Package ‘cbbinom’

September 2, 2024

Title Continuous Analog of a Beta-Binomial Distribution

Version 0.1.0

Description
Implementation of the d/p/q/r family of functions for a continuous analog to the standard discrete beta-binomial with continuous size parameter and continuous support with x in [0, size + 1].

License MIT + file LICENSE

Suggests extraDistr, ggplot2, testthat (>= 3.0.0)

Config/testthat/edition 3

Encoding UTF-8

RoxygenNote 7.3.2

LinkingTo Rcpp

Imports Rcpp

NeedsCompilation yes

Author Xiurui Zhu [aut, cre]

Maintainer Xiurui Zhu <zxr6@163.com>

Repository CRAN

Date/Publication 2024-09-02 12:00:08 UTC

Contents

cbbinom *The Continuous Beta-Binomial Distribution*

Description

Density, distribution function, quantile function and random generation for a continuous analog to the beta-binomial distribution with parameters `size` `alpha` and `beta`. The usage and help pages are modeled on the d-p-q-r families of functions for the commonly-used distributions in the `stats` package.

Usage

```
dcbbinom(
  x,
  size,
  alpha = 1,
  beta = 1,
  ncp = 0,
  log = FALSE,
  tol = 1e-06,
  max_iter = 10000L
)

pcbbinom(
  q,
  size,
  alpha = 1,
  beta = 1,
  ncp = 0,
  lower.tail = TRUE,
  log.p = FALSE,
  tol = 1e-06,
  max_iter = 10000L
)

qcbbinom(
  p,
  size,
  alpha = 1,
  beta = 1,
  ncp = 0,
  lower.tail = TRUE,
  log.p = FALSE,
  p_tol = 1e-06,
  p_max_iter = 10000L,
  root_tol = 1e-06,
  root_max_iter = 10000L
```

```
)

rcbbinom(
  n,
  size,
  alpha = 1,
  beta = 1,
  ncp = 0,
  p_tol = 1e-06,
  p_max_iter = 10000L,
  root_tol = 1e-06,
  root_max_iter = 10000L
)
```

Arguments

`x, q`	vector of quantiles.
`size`	number of trials (zero or more).
`alpha, beta`	non-negative parameters of the Beta distribution.
`ncp`	non-centrality parameter.
`log, log.p`	logical; if TRUE, probabilities p are given as log(p).
`tol, max_iter`	arguments passed on to `gen_hypergeo`.
`lower.tail`	logical; if TRUE (default), probabilities are $P[X \le x]$, otherwise, $P[X > x]$.
`p`	vector of probabilities.
`p_tol, p_max_iter`	same as `tol, max_iter`.
`root_tol, root_max_iter`	arguments passed on to `uniroot`.
`n`	number of observations. If `length(n) > 1`, the length is taken to be the number required.

Details

Derived from the continuous binomial distribution (Ilienko 2013), the continuous beta-binomial distribution is defined as:

$$P(x|n,\alpha,\beta) = \int_0^1 \frac{B_{1-p}(n+1-x,x)}{B(n+1-x,x)} \frac{p^{\alpha-1}(1-p)^{\beta-1}}{B(\alpha,\beta)} dp,$$

where x is the quantile, n is the size, $B_p(a,b) = \int_0^p u^{a-1}(1-u)^{b-1} du$ is the incomplete beta function.

When simplified, the distribution becomes:

$$P(x|n,\alpha,\beta) = \frac{\Gamma(n+1)B(n+1-x+\beta,\alpha)}{\Gamma(x)\Gamma(n+2-x)B(\alpha,\beta)} {}_3F_2(a;b;z),$$

where ${}_3F_2(a;b;z)$ is generalized hypergeometric function, $a = \{1-x, n+1-x, n+1-x+\beta\}$, $b = \{n+2-x, n+1-x+\alpha+\beta\}$, $z = 1$.

Heuristically speaking, this distribution spreads the standard probability mass at integer `x` to the interval `[x, x + 1]` in a continuous manner. As a result, the distribution looks like a smoothed version of the standard, discrete beta-binomial but shifted slightly to the right. The support of the continuous beta-binomial is `[0, size + 1]`, and the mean is approximately `size * alpha / (alpha + beta) + 1/2`.

Supplying `ncp` moves the support of beta-binomial to `[ncp, size + 1 + ncp]`, e.g. for the continuous beta-binomial with non-shifted mean, use `ncp = -0.5`.

Value

`dcbbinom` gives the density, `pcbbinom` the distribution function, `qcbbinom` the quantile function, and `rcbbinom` generates random deviates.

Invalid arguments will result in return value `NaN`, with a warning.

The length of the result is determined by `n` for `rcbbinom`, and is the maximum of the lengths of the numerical arguments for the other functions.

The numerical arguments other than `n` are recycled to the length of the result. Only the first elements of the logical arguments are used.

Numerical computation of the density function

For simplicity, the density function is computed numerically through differentiation. To achieve higher numerical resolution (given that $d \ln u / du > 1, 0 < u < 1$), it is computed as:

$$p(x|n, \alpha, \beta) = \frac{\partial P(x|n, \alpha, \beta)}{\partial x} = \frac{\partial \exp[\ln P(x|n, \alpha, \beta)]}{\partial x}$$

When simplified, it becomes:

$$p(x|n, \alpha, \beta) = \frac{\partial \exp[\ln P(x|n, \alpha, \beta)]}{\partial \ln P(x|n, \alpha, \beta)} \frac{\partial \ln P(x|n, \alpha, \beta)}{\partial x} = \frac{\partial \ln P(x|n, \alpha, \beta)}{\partial x} P(x|n, \alpha, \beta),$$

where the first term is computed numerically and the second term is the distribution function.

Note

Change log:

- 0.1.0 Xiurui Zhu - Initiate the function.

References

Ilienko, Andreii (2013). Continuous counterparts of Poisson and binomial distributions and their properties. Annales Univ. Sci. Budapest., Sect. Comp. 39: 137-147. http://ac.inf.elte.hu/Vol_039_2013/137_39.pdf

Examples

```
# Density function
dcbbinom(x = 5, size = 10, alpha = 2, beta = 4)
# Distribution function
(test_val <- pcbbinom(q = 5, size = 10, alpha = 2, beta = 4))
# Quantile function
qcbbinom(p = test_val, size = 10, alpha = 2, beta = 4)
# Random generation
set.seed(1111L)
rcbbinom(n = 10L, size = 10, alpha = 2, beta = 4)
```

---

gen_hypergeo *Generalized hypergeometric function*

Description

gen_hypergeo computes generalized hypergeometric function.

Usage

```
gen_hypergeo(U, L, x, tol, max_iter, check_mode, log)
```

Arguments

U, L	Numeric vectors for upper and lower values.
x	Numeric (1L) as common ratio.
tol	Numeric (1L) as convergence tolerance.
max_iter	Integer (1L) as iteration limit.
check_mode	Logical (1L) indicating whether the mode of x should be checked for obvious convergence failures.
log	Logical (1L) indicating whether result is given as log(result).

Value

Result of computation. Warnings are issued if failing to converge.

Note

Change log:

- 0.1.0 Xiurui Zhu - Initiate the function.

Author(s)

Xiurui Zhu

Examples

```
gen_hypergeo(U = c(1.1, 0.2, 0.3), L = c(10.1, 4 * pi), x = 1,
             max_iter = 10000L, tol = 1e-6, check_mode = TRUE, log = FALSE)
```

Index