Package ‘iZID’

July 22, 2025

Title Identify Zero-Inflated Distributions

Version 0.0.1

Description
Computes bootstrapped Monte Carlo estimate of p value of Kolmogorov-Smirnov (KS) test and likelihood ratio test for zero-inflated count data, based on the work of Aldirawi et al. (2019) <doi:10.1109/BHI.2019.8834661>. With the package, user can also find tools to simulate random deviates from zero inflated or hurdle models and obtain maximum likelihood estimate of unknown parameters in these models.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

RoxygenNote 6.1.1

Imports extraDistr (>= 1.8.11), methods (>= 3.1.0), rootSolve (>= 1.7), foreach (>= 1.4.7), doParallel (>= 1.0.15), stats (>= 3.1.0)

Depends R (>= 3.1.0)

NeedsCompilation no

Author Lei Wang [aut, cre, cph],
Hani Aldirawi [aut, cph],
Jie Yang [aut, cph]

Maintainer Lei Wang <`slimewanglei@163.com`>

Repository CRAN

Date/Publication 2019-11-06 12:40:02 UTC

Contents

---

bb.mle *Maximum likelihood estimate for beta binomial distributions*

Description

calculate maximum likelihood estimate and the corresponding log likelihood value for beta binomial, beta negative binomial, negative binomial and Poisson distributions.

Usage

```
bb.mle(x, n, alpha1, alpha2, lowerbound = 0.01, upperbound = 10000)

bnb.mle(x, r, alpha1, alpha2, lowerbound = 0.01, upperbound = 10000)

nb.mle(x, r, p, lowerbound = 0.01, upperbound = 10000)

poisson.mle(x)
```

Arguments

`x`	A vector of count data. Should be non-negative integers.
`n`	An initial value of the number of trials. Must be a positive number, but not required to be an integer.
`alpha1`	An initial value for the first shape parameter of beta distribution. Should be a positive number.
`alpha2`	An initial value for the second shape parameter of beta distribution. Should be a positive number.
`lowerbound`	A lower searching bound used in the optimization of likelihood function. Should be a small positive number. The default is 1e-2.
`upperbound`	An upper searching bound used in the optimization of likelihood function. Should be a large positive number. The default is 1e4.
`r`	An initial value of the number of success before which m failures are observed, where m is the element of x. Must be a positive number, but not required to be an integer.
`p`	An initial value of the probability of success, should be a positive value within (0,1).

Details

`bb.mle`, `bnb.mle`, `nb.mle` and `poisson.mle` calculate the maximum likelihood estimate of beta binomial, beta negative binomial, negative binomial and Poisson distributions, respectively.

Please NOTE that the arguments in the four functions are NOT CHECKED AT ALL! The user must be aware of their inputs to avoid getting suspicious results.

Suppose that X is a random count variable that only takes non-negative values. If p has a prior distribution $beta(alpha1, alpha2)$ and X follows a binomial distribution $b(n, p)$, then X follows the beta binomial distribution with

$P(X = k) = C(n, k)Beta(k + alpha1, n - k + alpha2)/Beta(alpha1, alpha2)$,

where $C(,)$ is the combination function, $Beta(,)$ is the beta function and $beta(,)$ stands for the beta distribution.

If X stands for the number of failures observed before the rth success, the probability of X taking the value k under the negative binomial distribution equals

$P(X = k) = C(k + r - 1, k)p^r(1 - p)^k$,

As in beta binomial distribution, assume the prior distribution of p is $beta(alpha1, alpha2)$. X follows a beta negative binomial distribution if X follows a negative binomial distribution with parameters r and p. The probability density function of a beta negative binomial distribution is defined as:

$P(X = k) = \Gamma(r + k)Beta(r + alpha1, k + alpha2)/Beta(alpha1, alpha2)/\Gamma(r)/k!$,

where Γ represents the Gamma function.

With the only parameter $lambda$, the probability density function of a Poisson distribution is

$P(X = k) = lambda^k exp(-lambda)/k!$

The maximum likelihood estimate of all four distributions can be derived by minimizing the corresponding negative log likelihood function. It is easy to deduce the sample estimate of $lambda$ which is equal to the sample mean. However, it is not so straightforward to solve the optimization problems of the other three distributions. Thus, we adopt the optimization algorithm "L-BFGS-B" by calling R basic function `optim`. Lower and upper bounds on the unknown parameters are required for the algorithm "L-BFGS-B", which are determined by the arguments `lowerbound` and `upperbound`. But note that for the estimate of p, the upper bound for searching is essentially `1-lowerbound`.

