Package 'nbconv'

May 9, 2026

Title Evaluate Arbitrary Negative Binomial Convolutions

Version 1.0.1

URL https://github.com/gbedwell/nbconv

BugReports https://github.com/gbedwell/nbconv/issues

Imports parallel, matrixStats, stats

Description Three distinct methods are implemented for evaluating the sums of arbitrary negative binomial distributions. These methods are: Furman's exact probability mass function (Furman (2007) <doi:10.1016/j.spl.2006.06.007>), saddlepoint approximation, and a method of moments approximation. Functions are provided to calculate the density function, the distribution function and the quantile function of the convolutions in question given said evaluation methods. Functions for generating random deviates from negative binomial convolutions and for directly calculating the mean, variance, skewness, and excess kurtosis of said convolutions are also provided.

Encoding UTF-8

RoxygenNote 7.2.0

License GPL (>= 3)

NeedsCompilation no

Author Gregory Bedwell [aut, cre, cph] (ORCID:
<https://orcid.org/0000-0003-0456-0032>)

Maintainer Gregory Bedwell <gregoryjbedwell@gmail.com>

Repository CRAN

Date/Publication 2023-07-06 18:20:02 UTC

Contents

---

dnbconv *Probability mass function*

Description

Calculates the PMF for the convolution of arbitrary negative binomial random variables.

Usage

```
dnbconv(
  counts,
  mus,
  ps,
  phis,
  method = c("exact", "moments", "saddlepoint"),
  n.terms = 1000,
  n.cores = 1,
  tolerance = 0.001,
  normalize = TRUE
)
```

Arguments

counts	The counts over which the convolution is evaluated. Should be a vector.
mus	Vector of individual mean values
ps	Vector of individual probabilities of success.
phis	Vector of individual dispersion parameters. Equivalent to 'size' in dnbinom.
method	The method by which to evaluate the PMF. One of "exact", "moments", or "saddlepoint".
n.terms	The number of terms to include in the series for evaluating the PMF at a given number of counts. Defaults to 1000.
n.cores	The number of CPU cores to use in the evaluation. Allows parallelization.
tolerance	The acceptable difference between the sum of the K distribution and 1.
normalize	Boolean. If TRUE, the PMF is normalized to sum to 1.

Value

A numeric vector of probability densities.

Examples

```
dnbconv(counts = 0:500, mus = c(100, 10), phis = c(5, 8), method = "exact")
```

nbconv_params *Summary statistics*

Description

Calculates distribution parameters for the convolution of arbitrary negative binomial random variables.

Usage

```
nbconv_params(mus, phis, ps)
```

Arguments

mus	Vector of individual mean values
phis	Vector of individual dispersion parameters. Equivalent to 'size' in dnbinom.
ps	Vector of individual probabilities of success.

Value

A named numeric vector of distribution parameters.

Examples

```
nbconv_params(mus = c(100, 10), phis = c(5, 8))
```

nb_sum_exact *Furman's PMF*

Description

Implements Furman's exact PMF for the evaluation of the sum of arbitrary NB random variables. Called by other functions. Not intended to be run alone.

Usage

```
nb_sum_exact(phis, ps, n.terms = 1000, counts, n.cores = 1, tolerance = 0.001)
```

Arguments

phis	Vector of individual dispersion parameters. Equivalent to 'size' in dnbinom.
ps	Vector of individual probabilities of success.
n.terms	The number of terms to include in the series for evaluating the PMF at a given number of counts. Defaults to 1000.
counts	The vector of counts over which the PMF is evaluated.
n.cores	The number of CPU cores to use in the evaluation. Allows parallelization.
tolerance	The acceptable difference between the sum of the K distribution and 1.

Value

A numeric vector of probability densities.

Examples

```
nb_sum_exact(ps = c(0.05, 0.44), phis = c(5, 8), counts = 0:500)
```

---

nb_sum_moments *Method of moments*

Description

Implements the method of moments approximation for the sum of arbitrary NB random variables. Called by other functions. Not intended to be run alone.

Usage

```
nb_sum_moments(mus, phis, counts)
```

Arguments

mus	Vector of individual mean values.
phis	Vector of individual dispersion parameters. Equivalent to 'size' in dnbinom.
counts	The vector of counts over which the PMF is evaluated.

