Package ‘PoissonBinomial’

March 28, 2020

Type Package
Title Efficient Computation of Ordinary and Generalized Poisson Binomial Distributions
Version 1.1.0
Date 2020-03-27
Maintainer Florian Junge <florian.junge@h-da.de>
License GPL (>= 2)
Encoding UTF-8
Imports Rcpp (>= 1.0.3)
LinkingTo Rcpp, BH
SystemRequirements fftw3 (>= 3.3)
RoxygenNote 7.1.0
Suggests knitr, rmarkdown, microbenchmark
VignetteBuilder knitr
NeedsCompilation yes
Author Florian Junge [aut, cre]
Repository CRAN
Date/Publication 2020-03-28 08:30:02 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)
dgpbivnom(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 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")

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.

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).

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)
```
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
)

qppbinom(qq, probs, wts = NULL, method = "DivideFFT")

rgppbinom(n, probs, wts = NULL, method = "DivideFFT")

dgpbinom(NULL, probs, wts = NULL, method = "DivideFFT")

dppbinom(75:100, probs, wts = NULL, method = "DivideFFT")

qppbinom(qq, probs, wts = NULL, method = "DivideFFT")

rgppbinom(n, probs, wts = NULL, method = "DivideFFT")

dgpbinom(NULL, probs, wts = NULL, method = "Convolve")

dppbinom(75:100, probs, wts = NULL, method = "Convolve")

qppbinom(qq, probs, wts = NULL, method = "Convolve")

rgppbinom(n, probs, wts = NULL, method = "Convolve")

dgpbinom(NULL, probs, wts = NULL, method = "Characteristic")

dppbinom(75:100, probs, wts = NULL, method = "Characteristic")

qppbinom(qq, probs, wts = NULL, method = "Characteristic")

rgppbinom(n, probs, wts = NULL, method = "Characteristic")

dgpbinom(NULL, probs, wts = NULL, method = "Normal")

dppbinom(75:100, probs, wts = NULL, method = "Normal")

qppbinom(qq, probs, wts = NULL, method = "Normal")

rgppbinom(n, probs, wts = NULL, method = "Normal")

dgpbinom(NULL, probs, wts = NULL, method = "RefinedNormal")

dppbinom(75:100, probs, wts = NULL, method = "RefinedNormal")

qppbinom(qq, probs, wts = NULL, method = "RefinedNormal")

rgppbinom(n, probs, wts = NULL, method = "RefinedNormal")

```r

dgbinom(NULL, pp, va, vb, method = "DivideFFT")

dppbinom(75:100, pp, va, vb, method = "DivideFFT")

qgbinom(qq, pp, va, vb, method = "DivideFFT")

rgbinom(100, pp, va, vb, method = "DivideFFT")

dgbinom(NULL, pp, va, vb, method = "Convolve")

dppbinom(75:100, pp, va, vb, method = "Convolve")

qgbinom(qq, pp, va, vb, method = "Convolve")

rgbinom(100, pp, va, vb, method = "Convolve")

dgbinom(NULL, pp, va, vb, method = "Characteristic")

dppbinom(75:100, pp, va, vb, method = "Characteristic")

qgbinom(qq, pp, va, vb, method = "Characteristic")

rgbinom(100, pp, va, vb, method = "Characteristic")

dgbinom(NULL, pp, va, vb, method = "Normal")

dppbinom(75:100, pp, va, vb, method = "Normal")

qgbinom(qq, pp, va, vb, method = "Normal")

rgbinom(100, pp, va, vb, method = "Normal")

dgbinom(NULL, pp, va, vb, method = "RefinedNormal")

dppbinom(75:100, pp, va, vb, method = "RefinedNormal")

qgbinom(qq, pp, va, vb, method = "RefinedNormal")

rgbinom(100, pp, va, vb, method = "RefinedNormal")
```
PoissonBinomial-Distribution

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

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

**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.

**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).
PoissonBinomial-Distribution

Value

dbinom gives the density, pbinom computes the distribution function, qbinom gives the quantile function and rbinom generates random deviates.

For rbinom, 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)
dbinom(NULL, pp, method = "DivideFFT")
pbinom(450:550, pp, method = "DivideFFT")
qbinom(qq, pp, method = "DivideFFT")
rbinom(100, pp, method = "DivideFFT")

dbinom(NULL, pp, method = "Convolve")
pbinom(450:550, pp, method = "Convolve")
qbinom(qq, pp, method = "Convolve")
rbinom(100, pp, method = "Convolve")

dbinom(NULL, pp, method = "Characteristic")
pbinom(450:550, pp, method = "Characteristic")
qbinom(qq, pp, method = "Characteristic")
rbinom(100, pp, method = "Characteristic")

dbinom(NULL, pp, method = "Recursive")
pbinom(450:550, pp, method = "Recursive")
qbinom(qq, pp, method = "Recursive")
rbinom(100, pp, method = "Recursive")

dbinom(NULL, pp, method = "Mean")
pbinom(450:550, pp, method = "Mean")
qbinom(qq, pp, method = "Mean")
rbinom(100, pp, method = "Mean")

dbinom(NULL, pp, method = "GeoMean")
pbinom(450:550, pp, method = "GeoMean")
qbinom(qq, pp, method = "GeoMean")
rbinom(100, pp, method = "GeoMean")
```

ppbinom(450:550, pp, method = "GeoMeanCounter")
qppbinom(qq, pp, method = "GeoMeanCounter")
rrpbinom(100, pp, method = "GeoMeanCounter")

dppbinom(NULL, pp, method = "Poisson")
ppbinom(450:550, pp, method = "Poisson")
qppbinom(qq, pp, method = "Poisson")
rrpbinom(100, pp, method = "Poisson")

dppbinom(NULL, pp, method = "Normal")
ppbinom(450:550, pp, method = "Normal")
qppbinom(qq, pp, method = "Normal")
rrpbinom(100, pp, method = "Normal")

dppbinom(NULL, pp, method = "RefinedNormal")
ppbinom(450:550, pp, method = "RefinedNormal")
qppbinom(qq, pp, method = "RefinedNormal")
rrpbinom(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