0.5)
gegenbauer_derivative(2, 1, 0.5, 2)
```

geometric_distribution
Geometric Distribution Functions

Description

Functions to compute the probability density function, cumulative distribution function, and quantile function for the Geometric distribution.

Usage

```
geometric_pdf(x, prob)

geometric_lpdf(x, prob)

geometric_cdf(x, prob)

geometric_lcdf(x, prob)

geometric_quantile(p, prob)
```

Arguments

x	quantile (non-negative integer)
prob	probability of success (0 < prob < 1)
p	probability (0 <= p <= 1)

Value

A single numeric value with the computed probability density, log-probability density, cumulative distribution, log-cumulative distribution, or quantile depending on the function called.

See Also

Boost Documentation for more details on the mathematical background.

Examples

```
# Geometric distribution with probability of success prob = 0.5
geometric_pdf(3, 0.5)
geometric_lpdf(3, 0.5)
geometric_cdf(3, 0.5)
geometric_lcdf(3, 0.5)
geometric_quantile(0.5, 0.5)
```

`hankel_functions` *Hankel Functions*

Description

Functions to compute cyclic and spherical Hankel functions of the first and second kinds.

Usage

```
cyl_hankel_1(v, x)

cyl_hankel_2(v, x)

sph_hankel_1(v, x)

sph_hankel_2(v, x)
```

Arguments

`v`	Order of the Hankel function
`x`	Argument of the Hankel function

Value

A single complex value with the computed Hankel function.

See Also

Boost Documentation for more details on the mathematical background.

Examples

```
cyl_hankel_1(2, 0.5)
cyl_hankel_2(2, 0.5)
sph_hankel_1(2, 0.5)
sph_hankel_2(2, 0.5)
```

`hermite_polynomials` *Hermite Polynomials and Related Functions*

Description

Functions to compute Hermite polynomials.

Usage

```
hermite(n, x)

hermite_next(n, x, Hn, Hnm1)
```

Arguments

n	Degree of the polynomial
x	Argument of the polynomial
Hn	Value of the Hermite polynomial $(H_n(x))$
Hnm1	Value of the Hermite polynomial $(H_{n-1}(x))$

Value

A single numeric value with the computed Hermite polynomial or its next value.

See Also

Boost Documentation for more details on the mathematical background.

Examples

```
# Hermite polynomial H_2(0.5)
hermite(2, 0.5)
# Next Hermite polynomial H_3(0.5) using H_2(0.5) and H_1(0.5)
hermite_next(2, 0.5, hermite(2, 0.5), hermite(1, 0.5))
```

---

holtsmark_distribution *Holtsmark Distribution Functions*

Description

Functions to compute the probability density function, cumulative distribution function, and quantile function for the Holtsmark distribution.

Usage

```
holtsmark_pdf(x, location = 0, scale = 1)

holtsmark_lpdf(x, location = 0, scale = 1)

holtsmark_cdf(x, location = 0, scale = 1)

holtsmark_lcdf(x, location = 0, scale = 1)

holtsmark_quantile(p, location = 0, scale = 1)
```

Arguments

x	quantile
location	location parameter (default is 0)
scale	scale parameter (default is 1)
p	probability (0 <= p <= 1)

Value

A single numeric value with the computed probability density, log-probability density, cumulative distribution, log-cumulative distribution, or quantile depending on the function called.

See Also

Boost Documentation for more details on the mathematical background.

Examples

```
# Holtsmark distribution with location 0 and scale 1
holtsmark_pdf(3)
holtsmark_lpdf(3)
holtsmark_cdf(3)
holtsmark_lcdf(3)
holtsmark_quantile(0.5)
```

---

hyperexponential_distribution

Hyperexponential Distribution Functions

Description

Functions to compute the probability density function, cumulative distribution function, and quantile function for the Hyperexponential distribution.

Usage

```
hyperexponential_pdf(x, probabilities, rates)

hyperexponential_lpdf(x, probabilities, rates)

hyperexponential_cdf(x, probabilities, rates)

hyperexponential_lcdf(x, probabilities, rates)

hyperexponential_quantile(p, probabilities, rates)
```

Arguments

x	quantile
probabilities	vector of probabilities (sum must be 1)
rates	vector of rates (all rates must be > 0)
p	probability (0 <= p <= 1)

Value

A single numeric value with the computed probability density, log-probability density, cumulative distribution, log-cumulative distribution, or quantile depending on the function called.

See Also

Boost Documentation for more details on the mathematical background.

Examples

```
# Hyperexponential distribution with probabilities = c(0.5, 0.5) and rates = c(1, 2)
hyperexponential_pdf(2, c(0.5, 0.5), c(1, 2))
hyperexponential_lpdf(2, c(0.5, 0.5), c(1, 2))
hyperexponential_cdf(2, c(0.5, 0.5), c(1, 2))
hyperexponential_lcdf(2, c(0.5, 0.5), c(1, 2))
hyperexponential_quantile(0.5, c(0.5, 0.5), c(1, 2))
```

---

hypergeometric_distribution

Hypergeometric Distribution Functions

Description

Functions to compute the probability density function, cumulative distribution function, and quantile function for the Hypergeometric distribution.

Usage

```
hypergeometric_pdf(x, r, n, N)

hypergeometric_lpdf(x, r, n, N)

hypergeometric_cdf(x, r, n, N)

hypergeometric_lcdf(x, r, n, N)

hypergeometric_quantile(p, r, n, N)
```

Arguments

x	quantile (non-negative integer)
r	number of successes in the population (r >= 0)
n	number of draws (n >= 0)
N	population size (N >= r)
p	probability (0 <= p <= 1)

Value

A single numeric value with the computed probability density, log-probability density, cumulative distribution, log-cumulative distribution, or quantile depending on the function called.

See Also

Boost Documentation for more details on the mathematical background.

Examples

```
# Hypergeometric distribution with r = 5, n = 10, N = 20
hypergeometric_pdf(3, 5, 10, 20)
hypergeometric_lpdf(3, 5, 10, 20)
hypergeometric_cdf(3, 5, 10, 20)
hypergeometric_lcdf(3, 5, 10, 20)
hypergeometric_quantile(0.5, 5, 10, 20)
```

---

hypergeometric_functions

Hypergeometric Functions

Description

Functions to compute various hypergeometric functions.

Usage

```
hypergeometric_1F0(a, z)

hypergeometric_0F1(b, z)

hypergeometric_2F0(a1, a2, z)

hypergeometric_1F1(a, b, z)

hypergeometric_1F1_regularized(a, b, z)

log_hypergeometric_1F1(a, b, z)

hypergeometric_pFq(a, b, z)
```

Arguments

a	Parameter of the hypergeometric function
z	Argument of the hypergeometric function
b	Second parameter of the hypergeometric function
a1	First parameter of the hypergeometric function
a2	Second parameter of the hypergeometric function

Value

A single numeric value with the computed hypergeometric function.

See Also

Boost Documentation for more details on the mathematical background.

Examples

```
# Hypergeometric Function 1F0
hypergeometric_1F0(2, 0.2)
# Hypergeometric Function 0F1
hypergeometric_0F1(1, 0.8)
# Hypergeometric Function 2F0
hypergeometric_2F0(0.1, -1, 0.1)
# Hypergeometric Function 1F1
hypergeometric_1F1(2, 3, 1)
# Regularised Hypergeometric Function 1F1
hypergeometric_1F1_regularized(2, 3, 1)
# Logarithm of the Hypergeometric Function 1F1
log_hypergeometric_1F1(2, 3, 1)
# Hypergeometric Function pFq
hypergeometric_pFq(c(2, 3), c(4, 5), 6)
```

---

inverse_chi_squared_distribution

Inverse Chi-Squared Distribution Functions

Description

Functions to compute the probability density function, cumulative distribution function, and quantile function for the Inverse Chi-Squared distribution.

Usage

```
inverse_chi_squared_pdf(x, df, scale = 1)

inverse_chi_squared_lpdf(x, df, scale = 1)

inverse_chi_squared_cdf(x, df, scale = 1)

inverse_chi_squared_lcdf(x, df, scale = 1)

inverse_chi_squared_quantile(p, df, scale = 1)
```

Arguments

`x`	quantile
`df`	degrees of freedom (df > 0)
`scale`	scale parameter (default is 1)
`p`	probability (0 <= p <= 1)

Value

A single numeric value with the computed probability density, log-probability density, cumulative distribution, log-cumulative distribution, or quantile depending on the function called.

See Also

Boost Documentation for more details on the mathematical background.

Examples

