Package ‘PoissonBinomial’

May 31, 2022

Type Package
Title Efficient Computation of Ordinary and Generalized Poisson Binomial Distributions
Version 1.2.5
Date 2022-05-31
Maintainer Florian Junge <florian.junge@h-da.de>
Language en-US
License GPL-3
Encoding UTF-8
Imports Rcpp (>= 1.0.3)
LinkingTo Rcpp
SystemRequirements fftw3 (>= 3.3)
Suggests knitr, rmarkdown, microbenchmark
VignetteBuilder knitr
URL https://github.com/DISOhda/PoissonBinomial
BugReports https://github.com/DISOhda/PoissonBinomial/issues
RoxygenNote 7.2.0
NeedsCompilation yes
Author Florian Junge [aut, cre]
Repository CRAN
Date/Publication 2022-05-31 10:10:02 UTC
R topics documented:

- PoissonBinomial-package .............................................. 2
- GenPoissonBinomial-Distribution ................................. 3
- PoissonBinomial-Distribution ..................................... 5

Index 9

PoissonBinomial-package

*Efficient Exact and Approximate Implementations for Computing Ordinary and Generalized Poisson Binomial Distributions*

Description

This package implements various algorithms for computing the probability mass function, the cumulative distribution function, quantiles and random numbers of both ordinary and generalized Poisson binomial distributions.

References


Examples

```r
# Functions for ordinary Poisson binomial distributions
set.seed(1)
pp <- c(1, 0, runif(10), 1, 0, 1)
qq <- seq(0, 1, 0.01)
dpbinom(NULL, pp)
ppbinom(7:10, pp, method = "DivideFFT")
qpbinom(qq, pp, method = "Convolve")
rpbinom(10, pp, method = "RefinedNormal")

# Functions for generalized Poisson binomial distributions
va <- rep(5, length(pp))
vb <- 1:length(pp)
dgpbinom(NULL, pp, va, vb, method = "Convolve")
pgpbinom(80:100, pp, va, vb, method = "Convolve")
qgpbinom(qq, pp, va, vb, method = "Convolve")
rgpbinom(100, pp, va, vb, method = "Convolve")
```
The Generalized Poisson Binomial Distribution

Description
Density, distribution function, quantile function and random generation for the generalized Poisson binomial distribution with probability vector \( \text{probs} \).

Usage

dgpbinom(x, probs, val_p, val_q, wts = NULL, method = "DivideFFT", log = FALSE)

pgpbinom(
  x,
  probs,
  val_p,
  val_q,
  wts = NULL,
  method = "DivideFFT",
  lower.tail = TRUE,
  log.p = FALSE
)

qgpbinom(
  p,
  probs,
  val_p,
  val_q,
  wts = NULL,
  method = "DivideFFT",
  lower.tail = TRUE,
  log.p = FALSE
)

rgpbinom(
  n,
  probs,
  val_p,
  val_q,
  wts = NULL,
  method = "DivideFFT",
  generator = "Sample"
)
Arguments

- **x**: Either a vector of observed sums or NULL. If NULL, probabilities of all possible observations are returned.
- **probs**: Vector of probabilities of success of each Bernoulli trial.
- **val_p**: Vector of values that each trial produces with probability in `probs`.
- **val_q**: Vector of values that each trial produces with probability in 1 - `probs`.
- **wts**: Vector of non-negative integer weights for the input probabilities.
- **method**: Character string that specifies the method of computation and must be one of "DivideFFT", "Convolve", "Characteristic", "Normal" or "RefinedNormal" (abbreviations are allowed).
- **log, log.p**: Logical value indicating if results are given as logarithms.
- **lower.tail**: Logical value indicating if results are $P[X \leq x]$ (if TRUE; default) or $P[X > x]$ (if FALSE).
- **p**: Vector of probabilities for computation of quantiles.
- **n**: Number of observations. If length(n) > 1, the length is taken to be the number required.
- **generator**: Character string that specifies the random number generator and must either be "Sample" or "Bernoulli" (abbreviations are allowed).

Details

See the references for computational details. The Divide and Conquer ("DivideFFT") and Direct Convolution ("Convolve") algorithms are derived and described in Biscarri, Zhao & Brunner (2018). They have been modified for use with the generalized Poisson binomial distribution. The Discrete Fourier Transformation of the Characteristic Function ("Characteristic") is derived in Zhang, Hong & Balakrishnan (2018), the Normal Approach ("Normal") and the Refined Normal Approach ("RefinedNormal") are described in Hong (2013). They were slightly adapted for the generalized Poisson binomial distribution.