Value

A numeric vector of probability densities.

Examples

```
nb_sum_moments(mus = c(100, 10), phis = c(5, 8), counts = 0:500)
```

nb_sum_saddlepoint *Saddlepoint approximation*

Description

Implements the saddlepoint approximation for the sum of arbitrary NB random variables. Called by other functions. Not intended to be run alone.

Usage

```
nb_sum_saddlepoint(mus, phis, counts, normalize = TRUE, n.cores = 1)
```

Arguments

mus	Vector of individual mean values.
phis	Vector of individual dispersion parameters. Equivalent to 'size' in dnbinom.
counts	The vector of counts over which the PMF is evaluated.
normalize	Boolean. If TRUE, the PMF is normalized to sum to 1.
n.cores	The number of CPU cores to use in the evaluation. Allows parallelization.

Details

Inspired by https://www.martinmodrak.cz/2019/06/20/approximate-densities-for-sums-of-variables-negative-binomials-and-saddlepoint/

Value

A numeric vector of probability densities.

Examples

```
nb_sum_saddlepoint(mus = c(100, 10), phis = c(5, 8), counts = 0:500)
```

pnbconv *Cumulative distribution function*

Description

Calculates the CDF for the convolution of arbitrary negative binomial random variables.

Usage

```
pnbconv(
  quants,
  mus,
  ps,
  phis,
  method = c("exact", "moments", "saddlepoint"),
  n.terms = 1000,
  n.cores = 1,
  tolerance = 0.001,
  normalize = TRUE
)
```

Arguments

quants	Vector of quantiles.
mus	Vector of individual mean values
ps	Vector of individual probabilities of success.
phis	Vector of individual dispersion parameters. Equivalent to 'size' in dnbinom.
method	The method by which to evaluate the PMF. One of "exact", "moments", or "saddlepoint".
n.terms	The number of terms to include in the series for evaluating the PMF at a given number of counts. Defaults to 1000.
n.cores	The number of CPU cores to use in the evaluation. Allows parallelization.
tolerance	The acceptable difference between the sum of the K distribution and 1.
normalize	Boolean. If TRUE, the PMF is normalized to sum to 1.

Value

A numeric vector of cumulative probability densities.

Examples

```
pnbconv(quants = 200, mus = c(100, 10), phis = c(5, 8), method = "exact")
```

---

qnbconv *Quantile function*

Description

Calculates the quantile function for the convolution of arbitrary negative binomial random variables.

Usage

```
qnbconv(
  probs,
  counts,
  mus,
  ps,
  phis,
  method = c("exact", "moments", "saddlepoint"),
  n.terms = 1000,
  n.cores = 1,
  tolerance = 0.001,
  normalize = TRUE
)
```

Arguments

probs	Vector of target (cumulative) probabilities.
counts	Vector of counts over which the PMF is evaluated.
mus	Vector of individual mean values
ps	Vector of individual probabilities of success.
phis	Vector of individual dispersion parameters. Equivalent to 'size' in dnbinom.
method	The method by which to evaluate the PMF. One of "exact", "moments", or "saddlepoint".
n.terms	The number of terms to include in the series for evaluating the PMF at a given number of counts. Defaults to 1000.
n.cores	The number of CPU cores to use in the evaluation. Allows parallelization.
tolerance	The acceptable difference between the sum of the K distribution and 1.
normalize	Boolean. If TRUE, the PMF is normalized to sum to 1.

Value

A numeric vector of quantiles.

Examples

```
qnbconv(probs = c(0.05, 0.25, 0.5, 0.75, 0.95), counts = 0:500,
       mus = c(100, 10), phis = c(5, 8), method = "exact")
```

rnbconv *Random deviates*

Description

Generates random samples from the convolution of arbitrary negative binomial random variables.

Usage

```
rnbconv(mus, phis, ps, n.samp, n.cores = 1)
```

Arguments

mus	Vector of individual mean values
phis	Vector of individual dispersion parameters. Equivalent to 'size' in dnbinom.
ps	Vector of individual probabilities of success.
n.samp	The number of samples per distribution
n.cores	The number of CPU cores to use in the evaluation. Allows parallelization.

Value

A numeric vector of random deviates.

Examples

```
rnbconv(mus = c(100, 10), phis = c(5, 8), n.samp = 10)
```

Index