```
# Inverse Chi-Squared distribution with 3 degrees of freedom, scale = 1
inverse_chi_squared_pdf(2, 3, 1)
inverse_chi_squared_lpdf(2, 3, 1)
inverse_chi_squared_cdf(2, 3, 1)
inverse_chi_squared_lcdf(2, 3, 1)
inverse_chi_squared_quantile(0.5, 3, 1)
```

`inverse_gamma_distribution`

Inverse Gamma Distribution Functions

Description

Functions to compute the probability density function, cumulative distribution function, and quantile function for the Inverse Gamma distribution.

Usage

```
inverse_gamma_pdf(x, shape, scale)

inverse_gamma_lpdf(x, shape, scale)

inverse_gamma_cdf(x, shape, scale)

inverse_gamma_lcdf(x, shape, scale)

inverse_gamma_quantile(p, shape, scale)
```

Arguments

x	quantile
shape	shape parameter (shape > 0)
scale	scale parameter (scale > 0)
p	probability (0 <= p <= 1)

Value

A single numeric value with the computed probability density, log-probability density, cumulative distribution, log-cumulative distribution, or quantile depending on the function called.

See Also

Boost Documentation for more details on the mathematical background.

Examples

```
# Inverse Gamma distribution with shape = 3, scale = 4
inverse_gamma_pdf(2, 3, 4)
inverse_gamma_lpdf(2, 3, 4)
inverse_gamma_cdf(2, 3, 4)
inverse_gamma_lcdf(2, 3, 4)
inverse_gamma_quantile(0.5, 3, 4)
```

---

inverse_gaussian_distribution — *Inverse Gaussian Distribution Functions*

Description

Functions to compute the probability density function, cumulative distribution function, and quantile function for the Inverse Gaussian distribution.

Usage

```
inverse_gaussian_pdf(x, mu, lambda)

inverse_gaussian_lpdf(x, mu, lambda)

inverse_gaussian_cdf(x, mu, lambda)

inverse_gaussian_lcdf(x, mu, lambda)

inverse_gaussian_quantile(p, mu, lambda)
```

Arguments

x	quantile
mu	mean parameter (mu > 0)
lambda	scale (precision) parameter (lambda > 0)
p	probability (0 <= p <= 1)

Value

A single numeric value with the computed probability density, log-probability density, cumulative distribution, log-cumulative distribution, or quantile depending on the function called.

See Also

Boost Documentation for more details on the mathematical background.

Examples

```
# Inverse Gaussian distribution with mu = 3, lambda = 4
inverse_gaussian_pdf(2, 3, 4)
inverse_gaussian_lpdf(2, 3, 4)
inverse_gaussian_cdf(2, 3, 4)
inverse_gaussian_lcdf(2, 3, 4)
inverse_gaussian_quantile(0.5, 3, 4)
```

---

inverse_hyperbolic_functions
Inverse Hyperbolic Functions

Description

Functions to compute the inverse hyperbolic functions: acosh, asinh, and atanh.

Usage

```
acosh_boost(x)

asinh_boost(x)

atanh_boost(x)
```

Arguments

x	Input numeric value

Value

A single numeric value with the computed inverse hyperbolic function.

See Also

Boost Documentation for more details on the mathematical background.

Examples

```
# Inverse Hyperbolic Cosine
acosh_boost(2)
# Inverse Hyperbolic Sine
asinh_boost(1)
# Inverse Hyperbolic Tangent
atanh_boost(0.5)
```

---

jacobi_elliptic_functions

Jacobi Elliptic Functions

Description

Functions to compute the Jacobi elliptic functions: sn, cn, dn, and others.

Usage

```
jacobi_elliptic(k, u)

jacobi_cd(k, u)

jacobi_cn(k, u)

jacobi_cs(k, u)

jacobi_dc(k, u)
```

```
jacobi_dn(k, u)

jacobi_ds(k, u)

jacobi_nc(k, u)

jacobi_nd(k, u)

jacobi_ns(k, u)

jacobi_sc(k, u)

jacobi_sd(k, u)

jacobi_sn(k, u)
```

Arguments

k	Elliptic modulus (0 <= k < 1)
u	Argument of the elliptic functions

Value

For `jacobi_elliptic`, a list containing the values of the Jacobi elliptic functions: `sn`, `cn`, `dn`. For individual functions, a single numeric value is returned.

See Also

Boost Documentation for more details on the mathematical background.

Examples

```
# Jacobi Elliptic Functions
k <- 0.5
u <- 2
jacobi_elliptic(k, u)
# Individual Jacobi Elliptic Functions
jacobi_cd(k, u)
jacobi_cn(k, u)
jacobi_cs(k, u)
jacobi_dc(k, u)
jacobi_dn(k, u)
jacobi_ds(k, u)
jacobi_nc(k, u)
jacobi_nd(k, u)
jacobi_ns(k, u)
jacobi_sc(k, u)
jacobi_sd(k, u)
jacobi_sn(k, u)
```

jacobi_polynomials *Jacobi Polynomials and Related Functions*

Description

Functions to compute Jacobi polynomials, their derivatives, and related functions.

Usage

```
jacobi(n, alpha, beta, x)

jacobi_prime(n, alpha, beta, x)

jacobi_double_prime(n, alpha, beta, x)

jacobi_derivative(n, alpha, beta, x, k)
```

Arguments

n	Degree of the polynomial
alpha	First parameter of the polynomial
beta	Second parameter of the polynomial
x	Argument of the polynomial
k	Order of the derivative

Value

A single numeric value with the computed Jacobi polynomial, its derivative, or k-th derivative.

See Also

Boost Documentation for more details on the mathematical background.

Examples

```
# Jacobi polynomial P_2^(1, 2)(0.5)
jacobi(2, 1, 2, 0.5)
# Derivative of the Jacobi polynomial P_2^(1, 2)'(0.5)
jacobi_prime(2, 1, 2, 0.5)
# Second derivative of the Jacobi polynomial P_2^(1, 2)''(0.5)
jacobi_double_prime(2, 1, 2, 0.5)
# 3rd derivative of the Jacobi polynomial P_2^(1, 2)^(k)(0.5)
jacobi_derivative(2, 1, 2, 0.5, 3)
```

`jacobi_theta_functions` *Jacobi Theta Functions*

Description

Functions to compute the Jacobi theta functions $(\theta_1, \theta_2, \theta_3, \theta_4)$ parameterised by either (q) or (τ).

Usage

```
jacobi_theta1(x, q)

jacobi_theta1tau(x, tau)

jacobi_theta2(x, q)

jacobi_theta2tau(x, tau)

jacobi_theta3(x, q)

jacobi_theta3tau(x, tau)

jacobi_theta3m1(x, q)

jacobi_theta3m1tau(x, tau)

jacobi_theta4(x, q)

jacobi_theta4tau(x, tau)

jacobi_theta4m1(x, q)

jacobi_theta4m1tau(x, tau)
```

Arguments

`x`	Input value
`q`	The nome parameter of the Jacobi theta function (0 < q < 1)
`tau`	The nome parameter of the Jacobi theta function (tau = u + iv, where u and v are real numbers)

Value

A single numeric value with the computed Jacobi theta function.

See Also

Boost Documentation for more details on the mathematical background.

Examples

```
# Jacobi Theta Functions
x <- 0.5
q <- 0.9
tau <- 1.5
jacobi_theta1(x, q)
jacobi_theta1tau(x, tau)
jacobi_theta2(x, q)
jacobi_theta2tau(x, tau)
jacobi_theta3(x, q)
jacobi_theta3tau(x, tau)
jacobi_theta3m1(x, q)
jacobi_theta3m1tau(x, tau)
jacobi_theta4(x, q)
jacobi_theta4tau(x, tau)
jacobi_theta4m1(x, q)
jacobi_theta4m1tau(x, tau)
```

---

kolmogorov_smirnov_distribution

Kolmogorov-Smirnov Distribution Functions

Description

Functions to compute the probability density function, cumulative distribution function, and quantile function for the Kolmogorov-Smirnov distribution.

Usage

```
kolmogorov_smirnov_pdf(x, n)

kolmogorov_smirnov_lpdf(x, n)

kolmogorov_smirnov_cdf(x, n)

kolmogorov_smirnov_lcdf(x, n)

kolmogorov_smirnov_quantile(p, n)
```

Arguments

x	quantile
n	sample size (n > 0)
p	probability (0 <= p <= 1)

Value

A single numeric value with the computed probability density, log-probability density, cumulative distribution, log-cumulative distribution, or quantile depending on the function called.

See Also

Boost Documentation for more details on the mathematical background.

Examples