In some special cases regarding the values of `probs`, the `method` parameter is ignored (see Introduction vignette).

Random numbers can be generated in two ways. The "Sample" method uses R's `sample` function to draw random values according to their probabilities that are calculated by `dgpbinom`. The "Bernoulli" procedure ignores the method parameter and simulates Bernoulli-distributed random numbers according to the probabilities in `probs` and sums them up. It is a bit slower than the "Sample" generator, but may yield better results, as it allows to obtain observations that cannot be generated by the "Sample" procedure, because `dgpbinom` may compute 0-probabilities, due to rounding, if the length of `probs` is large and/or its values contain a lot of very small values.

Value

dgpbinom gives the density, pgpbinom computes the distribution function, qgpbinom gives the quantile function and rgpbinom generates random deviates.

For rgpbinom, the length of the result is determined by `n`, and is the lengths of the numerical arguments for the other functions.
References


Examples

```r
set.seed(1)
pp <- c(1, 0, runif(10), 1, 0, 1)
qq <- seq(0, 1, 0.01)
va <- rep(5, length(pp))
vb <- 1:length(pp)

dgpbinom(NULL, pp, va, vb, method = "DivideFFT")
pgpbinom(75:100, pp, va, vb, method = "DivideFFT")
qgpbinom(qq, pp, va, vb, method = "DivideFFT")
rgpbinom(100, pp, va, vb, method = "DivideFFT")

dgpbinom(NULL, pp, va, vb, method = "Convolve")
pgpbinom(75:100, pp, va, vb, method = "Convolve")
qgpbinom(qq, pp, va, vb, method = "Convolve")
rgpbinom(100, pp, va, vb, method = "Convolve")

dgpbinom(NULL, pp, va, vb, method = "Characteristic")
pgpbinom(75:100, pp, va, vb, method = "Characteristic")
qgpbinom(qq, pp, va, vb, method = "Characteristic")
rgpbinom(100, pp, va, vb, method = "Characteristic")

dgpbinom(NULL, pp, va, vb, method = "Normal")
pgpbinom(75:100, pp, va, vb, method = "Normal")
qgpbinom(qq, pp, va, vb, method = "Normal")
rgpbinom(100, pp, va, vb, method = "Normal")

dgpbinom(NULL, pp, va, vb, method = "RefinedNormal")
pgpbinom(75:100, pp, va, vb, method = "RefinedNormal")
qgpbinom(qq, pp, va, vb, method = "RefinedNormal")
rgpbinom(100, pp, va, vb, method = "RefinedNormal")
```

PoissonBinomial-Distribution

The Poisson Binomial Distribution
Description

Density, distribution function, quantile function and random generation for the Poisson binomial distribution with probability vector probs.

Usage

dpbinom(x, probs, wts = NULL, method = "DivideFFT", log = FALSE)

ppbinom(
  x,
  probs,
  wts = NULL,
  method = "DivideFFT",
  lower.tail = TRUE,
  log.p = FALSE
)

qpbinom(
  p,
  probs,
  wts = NULL,
  method = "DivideFFT",
  lower.tail = TRUE,
  log.p = FALSE
)

rpbinom(n, probs, wts = NULL, method = "DivideFFT", generator = "Sample")

Arguments

x Either a vector of observed numbers of successes or NULL. If NULL, probabilities of all possible observations are returned.
probs Vector of probabilities of success of each Bernoulli trial.
wts Vector of non-negative integer weights for the input probabilities.
method Character string that specifies the method of computation and must be one of "DivideFFT", "Convolve", "Characteristic", "Recursive", "Mean", "GeoMean", "GeoMeanCounter", "Poisson", "Normal" or "RefinedNormal" (abbreviations are allowed).
log, log.p Logical value indicating if results are given as logarithms.
lower.tail Logical value indicating if results are \( P[X \leq x] \) (if TRUE; default) or \( P[X > x] \) (if FALSE).
p Vector of probabilities for computation of quantiles.
n Number of observations. If length(n) > 1, the length is taken to be the number required.
generator Character string that specifies the random number generator and must either be "Sample" (default) or "Bernoulli" (abbreviations are allowed). See Details for more information.
Details

See the references for computational details. The Divide and Conquer ("DivideFFT") and Direct Convolution ("Convolve") algorithms are derived and described in Biscarri, Zhao & Brunner (2018). The Discrete Fourier Transformation of the Characteristic Function ("Characteristic"), the Recursive Formula ("Recursive"), the Poisson Approximation ("Poisson"), the Normal Approach ("Normal") and the Refined Normal Approach ("RefinedNormal") are described in Hong (2013). The calculation of the Recursive Formula was modified to overcome the excessive memory requirements of Hong’s implementation. The "Mean" method is a naive binomial approach using the arithmetic mean of the probabilities of success. Similarly, the "GeoMean" and "GeoMeanCounter" procedures are binomial approximations, too, but they form the geometric mean of the probabilities of success ("GeoMean") and their counter probabilities ("GeoMeanCounter"), respectively.