Value

A row vector containing the maximum likelihood estimate of unknown parameters and the corresponding value of log likelihood.

With $bb.mle$, the following values are returned:

- n: the maximum likelihood estimate of n.
- alpha1: the maximum likelihood estimate of alpha1.
- alpha2: the maximum likelihood estimate of alpha2.
- loglik: the value of log likelihood with maximum likelihood estimates plugged-in.

With $bnb.mle$, the following values are returned:

- r: the maximum likelihood estimate of r.
- alpha1: the maximum likelihood estimate of alpha1.
- alpha2: the maximum likelihood estimate of alpha2.
- loglik: the value of log likelihood with maximum likelihood estimates plugged-in.

With $nb.mle$, the following values are returned:

- r: the maximum likelihood estimate of r.
- p: the maximum likelihood estimate of p.
- loglik: the value of log likelihood with maximum likelihood estimates plugged-in.

With $poisson.mle$, the following values are returned:

- lambda: the maximum likelihood estimate of lambda.
- loglik: the value of log likelihood with maximum likelihood estimate plugged-in.

Reference

- H. Aldirawi, J. Yang, A. A. Metwally (2019). Identifying Appropriate Probabilistic Models for Sparse Discrete Omics Data, accepted for publication in 2019 IEEE EMBS International Conference on Biomedical & Health Informatics (BHI).

Examples

```
x=extraDistr::rbbinom(2000,12,2,4)
bb.mle(x,3,1,1)
x=extraDistr::rbnbinom(2000,8,3,5)
bnb.mle(x, 3.3, 1, 1)
x=stats::rnbinom(2000,size=5,prob=0.3)
nb.mle(x, 7, 0.5)
x=stats::rpois(2000,7)
poisson.mle(x)
```

`bb.zihmle` *Maximum likelihood estimate for zero-inflated or hurdle beta binomial distributions.*

Description

Calculate maximum likelihood estimate and the corresponding log likelihood value for zero-inflated or hurdle beta binomial, beta negative binomial, negative binomial and Poisson distributions.

Usage

```
bb.zihmle(x, n, alpha1, alpha2, type = c("zi", "h"), lowerbound = 0.01,
  upperbound = 10000)

bnb.zihmle(x, r, alpha1, alpha2, type = c("zi", "h"),
  lowerbound = 0.01, upperbound = 10000)

nb.zihmle(x, r, p, type = c("zi", "h"), lowerbound = 0.01,
  upperbound = 10000)

poisson.zihmle(x, type = c("zi", "h"), lowerbound = 0.01,
  upperbound = 10000)
```

Arguments

`x`	A vector of count data. Should be non-negative integers.
`n`	An initial value of the number of trials. Must be a positive number, but not required to be an integer.
`alpha1`	An initial value for the first shape parameter of beta distribution. Should be a positive number.
`alpha2`	An initial value for the second shape parameter of beta distribution. Should be a positive number.
`type`	The type of distribution used to calculate the sample estimate, where 'zi' and 'h' stand for zero-inflated and hurdle distributions respectively.
`lowerbound`	A lower searching bound used in the optimization of likelihood function. Should be a small positive number. The default is 1e-2.
`upperbound`	An upper searching bound used in the optimization of likelihood function. Should be a large positive number. The default is 1e4.
`r`	An initial value of the number of success before which m failures are observed, where m is the element of x. Must be a positive number, but not required to be an integer.
`p`	An initial value of the probability of success, should be a positive value within (0,1).

Details

By setting `type='zi'`, `bb.zihmle`, `bnb.zihmle`, `nb.zihmle` and `poisson.zihmle` calculate the maximum likelihood estimate of zero-inflated beta binomial, beta negative binomial, negative binomial and Poisson distributions, respectively.

By setting `type='h'`, `bb.zihmle`, `bnb.zihmle`, `nb.zihmle` and `poisson.zihmle` calculate the maximum likelihood estimate of hurdle beta binomial, beta negative binomial, negative binomial and Poisson distributions, respectively.

Please NOTE that the arguments in the four functions are NOT CHECKED AT ALL! The user must be aware of their inputs to avoid getting suspicious results.

For zero-inflated models, zeros occurred by either sampling process or specific structure of data with the structural parameter $0 < \phi < 1$. The density function for a zero-inflated model is

$P_{zi}(X = k) = \phi 1_{k=0} + (1 - \phi)P(X = k)$,

where $P(X = k)$ is the probability under standard distributions.