```
# Kolmogorov-Smirnov distribution with sample size n = 10
kolmogorov_smirnov_pdf(0.5, 10)
kolmogorov_smirnov_lpdf(0.5, 10)
kolmogorov_smirnov_cdf(0.5, 10)
kolmogorov_smirnov_lcdf(0.5, 10)
kolmogorov_smirnov_quantile(0.5, 10)
```

---

laguerre_polynomials *Laguerre Polynomials and Related Functions*

Description

Functions to compute Laguerre polynomials of the first kind.

Usage

```
laguerre(n, x)

laguerre_m(n, m, x)

laguerre_next(n, x, Ln, Lnm1)

laguerre_next_m(n, m, x, Ln, Lnm1)
```

Arguments

`n`	Degree of the polynomial
`x`	Argument of the polynomial
`m`	Order of the polynomial (for Laguerre polynomials of the first kind)
`Ln`	Value of the Laguerre polynomial $(L_n(x))$
`Lnm1`	Value of the Laguerre polynomial $(L_{n-1}(x))$

Value

A single numeric value with the computed Laguerre polynomial, its derivative, or related functions.

See Also

Boost Documentation for more details on the mathematical background.

Examples

```
# Laguerre polynomial of the first kind L_2(0.5)
laguerre(2, 0.5)
# Laguerre polynomial of the first kind with order 1 L_2^1(0.5)
laguerre_m(2, 1, 0.5)
# Next Laguerre polynomial of the first kind L_3(0.5) using L_2(0.5) and L_1(0.5)
laguerre_next(2, 0.5, laguerre(2, 0.5), laguerre(1, 0.5))
# Next Laguerre polynomial of the first kind with order 1 L_3^1(0.5) using L_2^1(0.5) and L_1^1(0.5)
laguerre_next_m(2, 1, 0.5, laguerre_m(2, 1, 0.5), laguerre_m(1, 1, 0.5))
```

---

lambert_w_function *Lambert W Function and Its Derivatives*

Description

Functions to compute the Lambert W function and its derivatives for the principal branch (W_0) and the branch -1 (W_{-1}).

Usage

```
lambert_w0(z)

lambert_wm1(z)

lambert_w0_prime(z)

lambert_wm1_prime(z)
```

Arguments

z	Argument of the Lambert W function

Value

A single numeric value with the computed Lambert W function or its derivative.

See Also

Boost Documentation for more details on the mathematical background.

Examples

```
# Lambert W Function (Principal Branch)
lambert_w0(0.3)
# Lambert W Function (Branch -1)
lambert_wm1(-0.3)
# Derivative of the Lambert W Function (Principal Branch)
lambert_w0_prime(0.3)
# Derivative of the Lambert W Function (Branch -1)
lambert_wm1_prime(-0.3)
```

landau_distribution *Landau Distribution Functions*

Description

Functions to compute the probability density function, cumulative distribution function, and quantile function for the Landau distribution.

Usage

```
landau_pdf(x, location = 0, scale = 1)

landau_lpdf(x, location = 0, scale = 1)

landau_cdf(x, location = 0, scale = 1)

landau_lcdf(x, location = 0, scale = 1)

landau_quantile(p, location = 0, scale = 1)
```

Arguments

x	quantile
location	location parameter (default is 0)
scale	scale parameter (default is 1)
p	probability (0 <= p <= 1)

Value

A single numeric value with the computed probability density, log-probability density, cumulative distribution, log-cumulative distribution, or quantile depending on the function called.

See Also

Boost Documentation for more details on the mathematical background.

Examples

```
# Landau distribution with location 0 and scale 1
landau_pdf(3)
landau_lpdf(3)
landau_cdf(3)
landau_lcdf(3)
landau_quantile(0.5)
```

laplace_distribution *Laplace Distribution Functions*

Description

Functions to compute the probability density function, cumulative distribution function, and quantile function for the Laplace distribution.

Usage

```
laplace_pdf(x, location = 0, scale = 1)

laplace_lpdf(x, location = 0, scale = 1)

laplace_cdf(x, location = 0, scale = 1)

laplace_lcdf(x, location = 0, scale = 1)

laplace_quantile(p, location = 0, scale = 1)
```

Arguments

x	quantile
location	location parameter (default is 0)
scale	scale parameter (default is 1)
p	probability (0 <= p <= 1)

Value

A single numeric value with the computed probability density, log-probability density, cumulative distribution, log-cumulative distribution, or quantile depending on the function called.

See Also

Boost Documentation for more details on the mathematical background.

Examples

```
# Laplace distribution with location = 0, scale = 1
laplace_pdf(0)
laplace_lpdf(0)
laplace_cdf(0)
laplace_lcdf(0)
laplace_quantile(0.5)
```

legendre_polynomials *Legendre Polynomials and Related Functions*

Description

Functions to compute Legendre polynomials of the first and second kind, their derivatives, zeros, and related functions.

Usage

```
legendre_p(n, x)

legendre_p_prime(n, x)

legendre_p_zeros(n)

legendre_p_m(n, m, x)

legendre_q(n, x)

legendre_next(n, x, Pl, Plm1)

legendre_next_m(n, m, x, Pl, Plm1)
```

Arguments

n	Degree of the polynomial
x	Argument of the polynomial
m	Order of the polynomial (for Legendre polynomials of the first kind)
Pl	Value of the Legendre polynomial $(P_l(x))$
Plm1	Value of the Legendre polynomial $(P_{l-1}(x))$

Value

A single numeric value with the computed Legendre polynomial, its derivative, zeros, or related functions.

See Also

Boost Documentation for more details on the mathematical background.

Examples

```
# Legendre polynomial of the first kind P_2(0.5)
legendre_p(2, 0.5)
# Derivative of the Legendre polynomial of the first kind P_2'(0.5)
legendre_p_prime(2, 0.5)
```

```
# Zeros of the Legendre polynomial of the first kind P_2
legendre_p_zeros(2)
# Legendre polynomial of the first kind with order 1 P_2^1(0.5)
legendre_p_m(2, 1, 0.5)
# Legendre polynomial of the second kind Q_2(0.5)
legendre_q(2, 0.5)
# Next Legendre polynomial of the first kind P_3(0.5) using P_2(0.5) and P_1(0.5)
legendre_next(2, 0.5, legendre_p(2, 0.5), legendre_p(1, 0.5))
# Next Legendre polynomial of the first kind with order 1 P_3^1(0.5) using P_2^1(0.5) and P_1^1(0.5)
legendre_next_m(2, 1, 0.5, legendre_p_m(2, 1, 0.5), legendre_p_m(1, 1, 0.5))
```

`linear_regression` *Linear Regression Functions*

Description

Functions to perform linear regression.

Usage

```
simple_ordinary_least_squares(x, y)

simple_ordinary_least_squares_with_R_squared(x, y)
```

Arguments

x	A numeric vector.
y	A numeric vector.

Value

A two-element numeric vector containing the intercept and slope of the regression line, or a three-element vector containing the intercept, slope, and R-squared value if applicable.

See Also

Boost Documentation for more details on the mathematical background.

Examples

```
# Simple Ordinary Least Squares
x <- c(1, 2, 3, 4, 5)
y <- c(2, 3, 5, 7, 11)
simple_ordinary_least_squares(x, y)
# Simple Ordinary Least Squares with R-squared
simple_ordinary_least_squares_with_R_squared(x, y)
```

ljung_box_test — *Ljung-Box Test for Autocorrelation*

Description

Functions to perform the Ljung-Box test for autocorrelation.

Usage

```
ljung_box(v, lags = -1, fit_dof = 0)
```

Arguments

`v`	A numeric vector.
`lags`	A single integer value.
`fit_dof`	A single integer value.

Value

A two-element numeric vector containing the test statistic and the p-value.

See Also

Boost Documentation for more details on the mathematical background.

Examples

```
# Ljung-Box test for autocorrelation
ljung_box(c(1, 2, 3, 4, 5), lags = 2, fit_dof = 0)
```

logistic_distribution — *Logistic Distribution Functions*

Description

Functions to compute the probability density function, cumulative distribution function, and quantile function for the Logistic distribution.

Usage

```
logistic_pdf(x, location = 0, scale = 1)

logistic_lpdf(x, location = 0, scale = 1)

logistic_cdf(x, location = 0, scale = 1)

logistic_lcdf(x, location = 0, scale = 1)

logistic_quantile(p, location = 0, scale = 1)
```

Arguments

x	quantile
location	location parameter (default is 0)
scale	scale parameter (default is 1)
p	probability (0 <= p <= 1)

Value

A single numeric value with the computed probability density, log-probability density, cumulative distribution, log-cumulative distribution, or quantile depending on the function called.

See Also

Boost Documentation for more details on the mathematical background.

Examples