In some special cases regarding the values of probs, the method parameter is ignored (see Introduction vignette).

Random numbers can be generated in two ways. The "Sample" method uses R’s sample function to draw random values according to their probabilities that are calculated by dgpbinom. The "Bernoulli" procedure ignores the method parameter and simulates Bernoulli-distributed random numbers according to the probabilities in probs and sums them up. It is a bit slower than the "Sample" generator, but may yield better results, as it allows to obtain observations that cannot be generated by the "Sample" procedure, because dgpbinom may compute 0-probabilities, due to rounding, if the length of probs is large and/or its values contain a lot of very small values.

Value
dpbinom gives the density, ppbinom computes the distribution function, qppbinom gives the quantile function and rpbinom generates random deviates.

For rpbinom, the length of the result is determined by n, and is the lengths of the numerical arguments for the other functions.

References


Examples

```r
set.seed(1)
pp <- c(0, 0, runif(995), 1, 1, 1)
qq <- seq(0, 1, 0.01)

dpbinom(NULL, pp, method = "DivideFFT")
ppbinom(450:550, pp, method = "DivideFFT")
qppbinom(qq, pp, method = "DivideFFT")
rppbinom(100, pp, method = "DivideFFT")
```
dpbinom(NULL, pp, method = "Convolve")
ppbinom(450:550, pp, method = "Convolve")
qpbinom(qq, pp, method = "Convolve")
rpbinom(100, pp, method = "Convolve")

dpbinom(NULL, pp, method = "Characteristic")
ppbinom(450:550, pp, method = "Characteristic")
qpbinom(qq, pp, method = "Characteristic")
rpbinom(100, pp, method = "Characteristic")

dpbinom(NULL, pp, method = "Recursive")
ppbinom(450:550, pp, method = "Recursive")
qpbinom(qq, pp, method = "Recursive")
rpbinom(100, pp, method = "Recursive")

dpbinom(NULL, pp, method = "Mean")
ppbinom(450:550, pp, method = "Mean")
qpbinom(qq, pp, method = "Mean")
rpbinom(100, pp, method = "Mean")

dpbinom(NULL, pp, method = "GeoMean")
ppbinom(450:550, pp, method = "GeoMean")
qpbinom(qq, pp, method = "GeoMean")
rpbinom(100, pp, method = "GeoMean")

dpbinom(NULL, pp, method = "GeoMeanCounter")
ppbinom(450:550, pp, method = "GeoMeanCounter")
qpbinom(qq, pp, method = "GeoMeanCounter")
rpbinom(100, pp, method = "GeoMeanCounter")

dpbinom(NULL, pp, method = "Poisson")
ppbinom(450:550, pp, method = "Poisson")
qpbinom(qq, pp, method = "Poisson")
rpbinom(100, pp, method = "Poisson")

dpbinom(NULL, pp, method = "Normal")
ppbinom(450:550, pp, method = "Normal")
qpbinom(qq, pp, method = "Normal")
rpbinom(100, pp, method = "Normal")

dpbinom(NULL, pp, method = "RefinedNormal")
ppbinom(450:550, pp, method = "RefinedNormal")
qpbinom(qq, pp, method = "RefinedNormal")
rpbinom(100, pp, method = "RefinedNormal")


Index

dgpbinom
    (GenPoissonBinomial-Distribution), 3
dpbinom (PoissonBinomial-Distribution), 5
GenPoissonBinomial-Distribution, 3
pgpbinom
    (GenPoissonBinomial-Distribution), 3
PoissonBinomial-Distribution, 5
PoissonBinomial-package, 2
ppbinom (PoissonBinomial-Distribution), 5
qgpbinom
    (GenPoissonBinomial-Distribution), 3
qpbinom (PoissonBinomial-Distribution), 5
rgpbinom
    (GenPoissonBinomial-Distribution), 3
rpbinom (PoissonBinomial-Distribution), 5