Aldirawi et al. (2019) proposed an estimating procedure for zero-inflated models by optimizing over a reparametrization of the likelihood function where ϕ and the rest unknown parameters are separable. When X comes from a zero-inflated distribution, the maximum likelihood estimate of parameters except for ϕ are obtained by minimizing the truncated version of negative log likelihood function. However, in the zero-deflated case, $\phi = 0$ and the sample estimate of other parameters are identical to those for its corresponding standard distributions. Meanwhile, an warning message is shown on the screen such that 'cannot obtain mle with the current model type, the output estimate is derived from general ... distribution'.

For hurdle models, all zeros occurred purely by the structure of data with the structural parameter $0 < \phi < 1$. The density function for a hurdle model is

$P_h(X = k) = \phi 1_{k=0} + (1 - \phi)P_{tr}(X = k)$,

where $P_{tr}(X = k)$ is the truncated probability under standard distributions, where $P_{tr}(X = 0) = 0$ and $P_{tr}(X = k) = P(X = k)/(1 - P(X = 0))$. Since ϕ and other unknown parameters are separable in the joint likelihood function, ϕ can be estimated by a value with respect to the number of positive samples. The sample estimate of other parameters can be obtained by the same procedure for zero-inflated model.

A warning message may also occur when the algorithm of `optim` does not converge and the resulting estimates are not valid. In this case, the results from the corresponding general distribution are output instead.

Value

A row vector containing the maximum likelihood estimate of unknown parameters and the corresponding value of log likelihood.

With `bb.zihmle`, the following values are returned:

- n: the maximum likelihood estimate of n.
- alpha1: the maximum likelihood estimate of alpha1.
- alpha2: the maximum likelihood estimate of alpha2.
- phi: the maximum likelihood estimate of ϕ.
- loglik: the value of log likelihood with maximum likelihood estimates plugged-in.

With `bnb.zihmle`, the following values are returned:

- r: the maximum likelihood estimate of r.
- alpha1: the maximum likelihood estimate of alpha1.
- alpha2: the maximum likelihood estimate of alpha2.
- phi: the maximum likelihood estimate of ϕ.
- loglik: the value of log likelihood with maximum likelihood estimates plugged-in.

With `nb.zihmle`, the following values are returned:

- r: the maximum likelihood estimate of r.
- p: the maximum likelihood estimate of p.
- phi: the maximum likelihood estimate of ϕ.
- loglik: the value of log likelihood with maximum likelihood estimates plugged-in.

With `poisson.zihmle`, the following values are returned:

- lambda: the maximum likelihood estimate of lambda.
- phi: the maximum likelihood estimate of ϕ.
- loglik: the value of log likelihood with maximum likelihood estimate plugged-in.

Reference

- H. Aldirawi, J. Yang, A. A. Metwally (2019). Identifying Appropriate Probabilistic Models for Sparse Discrete Omics Data, accepted for publication in 2019 IEEE EMBS International Conference on Biomedical & Health Informatics (BHI).
- H. Aldirawi, J. Yang (2019). Model Selection and Regression Analysis for Zero-altered or Zero-inflated Data, Statistical Laboratory Technical Report, no.2019-01, University of Illinois at Chicago.

Examples

```
t1=sample.h(N=2000,phi=0.2,distri='Poisson',lambda=5)  ##hurdle poisson random values
t2=sample.h(N=2000,phi=0.2,distri='nb',r=10,p=0.6)   ##hurdle negative binomial
t3=sample.zi(N=2000,phi=0.2,distri='bb',alpha1=8,alpha2=9,n=10)  ##zero-inflated beta binomial
##zero-inflated beta negative binomial.
t4=sample.zi(N=2000,phi=0.2,distri='bnb',r=10,alpha1=8,alpha2=9)
bb.zihmle(t3,3,1,1,type='h')
bnb.zihmle(t4, 3.3, 1, 1,type='h')
nb.zihmle(t2, 7, 0.5,type='zi')
poisson.zihmle(t1,type='zi')
```

---

dis.kstest *The Monte Carlo estimate for the p-value of a discrete KS Test*

Description

Computes the Monte Carlo estimate for the p-value of a discrete one-sample Kolmogorov-Smirnov (KS) Test for Poisson, negative binomial, beta binomial, beta negative binomial distributions and their zero-inflated as well as hurdle versions.