```
# Logistic distribution with location = 0, scale = 1
logistic_pdf(0)
logistic_lpdf(0)
logistic_cdf(0)
logistic_lcdf(0)
logistic_quantile(0.5)
```

---

lognormal_distribution

Log Normal Distribution Functions

Description

Functions to compute the probability density function, cumulative distribution function, and quantile function for the Log Normal distribution.

Usage

```
lognormal_pdf(x, location = 0, scale = 1)

lognormal_lpdf(x, location = 0, scale = 1)

lognormal_cdf(x, location = 0, scale = 1)

lognormal_lcdf(x, location = 0, scale = 1)

lognormal_quantile(p, location = 0, scale = 1)
```

Arguments

`x`	quantile
`location`	location parameter (default is 0)
`scale`	scale parameter (default is 1)
`p`	probability (0 <= p <= 1)

Value

A single numeric value with the computed probability density, log-probability density, cumulative distribution, log-cumulative distribution, or quantile depending on the function called.

See Also

Boost Documentation for more details on the mathematical background.

Examples

```
# Log Normal distribution with location = 0, scale = 1
lognormal_pdf(0)
lognormal_lpdf(0)
lognormal_cdf(0)
lognormal_lcdf(0)
lognormal_quantile(0.5)
```

---

makima *Modified Akima Interpolator*

Description

Constructs a Modified Akima interpolator given the vectors of abscissas, ordinates, and derivatives.

Usage

```
makima(x, y, left_endpoint_derivative = NULL, right_endpoint_derivative = NULL)
```

Arguments

`x` Numeric vector of abscissas (x-coordinates).

`y` Numeric vector of ordinates (y-coordinates).

`left_endpoint_derivative`
Optional numeric value of the derivative at the left endpoint.

`right_endpoint_derivative`
Optional numeric value of the derivative at the right endpoint.

Value

An object of class `makima` with methods:

- `interpolate(xi)`: Evaluate the interpolator at point `xi`.
- `prime(xi)`: Evaluate the derivative of the interpolator at point `xi`.
- `push_back(x, y)`: Add a new control point

Examples

```
x <- c(0, 1, 2, 3)
y <- c(0, 1, 0, 1)
interpolator <- makima(x, y)
xi <- 0.5
interpolator$interpolate(xi)
interpolator$prime(xi)
interpolator$push_back(4, 1)
```

`mapairy_distribution` *Map-Airy Distribution Functions*

Description

Functions to compute the probability density function, cumulative distribution function, and quantile function for the Map-Airy distribution.

Usage

```
mapairy_pdf(x, location = 0, scale = 1)

mapairy_lpdf(x, location = 0, scale = 1)

mapairy_cdf(x, location = 0, scale = 1)

mapairy_lcdf(x, location = 0, scale = 1)

mapairy_quantile(p, location = 0, scale = 1)
```

Arguments

x	quantile
location	location parameter (default is 0)
scale	scale parameter (default is 1)
p	probability (0 <= p <= 1)

Value

A single numeric value with the computed probability density, log-probability density, cumulative distribution, log-cumulative distribution, or quantile depending on the function called.

See Also

Boost Documentation for more details on the mathematical background.

Examples

```
# Map-Airy distribution with location 0 and scale 1
mapairy_pdf(3)
mapairy_lpdf(3)
mapairy_cdf(3)
mapairy_lcdf(3)
mapairy_quantile(0.5)
```

---

negative_binomial_distribution

Negative Binomial Distribution Functions

Description

Functions to compute the probability density function, cumulative distribution function, and quantile function for the Negative Binomial distribution.

Usage

```
negative_binomial_pdf(x, successes, success_fraction)

negative_binomial_lpdf(x, successes, success_fraction)

negative_binomial_cdf(x, successes, success_fraction)

negative_binomial_lcdf(x, successes, success_fraction)

negative_binomial_quantile(p, successes, success_fraction)
```

Arguments

`x`	quantile
`successes`	number of successes (successes >= 0)
`success_fraction`	probability of success on each trial (0 <= success_fraction <= 1)
`p`	probability (0 <= p <= 1)

Value

A single numeric value with the computed probability density, log-probability density, cumulative distribution, log-cumulative distribution, or quantile depending on the function called.

See Also

Boost Documentation for more details on the mathematical background.

Examples

```
negative_binomial_pdf(3, 5, 0.5)
negative_binomial_lpdf(3, 5, 0.5)
negative_binomial_cdf(3, 5, 0.5)
negative_binomial_lcdf(3, 5, 0.5)
negative_binomial_quantile(0.5, 5, 0.5)
```

---

non_central_beta_distribution

Noncentral Beta Distribution Functions

Description

Functions to compute the probability density function, cumulative distribution function, and quantile function for the Noncentral Beta distribution.

Usage

```
non_central_beta_pdf(x, alpha, beta, lambda)

non_central_beta_lpdf(x, alpha, beta, lambda)

non_central_beta_cdf(x, alpha, beta, lambda)

non_central_beta_lcdf(x, alpha, beta, lambda)

non_central_beta_quantile(p, alpha, beta, lambda)
```

Arguments

`x`	quantile (0 <= x <= 1)
`alpha`	first shape parameter (alpha > 0)
`beta`	second shape parameter (beta > 0)
`lambda`	noncentrality parameter (lambda >= 0)
`p`	probability (0 <= p <= 1)

Value

A single numeric value with the computed probability density, log-probability density, cumulative distribution, log-cumulative distribution, or quantile depending on the function called.

See Also

Boost Documentation for more details on the mathematical background.

Examples

```
# Noncentral Beta distribution with shape parameters alpha = 2, beta = 3
# and noncentrality parameter lambda = 1
non_central_beta_pdf(0.5, 2, 3, 1)
non_central_beta_lpdf(0.5, 2, 3, 1)
non_central_beta_cdf(0.5, 2, 3, 1)
non_central_beta_lcdf(0.5, 2, 3, 1)
non_central_beta_quantile(0.5, 2, 3, 1)
```

---

non_central_chi_squared_distribution

Noncentral Chi-Squared Distribution Functions

Description

Functions to compute the probability density function, cumulative distribution function, and quantile function for the Noncentral Chi-Squared distribution.

Usage

```
non_central_chi_squared_pdf(x, df, lambda)

non_central_chi_squared_lpdf(x, df, lambda)

non_central_chi_squared_cdf(x, df, lambda)

non_central_chi_squared_lcdf(x, df, lambda)

non_central_chi_squared_quantile(p, df, lambda)
```

Arguments

x	quantile
df	degrees of freedom (df > 0)
lambda	noncentrality parameter (lambda >= 0)
p	probability (0 <= p <= 1)

Value

A single numeric value with the computed probability density, log-probability density, cumulative distribution, log-cumulative distribution, or quantile depending on the function called.

See Also

Boost Documentation for more details on the mathematical background.

Examples

```
## Not run:
# Noncentral Chi-Squared distribution with 3 degrees of freedom and noncentrality
# parameter 1
non_central_chi_squared_pdf(2, 3, 1)
non_central_chi_squared_lpdf(2, 3, 1)
non_central_chi_squared_cdf(2, 3, 1)
non_central_chi_squared_lcdf(2, 3, 1)
non_central_chi_squared_quantile(0.5, 3, 1)

## End(Not run)
```

---

non_central_f_distribution
Noncentral F Distribution Functions

Description

Functions to compute the probability density function, cumulative distribution function, and quantile function for the Fisher F distribution.

Usage

```
non_central_f_pdf(x, df1, df2, lambda)

non_central_f_lpdf(x, df1, df2, lambda)

non_central_f_cdf(x, df1, df2, lambda)

non_central_f_lcdf(x, df1, df2, lambda)

non_central_f_quantile(p, df1, df2, lambda)
```

Arguments

x	quantile
df1	degrees of freedom for the numerator (df1 > 0)
df2	degrees of freedom for the denominator (df2 > 0)
lambda	noncentrality parameter (lambda >= 0)
p	probability (0 <= p <= 1)

Value

A single numeric value with the computed probability density, log-probability density, cumulative distribution, log-cumulative distribution, or quantile depending on the function called.

See Also

Boost Documentation for more details on the mathematical background.

Examples

```
# Noncentral F distribution with df1 = 5, df2 = 2 and noncentrality
# parameter 1
non_central_f_pdf(1, 5, 2, 1)
non_central_f_lpdf(1, 5, 2, 1)
non_central_f_cdf(1, 5, 2, 1)
non_central_f_lcdf(1, 5, 2, 1)
non_central_f_quantile(0.5, 5, 2, 1)
```

---

non_central_t_distribution

Noncentral T Distribution Functions

Description

Functions to compute the probability density function, cumulative distribution function, and quantile function for the Noncentral T distribution.

Usage

```
non_central_t_pdf(x, df, delta)

non_central_t_lpdf(x, df, delta)

non_central_t_cdf(x, df, delta)

non_central_t_lcdf(x, df, delta)

non_central_t_quantile(p, df, delta)
```

Arguments

`x`	quantile
`df`	degrees of freedom (df > 0)
`delta`	noncentrality parameter (delta >= 0)
`p`	probability (0 <= p <= 1)

Value

A single numeric value with the computed probability density, log-probability density, cumulative distribution, log-cumulative distribution, or quantile depending on the function called.

See Also

Boost Documentation for more details on the mathematical background.

Examples

```
# Noncentral T distribution with 3 degrees of freedom and noncentrality parameter 1
non_central_t_pdf(0, 3, 1)
non_central_t_lpdf(0, 3, 1)
non_central_t_cdf(0, 3, 1)
non_central_t_lcdf(0, 3, 1)
non_central_t_quantile(0.5, 3, 1)
```

`normal_distribution` *Normal Distribution Functions*

Description

Functions to compute the probability density function, cumulative distribution function, and quantile function for the Normal distribution.

Usage

```
normal_pdf(x, mean = 0, sd = 1)

normal_lpdf(x, mean = 0, sd = 1)

normal_cdf(x, mean = 0, sd = 1)

normal_lcdf(x, mean = 0, sd = 1)

normal_quantile(p, mean = 0, sd = 1)
```

Arguments

x	quantile
mean	mean parameter (default is 0)
sd	standard deviation parameter (default is 1)
p	probability (0 <= p <= 1)

Value

A single numeric value with the computed probability density, log-probability density, cumulative distribution, log-cumulative distribution, or quantile depending on the function called.

See Also

Boost Documentation for more details on the mathematical background.

Examples

```
# Normal distribution with mean = 0, sd = 1
normal_pdf(0)
normal_lpdf(0)
normal_cdf(0)
normal_lcdf(0)
normal_quantile(0.5)
```

---

number_series *Number Series*

Description

Functions to compute Bernoulli numbers, tangent numbers, fibonacci numbers, and prime numbers.

Usage

```
bernoulli_b2n(n = NULL, start_index = NULL, number_of_bernoullis_b2n = NULL)

max_bernoulli_b2n()

unchecked_bernoulli_b2n(n)

tangent_t2n(n = NULL, start_index = NULL, number_of_tangent_t2n = NULL)

prime(n)

max_prime()

fibonacci(n)

unchecked_fibonacci(n)
```

Arguments

`n`	Index of number to compute (must be a non-negative integer)
`start_index`	The starting index for the range of numbers (must be a non-negative integer)
`number_of_bernoullis_b2n`	The number of Bernoulli numbers to compute
`number_of_tangent_t2n`	The number of tangent numbers to compute

Details

Efficient computation of Bernoulli numbers, tangent numbers, fibonacci numbers, and prime numbers.

The `checked_` functions ensure that the input is within valid bounds, while the `unchecked_` functions do not perform such checks, allowing for potentially faster computation at the risk of overflow or invalid input.

The `max_` functions return the maximum index for which the respective numbers can be computed using precomputed lookup tables.

Value

A single numeric value for the Bernoulli numbers, tangent numbers, fibonacci numbers, or prime numbers, or a vector of values for ranges.

See Also

Boost Documentation for more details on the mathematical background.

Examples

```
bernoulli_b2n(10)
max_bernoulli_b2n()
unchecked_bernoulli_b2n(10)
bernoulli_b2n(start_index = 0, number_of_bernoullis_b2n = 10)
tangent_t2n(10)
tangent_t2n(start_index = 0, number_of_tangent_t2n = 10)
prime(10)
max_prime()
fibonacci(10)
unchecked_fibonacci(10)
```

`numerical_differentiation` *Numerical Differentiation*

Description

Functions for numerical differentiation using finite difference methods and complex step methods.

Usage

```
finite_difference_derivative(f, x, order = 1)

complex_step_derivative(f, x)
```

Arguments

`f`	A function to differentiate. It should accept a single numeric value and return a single numeric value.
`x`	The point at which to evaluate the derivative.
`order`	The order of accuracy of the derivative to compute. Default is 1.

Value

The approximate value of the derivative at the point `x`.

Examples

```
# Finite difference derivative of sin(x) at pi/4
finite_difference_derivative(sin, pi / 4)
# Complex step derivative of exp(x) at 1.7
complex_step_derivative(exp, 1.7)
```

`numerical_integration` *Numerical Integration*

Description

Functions for numerical integration using various methods such as trapezoidal rule, Gauss-Legendre quadrature, and Gauss-Kronrod quadrature.

Usage

```
trapezoidal(f, a, b, tol = sqrt(.Machine$double.eps), max_refinements = 12)

gauss_legendre(f, a, b, points = 7)

gauss_kronrod(
  f,
  a,
  b,
  points = 15,
  max_depth = 15,
  tol = sqrt(.Machine$double.eps)
)
```

Arguments

`f`	A function to integrate. It should accept a single numeric value and return a single numeric value.
`a`	The lower limit of integration.
`b`	The upper limit of integration.
`tol`	The tolerance for the approximation. Default is `sqrt(.Machine$double.eps)`.
`max_refinements`	The maximum number of refinements to apply. Default is 12.
`points`	The number of evaluation points to use in the Gauss-Legendre or Gauss-Kronrod quadrature.
`max_depth`	Sets the maximum number of interval splittings for Gauss-Kronrod permitted before stopping. Set this to zero for non-adaptive quadrature.

Value

A single numeric value with the computed integral.

Examples

```
# Trapezoidal rule integration of sin(x) from 0 to pi
trapezoidal(sin, 0, pi)
# Gauss-Legendre integration of exp(x) from 0 to 1
gauss_legendre(exp, 0, 1, points = 7)
# Adaptive Gauss-Kronrod integration of log(x) from 1 to 2
gauss_kronrod(log, 1, 2, points = 15, max_depth = 10)
```

`ooura_fourier_integrals` *Ooura Fourier Integrals*

Description

Computing Fourier sine and cosine integrals using Ooura's method.

Usage

```
ooura_fourier_sin(
  f,
  omega = 1,
  relative_error_tolerance = sqrt(.Machine$double.eps),
  levels = 8
)

ooura_fourier_cos(
  f,
  omega = 1,
  relative_error_tolerance = sqrt(.Machine$double.eps),
  levels = 8
)
```

Arguments

`f`	A function to integrate. It should accept a single numeric value and return a single numeric value.
`omega`	The frequency parameter for the sine integral.
`relative_error_tolerance`	The relative error tolerance for the approximation.
`levels`	The number of levels of refinement to apply. Default is 8.

Value

A single numeric value with the computed Fourier sine or cosine integral, with attribute 'relative_error' indicating the relative error of the approximation.

Examples

```
## Not run:
# Fourier sine integral of sin(x) with omega = 1
ooura_fourier_sin(function(x) { 1 / x }, omega = 1)
# Fourier cosine integral of cos(x) with omega = 1
ooura_fourier_cos(function(x) { 1/ (x * x + 1) }, omega = 1)

## End(Not run)
```

owens_t *Owens T Function*

Description

Computes the Owens T function of h and a, giving the probability of the event (X > h and 0 < Y < a * X) where X and Y are independent standard normal random variables.

Usage

```
owens_t(h, a)
```

Arguments

h	The first argument of the Owens T function
a	The second argument of the Owens T function

Value

The value of the Owens T function at (h, a).

See Also

Boost Documentation for more details on the mathematical background.

Examples

```
# Owens T Function
owens_t(1, 0.5)
```

pareto_distribution *Pareto Distribution Functions*

Description

Functions to compute the probability density function, cumulative distribution function, and quantile function for the Pareto distribution.

Usage

```
pareto_pdf(x, shape = 1, scale = 1)

pareto_lpdf(x, shape = 1, scale = 1)

pareto_cdf(x, shape = 1, scale = 1)

pareto_lcdf(x, shape = 1, scale = 1)

pareto_quantile(p, shape = 1, scale = 1)
```

Arguments

`x`	quantile
`shape`	shape parameter (default is 1)
`scale`	scale parameter (default is 1)
`p`	probability (0 <= p <= 1)

Value

A single numeric value with the computed probability density, log-probability density, cumulative distribution, log-cumulative distribution, or quantile depending on the function called.

See Also

Boost Documentation for more details on the mathematical background.

Examples

```
# Pareto distribution with shape = 1, scale = 1
pareto_pdf(1)
pareto_lpdf(1)
pareto_cdf(1)
pareto_lcdf(1)
pareto_quantile(0.5)
```

---

pchip *PCHIP Interpolator*

Description

Constructs a PCHIP interpolator given the vectors of abscissas, ordinates, and derivatives.

Usage

```
pchip(x, y, left_endpoint_derivative = NULL, right_endpoint_derivative = NULL)
```

Arguments

- `x` — Numeric vector of abscissas (x-coordinates).
- `y` — Numeric vector of ordinates (y-coordinates).
- `left_endpoint_derivative` — Optional numeric value of the derivative at the left endpoint.
- `right_endpoint_derivative` — Optional numeric value of the derivative at the right endpoint.

Value

An object of class `pchip` with methods:

- `interpolate(xi)`: Evaluate the interpolator at point `xi`.
- `prime(xi)`: Evaluate the derivative of the interpolator at point `xi`.
- `push_back(x, y)`: Add a new control point

Examples

```
x <- c(0, 1, 2, 3)
y <- c(0, 1, 0, 1)
interpolator <- pchip(x, y)
xi <- 0.5
interpolator$interpolate(xi)
interpolator$prime(xi)
interpolator$push_back(4, 1)
```

---

poisson_distribution *Poisson Distribution Functions*

Description

Functions to compute the probability density function, cumulative distribution function, and quantile function for the Poisson distribution.

Usage

```
poisson_pdf(x, lambda = 1)

poisson_lpdf(x, lambda = 1)

poisson_cdf(x, lambda = 1)

poisson_lcdf(x, lambda = 1)

poisson_quantile(p, lambda = 1)
```

Arguments

`x`	quantile
`lambda`	rate parameter (default is 1)
`p`	probability (0 <= p <= 1)

Value

A single numeric value with the computed probability density, log-probability density, cumulative distribution, log-cumulative distribution, or quantile depending on the function called.

See Also

Boost Documentation for more details on the mathematical background.

Examples

```
# Poisson distribution with lambda = 1
poisson_pdf(0, 1)
poisson_lpdf(0, 1)
poisson_cdf(0, 1)
poisson_lcdf(0, 1)
poisson_quantile(0.5, 1)
```

`polynomial_root_finding` *Polynomial Root-Finding*

Description

Functions for finding roots of polynomials of various degrees.

Usage

```
quadratic_roots(a, b, c)

cubic_roots(a, b, c, d)

cubic_root_residual(a, b, c, d, root)

cubic_root_condition_number(a, b, c, d, root)

quartic_roots(a, b, c, d, e)
```

Arguments

a	Coefficient of the polynomial term (e.g., for quadratic ax^2 + bx + c, a is the coefficient of x^2).
b	Coefficient of the linear term (e.g., for quadratic ax^2 + bx + c, b is the coefficient of x).
c	Constant term (e.g., for quadratic ax^2 + bx + c, c is the constant).
d	Coefficient of the cubic term (for cubic ax^3 + bx^2 + cx + d, d is the constant).
root	The root to evaluate the residual or condition number at.
e	Coefficient of the quartic term (for quartic ax^4 + bx^3 + cx^2 + dx + e, e is the constant).

Details

This package provides functions to find roots of quadratic, cubic, and quartic polynomials. The functions return the roots as numeric vectors.

Value

A numeric vector of the polynomial roots, residual, or condition number.

Examples

```
# Example of finding quadratic roots
quadratic_roots(1, -3, 2)
# Example of finding cubic roots
cubic_roots(1, -6, 11, -6)
# Example of finding quartic roots
quartic_roots(1, -10, 35, -50, 24)
# Example of finding cubic root residual
cubic_root_residual(1, -6, 11, -6, 1)
# Example of finding cubic root condition number
cubic_root_condition_number(1, -6, 11, -6, 1)
```

---

quintic_hermite *Quintic Hermite Interpolator*

Description

Constructs a quintic Hermite interpolator given the vectors of abscissas, ordinates, first derivatives, and second derivatives.

Usage

```
quintic_hermite(x, y, dydx, d2ydx2)
```

Arguments

`x`	Numeric vector of abscissas (x-coordinates).
`y`	Numeric vector of ordinates (y-coordinates).
`dydx`	Numeric vector of first derivatives (slopes) at each point.
`d2ydx2`	Numeric vector of second derivatives at each point.

Value

An object of class `quintic_hermite` with methods:

- `interpolate(xi)`: Evaluate the interpolator at point `xi`.
- `prime(xi)`: Evaluate the derivative of the interpolator at point `xi`.
- `double_prime(xi)`: Evaluate the second derivative of the interpolator at point `xi`.
- `push_back(x, y, dydx, d2ydx2)`: Add a new control point to the interpolator.
- `domain()`: Get the domain of the interpolator.

Examples

```
x <- c(0, 1, 2)
y <- c(0, 1, 0)
dydx <- c(1, 0, -1)
d2ydx2 <- c(0, -1, 0)
interpolator <- quintic_hermite(x, y, dydx, d2ydx2)
xi <- 0.5
interpolator$interpolate(xi)
interpolator$prime(xi)
interpolator$double_prime(xi)
interpolator$push_back(3, 0, 1, 0)
interpolator$domain()
```

`rayleigh_distribution` *Rayleigh Distribution Functions*

Description

Functions to compute the probability density function, cumulative distribution function, and quantile function for the Rayleigh distribution.

Usage

```
rayleigh_pdf(x, scale = 1)

rayleigh_lpdf(x, scale = 1)

rayleigh_cdf(x, scale = 1)
```

```
rayleigh_lcdf(x, scale = 1)

rayleigh_quantile(p, scale = 1)
```

Arguments

x	quantile
scale	scale parameter (default is 1)
p	probability (0 <= p <= 1)

Value

A single numeric value with the computed probability density, log-probability density, cumulative distribution, log-cumulative distribution, or quantile depending on the function called.

See Also

Boost Documentation for more details on the mathematical background.

Examples

```
# Rayleigh distribution with scale = 1
rayleigh_pdf(1)
rayleigh_lpdf(1)
rayleigh_cdf(1)
rayleigh_lcdf(1)
rayleigh_quantile(0.5)
```

---

rootfinding_and_minimisation

Root-Finding and Minimisation Functions

Description

Functions for root-finding and minimisation using various algorithms.

Usage

```
bisect(
  f,
  lower,
  upper,
  digits = .Machine$double.digits,
  max_iter = .Machine$integer.max
)

bracket_and_solve_root(
  f,
```

```
  guess,
  factor,
  rising,
  digits = .Machine$double.digits,
  max_iter = .Machine$integer.max
)

toms748_solve(
  f,
  lower,
  upper,
  digits = .Machine$double.digits,
  max_iter = .Machine$integer.max
)

newton_raphson_iterate(
  f,
  guess,
  lower,
  upper,
  digits = .Machine$double.digits,
  max_iter = .Machine$integer.max
)

halley_iterate(
  f,
  guess,
  lower,
  upper,
  digits = .Machine$double.digits,
  max_iter = .Machine$integer.max
)

schroder_iterate(
  f,
  guess,
  lower,
  upper,
  digits = .Machine$double.digits,
  max_iter = .Machine$integer.max
)

brent_find_minima(
  f,
  lower,
  upper,
  digits = .Machine$double.digits,
  max_iter = .Machine$integer.max
```

```
)
```

Arguments

f	A function to find the root of or to minimise. It should take and return a single numeric value for root-finding, or a numeric vector for minimisation.
lower	The lower bound of the interval to search for the root or minimum.
upper	The upper bound of the interval to search for the root or minimum.
digits	The number of significant digits to which the root or minimum should be found. Defaults to double precision.
max_iter	The maximum number of iterations to perform. Defaults to the maximum integer value.
guess	A numeric value that is a guess for the root or minimum.
factor	Size of steps to take when searching for the root.
rising	If TRUE, the function is assumed to be rising, otherwise it is assumed to be falling.

Details

This package provides a set of functions for finding roots of equations and minimising functions using different numerical methods. The methods include bisection, bracket and solve, TOMS 748, Newton-Raphson, Halley's method, Schroder's method, and Brent's method. It also includes functions for finding roots of polynomials (quadratic, cubic, quartic) and computing minima.

Value

A list containing the root or minimum value, the value of the function at that point, and the number of iterations performed.

See Also

Boost Documentation for more details on the mathematical background.

Examples

```
f <- function(x) x^2 - 2
bisect(f, lower = 0, upper = 2)
bracket_and_solve_root(f, guess = 1, factor = 0.1, rising = TRUE)
toms748_solve(f, lower = 0, upper = 2)
f <- function(x) c(x^2 - 2, 2 * x)
newton_raphson_iterate(f, guess = 1, lower = 0, upper = 2)
f <- function(x) c(x^2 - 2, 2 * x, 2)
halley_iterate(f, guess = 1, lower = 0, upper = 2)
schroder_iterate(f, guess = 1, lower = 0, upper = 2)
f <- function(x) (x - 2)^2 + 1
brent_find_minima(f, lower = 0, upper = 4)
```

runs_tests *Runs Tests*

Description

Functions for Runs Tests.

Usage

```
runs_above_and_below_threshold(v, threshold)

runs_above_and_below_median(v)
```

Arguments

`v`	A numeric vector.
`threshold`	A single numeric value.

Value

A two-element numeric vector containing the t-statistic and the p-value.

See Also

Boost Documentation for more details on the mathematical background.

Examples

```
# Runs Above and Below Threshold
runs_above_and_below_threshold(c(1, 2, 3, 4, 5), threshold = 3)
# Runs Above and Below Median
runs_above_and_below_median(c(1, 2, 3, 4, 5))
```

saspoint5_distribution *SαS Point5 Distribution Functions*

Description

Functions to compute the probability density function, cumulative distribution function, and quantile function for the SαS Point5 distribution.

Usage

```
saspoint5_pdf(x, location = 0, scale = 1)

saspoint5_lpdf(x, location = 0, scale = 1)

saspoint5_cdf(x, location = 0, scale = 1)

saspoint5_lcdf(x, location = 0, scale = 1)

saspoint5_quantile(p, location = 0, scale = 1)
```

Arguments

x	quantile
location	location parameter (default is 0)
scale	scale parameter (default is 1)
p	probability (0 <= p <= 1)

Value

A single numeric value with the computed probability density, log-probability density, cumulative distribution, log-cumulative distribution, or quantile depending on the function called.

See Also

Boost Documentation for more details on the mathematical background.

Examples

```
# SaS Point5 distribution with location 0 and scale 1
saspoint5_pdf(3)
saspoint5_lpdf(3)
saspoint5_cdf(3)
saspoint5_lcdf(3)
saspoint5_quantile(0.5)
```

---

signal_statistics *Signal Statistics Functions*

Description

Functions to compute various signal statistics.

Usage

```
absolute_gini_coefficient(x)

sample_absolute_gini_coefficient(x)

hoyer_sparsity(x)

oracle_snr(signal, noisy_signal)

oracle_snr_db(signal, noisy_signal)

m2m4_snr_estimator(noisy_signal, signal_kurtosis = 1, noise_kurtosis = 3)

m2m4_snr_estimator_db(noisy_signal, signal_kurtosis = 1, noise_kurtosis = 3)
```

Arguments

`x`	A numeric vector.
`signal`	A numeric vector.
`noisy_signal`	A numeric vector.
`signal_kurtosis`	A single numeric value.
`noise_kurtosis`	A single numeric value.

Value

A numeric value or vector with the computed statistic.

See Also

Boost Documentation for more details on the mathematical background.

Examples

```
# Absolute Gini Coefficient
absolute_gini_coefficient(c(1, 2, 3, 4, 5))
# Sample Absolute Gini Coefficient
sample_absolute_gini_coefficient(c(1, 2, 3, 4, 5))
# Hoyer Sparsity
hoyer_sparsity(c(1, 0, 0, 2, 3))

signal <- c(1, 2, 3)
noisy_signal <- c(1.1, 2.1, 3.1)
# Oracle SNR
oracle_snr(signal, noisy_signal)
# Oracle SNR in dB
oracle_snr_db(signal, noisy_signal)
# M2M4 SNR Estimator
m2m4_snr_estimator(noisy_signal, 3, 2)
```

```
# M2M4 SNR Estimator in dB
m2m4_snr_estimator_db(noisy_signal, 3, 2)
```

---

sinus_cardinal_hyperbolic_functions
Sinus Cardinal and Hyperbolic Functions

Description

Functions to compute the sinus cardinal function and hyperbolic sinus cardinal function.

Usage

```
sinc_pi(x)

sinhc_pi(x)
```

Arguments

x	Input value

Value

A single numeric value with the computed sinus cardinal or hyperbolic sinus cardinal function.

See Also

Boost Documentation for more details on the mathematical background.

Examples

```
# Sinus cardinal function
sinc_pi(0.5)
# Hyperbolic sinus cardinal function
sinhc_pi(0.5)
```

skew_normal_distribution
Skew Normal Distribution Functions

Description

Functions to compute the probability density function, cumulative distribution function, and quantile function for the Skew Normal distribution.

Usage

```
skew_normal_pdf(x, location = 0, scale = 1, shape = 0)

skew_normal_lpdf(x, location = 0, scale = 1, shape = 0)

skew_normal_cdf(x, location = 0, scale = 1, shape = 0)

skew_normal_lcdf(x, location = 0, scale = 1, shape = 0)

skew_normal_quantile(p, location = 0, scale = 1, shape = 0)
```

Arguments

x	quantile
location	location parameter (default is 0)
scale	scale parameter (default is 1)
shape	shape parameter (default is 0)
p	probability (0 <= p <= 1)

Value

A single numeric value with the computed probability density, log-probability density, cumulative distribution, log-cumulative distribution, or quantile depending on the function called.

See Also

Boost Documentation for more details on the mathematical background.

Examples

```
# Skew Normal distribution with location = 0, scale = 1, shape = 0
skew_normal_pdf(0)
skew_normal_lpdf(0)
skew_normal_cdf(0)
skew_normal_lcdf(0)
skew_normal_quantile(0.5)
```

spherical_harmonics *Spherical Harmonics*

Description

Functions to compute spherical harmonics and related functions.

Usage

```
spherical_harmonic(n, m, theta, phi)

spherical_harmonic_r(n, m, theta, phi)

spherical_harmonic_i(n, m, theta, phi)
```

Arguments

n	Degree of the spherical harmonic
m	Order of the spherical harmonic
theta	Polar angle (colatitude)
phi	Azimuthal angle (longitude)

Value

A single complex value with the computed spherical harmonic function, or its real and imaginary parts.

See Also

Boost Documentation

Examples

```
# Spherical harmonic function Y_2^1(0.5, 0.5)
spherical_harmonic(2, 1, 0.5, 0.5)
# Real part of the spherical harmonic function Y_2^1(0.5, 0.5)
spherical_harmonic_r(2, 1, 0.5, 0.5)
# Imaginary part of the spherical harmonic function Y_2^1(0.5, 0.5)
spherical_harmonic_i(2, 1, 0.5, 0.5)
```

students_t_distribution

Student's T Distribution Functions

Description

Functions to compute the probability density function, cumulative distribution function, and quantile function for the Student's t distribution.

Usage

```
students_t_pdf(x, df = 1)

students_t_lpdf(x, df = 1)

students_t_cdf(x, df = 1)

students_t_lcdf(x, df = 1)

students_t_quantile(p, df = 1)
```

Arguments

x	quantile
df	degrees of freedom (default is 1)
p	probability (0 <= p <= 1)

Value

A single numeric value with the computed probability density, log-probability density, cumulative distribution, log-cumulative distribution, or quantile depending on the function called.

See Also

Boost Documentation for more details on the mathematical background.

Examples

```
# Student's t distribution with 3 degrees of freedom
students_t_pdf(0, 3)
students_t_lpdf(0, 3)
students_t_cdf(0, 3)
students_t_lcdf(0, 3)
students_t_quantile(0.5, 3)
```

`triangular_distribution` *Triangular Distribution Functions*

Description

Functions to compute the probability density function, cumulative distribution function, and quantile function for the Triangular distribution.

Usage

```
triangular_pdf(x, lower = 0, mode = 1, upper = 2)

triangular_lpdf(x, lower = 0, mode = 1, upper = 2)

triangular_cdf(x, lower = 0, mode = 1, upper = 2)

triangular_lcdf(x, lower = 0, mode = 1, upper = 2)

triangular_quantile(p, lower = 0, mode = 1, upper = 2)
```

Arguments

x	quantile
lower	lower limit of the distribution (default is 0)
mode	mode of the distribution (default is 1)
upper	upper limit of the distribution (default is 2)
p	probability (0 <= p <= 1)

Value

A single numeric value with the computed probability density, log-probability density, cumulative distribution, log-cumulative distribution, or quantile depending on the function called.

See Also

Boost Documentation for more details on the mathematical background.

Examples

```
# Triangular distribution with lower = 0, mode = 1, upper = 2
triangular_pdf(1)
triangular_lpdf(1)
triangular_cdf(1)
triangular_lcdf(1)
triangular_quantile(0.5)
```

t_tests *T-Tests*

Description

Functions for T - Tests.

Usage

```
one_sample_t_test_params(
  sample_mean,
  sample_variance,
  num_samples,
  assumed_mean
)

one_sample_t_test(u, assumed_mean)

two_sample_t_test(u, v)

paired_samples_t_test(u, v)
```

Arguments

`sample_mean`	A single value.
`sample_variance`	A single value.
`num_samples`	A single value.
`assumed_mean`	A single value.
`u`	A numeric vector.
`v`	A numeric vector.

Value

A two-element numeric vector containing the t-statistic and the p-value.

See Also

Boost Documentation for more details on the mathematical background.

Examples

```
# One Sample T-Test Parameters
one_sample_t_test_params(sample_mean = 2, sample_variance = 1, num_samples = 30, assumed_mean = 0)
# One Sample T-Test
one_sample_t_test(c(5, 6, 7), assumed_mean = 4)
# Two Sample T-Test
```

```
two_sample_t_test(c(5, 6, 7), c(4, 5, 6))
# Paired Samples T-Test
paired_samples_t_test(c(5, 6, 7), c(4, 5, 6))
```

`uniform_distribution` *Uniform Distribution Functions*

Description

Functions to compute the probability density function, cumulative distribution function, and quantile function for the Uniform distribution.

Usage

```
uniform_pdf(x, lower = 0, upper = 1)

uniform_lpdf(x, lower = 0, upper = 1)

uniform_cdf(x, lower = 0, upper = 1)

uniform_lcdf(x, lower = 0, upper = 1)

uniform_quantile(p, lower = 0, upper = 1)
```

Arguments

x	quantile
lower	lower bound of the distribution (default is 0)
upper	upper bound of the distribution (default is 1)
p	probability (0 <= p <= 1)

Value

A single numeric value with the computed probability density, log-probability density, cumulative distribution, log-cumulative distribution, or quantile depending on the function called.

See Also

Boost Documentation for more details on the mathematical background.

Examples

```
# Uniform distribution with lower = 0, upper = 1
uniform_pdf(0.5)
uniform_lpdf(0.5)
uniform_cdf(0.5)
uniform_lcdf(0.5)
uniform_quantile(0.5)
```

univariate_statistics *Univariate Statistics Functions*

Description

Functions to compute various univariate statistics.

Usage

```
mean_boost(x)

variance(x)

sample_variance(x)

mean_and_sample_variance(x)

skewness(x)

kurtosis(x)

excess_kurtosis(x)

first_four_moments(x)

median_boost(x)

median_absolute_deviation(x)

interquartile_range(x)

gini_coefficient(x)

sample_gini_coefficient(x)

mode_boost(x)
```

Arguments

x	A numeric vector.

Value

A numeric value or vector with the computed statistic.

See Also

Boost Documentation for more details on the mathematical background.

Examples

```
# Mean
mean_boost(c(1, 2, 3, 4, 5))
# Variance
variance(c(1, 2, 3, 4, 5))
# Sample Variance
sample_variance(c(1, 2, 3, 4, 5))
# Mean and Sample Variance
mean_and_sample_variance(c(1, 2, 3, 4, 5))
# Skewness
skewness(c(1, 2, 3, 4, 5))
# Kurtosis
kurtosis(c(1, 2, 3, 4, 5))
# Excess Kurtosis
excess_kurtosis(c(1, 2, 3, 4, 5))
# First Four Moments
first_four_moments(c(1, 2, 3, 4, 5))
# Median
median_boost(c(1, 2, 3, 4, 5))
# Median Absolute Deviation
median_absolute_deviation(c(1, 2, 3, 4, 5))
# Interquartile Range
interquartile_range(c(1, 2, 3, 4, 5))
# Gini Coefficient
gini_coefficient(c(1, 2, 3, 4, 5))
# Sample Gini Coefficient
sample_gini_coefficient(c(1, 2, 3, 4, 5))
# Mode
mode_boost(c(1, 2, 2, 3, 4))
```

---

vector_functionals   *Vector Functionals*

Description

Functions to compute various vector norms and distances.

Usage

```
l0_pseudo_norm(x)

hamming_distance(x, y)

l1_norm(x)

l1_distance(x, y)

l2_norm(x)
```

```
l2_distance(x, y)

sup_norm(x)

sup_distance(x, y)

lp_norm(x, p)

lp_distance(x, y, p)

total_variation(x)
```

Arguments

x	A numeric vector.
y	A numeric vector of the same length as x (for distance functions).
p	A positive integer indicating the order of the norm or distance (for Lp functions).

Value

A single numeric value with the computed norm or distance.

See Also

Boost Documentation for more details on the mathematical background.

Examples

```
# L0 Pseudo Norm
l0_pseudo_norm(c(1, 0, 2, 0, 3))
# Hamming Distance
hamming_distance(c(1, 0, 1), c(0, 1, 1))
# L1 Norm
l1_norm(c(1, -2, 3))
# L1 Distance
l1_distance(c(1, -2, 3), c(4, -5, 6))
# L2 Norm
l2_norm(c(3, 4))
# L2 Distance
l2_distance(c(3, 4), c(0, 0))
# Supremum Norm
sup_norm(c(1, -2, 3))
# Supremum Distance
sup_distance(c(1, -2, 3), c(4, -5, 6))
# Lp Norm
lp_norm(c(1, -2, 3), 3)
# Lp Distance
lp_distance(c(1, -2, 3), c(4, -5, 6), 3)
# Total Variation
total_variation(c(1, 2, 1, 3))
```

weibull_distribution *Weibull Distribution Functions*

Description

Functions to compute the probability density function, cumulative distribution function, and quantile function for the Weibull distribution.

Usage

```
weibull_pdf(x, shape = 1, scale = 1)

weibull_lpdf(x, shape = 1, scale = 1)

weibull_cdf(x, shape = 1, scale = 1)

weibull_lcdf(x, shape = 1, scale = 1)

weibull_quantile(p, shape = 1, scale = 1)
```

Arguments

x	quantile
shape	shape parameter (default is 1)
scale	scale parameter (default is 1)
p	probability (0 <= p <= 1)

Value

A single numeric value with the computed probability density, log-probability density, cumulative distribution, log-cumulative distribution, or quantile depending on the function called.

See Also

Boost Documentation for more details on the mathematical background.

Examples

```
# Weibull distribution with shape = 1, scale = 1
weibull_pdf(1)
weibull_lpdf(1)
weibull_cdf(1)
weibull_lcdf(1)
weibull_quantile(0.5)
```

zeta — *Riemann Zeta Function*

Description

Computes the Riemann zeta function $(\zeta(s))$ for argument (z).

Usage

```
zeta(z)
```

Arguments

`z`	Real number input

Value

The value of the Riemann zeta function $(\zeta(z))$.

Examples

```
# Riemann Zeta Function
zeta(2)  # Should return pi^2 / 6
```

z_tests — *Z-Tests*

Description

Functions for Z - Tests.

Usage

```
one_sample_z_test_params(
  sample_mean,
  sample_variance,
  num_samples,
  assumed_mean
)

one_sample_z_test(u, assumed_mean)

two_sample_z_test(u, v)
```

Arguments

`sample_mean`	A single value.
`sample_variance`	A single value.
`num_samples`	A single value.
`assumed_mean`	A single value.
`u`	A numeric vector.
`v`	A numeric vector.

Value

A two-element numeric vector containing the t-statistic and the p-value.

See Also

Boost Documentation for more details on the mathematical background.

Examples

```
# One Sample T-Test Parameters
one_sample_z_test_params(sample_mean = 2, sample_variance = 1, num_samples = 30, assumed_mean = 0)
# One Sample T-Test
one_sample_z_test(c(5, 6, 7), assumed_mean = 4)
# Two Sample T-Test
two_sample_z_test(c(5, 6, 7), c(4, 5, 6))
```

Index