Usage

```
dis.kstest(x, nsim = 100, bootstrap = TRUE, distri = "Poisson",
  r = NULL, p = NULL, alpha1 = NULL, alpha2 = NULL, n = NULL,
  lowerbound = 0.01, upperbound = 10000, parallel = FALSE)
```

Arguments

`x`	A vector of count data. Should be non-negative integers. If elements of x are not integers, they will be automatically rounded up to the smallest integers that are no less than themselves.
`nsim`	The number of bootstrapped samples or simulated samples generated to compute p-value. If it is not an integer, nsim will be automatically rounded up to the smallest integer that is no less than nsim. Should be greater than 30. Default is 100.
`bootstrap`	Whether to generate bootstrapped samples or not. See Details. 'TRUE' or any numeric non-zero value indicates the generation of bootstrapped samples. The default is 'TRUE'.

`distri`	The distribution used as the null hypothesis. Can be one of {'poisson','nb','bb', 'bnb','zip','zinb','zibb', zibnb','ph','nbh','bbh','bnbh'}, which corresponds to Poisson, negative binomial, beta binomial and beta negative binomial distributions and their zero-inflated as well as hurdle versions, respectively. Default is 'Poisson'.
`r`	An initial value of the number of success before which m failures are observed, where m is the element of x. Must be a positive number, but not required to be an integer.
`p`	An initial value of the probability of success, should be a positive value within (0,1).
`alpha1`	An initial value for the first shape parameter of beta distribution. Should be a positive number.
`alpha2`	An initial value for the second shape parameter of beta distribution. Should be a positive number.
`n`	An initial value of the number of trials. Must be a positive number, but not required to be an integer.
`lowerbound`	A lower searching bound used in the optimization of likelihood function. Should be a small positive number. The default is 1e-2.
`upperbound`	An upper searching bound used in the optimization of likelihood function. Should be a large positive number. The default is 1e4.
`parallel`	whether to use multiple threads to parallelize computation. Default is FALSE. Please aware that it may take longer time to execute the program with `parallel=FALSE`.

Details

For arguments `nsim`, `bootstrap`, `distri`, if the length is larger than 1, only the first element will be used. For other arguments except for `x`, the first valid value will be used if the input is not `NULL`, otherwise some naive sample estimates will be fed into the algorithm. Note that only the initial values that are occurred in the null distribution `distri` are needed. For example, with `distri=poisson`, user may provide a value for `lambda` but not for `r` or `p`, though it won't disturb the algorithm.

With an output p-value less than some user-specified significance level, `x` is very likely from a distribution other than the `distri`, given the current data. If p-values of more than one distributions are greater than the pre-specified significance level, user may consider a following likelihood ratio test to select a 'better' distribution.

The methodology of computing Monte Carlo p-value is taken from Aldirawi et al. (2019). With `bootstrap=TRUE`, `nsim` bootstrapped samples will be generated by resampling `x` without replacement. Otherwise, `nsim` samples are simulated from the null distribution with the maximum likelihood estimate of original data `x`. Then compute the maximum likelihood estimates of `nsim` bootstrapped or simulated samples, based on which `nsim` new samples are generated under the null distribution. `nsim` KS statistics are calculated for the `nsim` new samples, then the Monte Carlo p-value is resulted from comparing the `nsim` KS statistics and the statistic of original data `x`.

During the process of computing maximum likelihood estimates, the negative log likelihood function is minimized via basic R function `optim` with the searching interval decided by `lowerbound` and `upperbound`, except that the optimization of `p` takes `1-lowerbound` as the upper searching bound.

To accelerate the whole process, the algorithm uses the parallel strategy via the packages `foreach` and `doParallel`.

Value

An object of class 'dis.kstest' including the following elements:

- x: `x` used in computation.
- nsim: nsim used in computation.
- bootstrap: bootstrap used in computation.
- distri: distri used in computation..
- lowerbound: lowerbound used in computation.
- upperbound: upperboound used in computation.
- mle_new: A matrix of the maximum likelihood estimates of unknown parameters under the null distribution, using $nsim$ bootstrapped or simulated samples.
- mle_ori: A row vector of the maximum likelihood estimates of unknown parameters under the null distribution, using the original data `x`.
- pvalue: Monte Carlo p-value of the one-sample KS test.
- N: length of `x`.
- r: initial value of r used in computation.
- p: initial value of p used in computation.
- alpha1: initial value of alpha1 used in computation.
- alpha2: initial value of alpha2 used in computation.
- n: initial value of n used in computation.

Reference

- H. Aldirawi, J. Yang, A. A. Metwally (2019). Identifying Appropriate Probabilistic Models for Sparse Discrete Omics Data, accepted for publication in 2019 IEEE EMBS International Conference on Biomedical & Health Informatics (BHI).
- T. Wolodzko (2019). extraDistr: Additional Univariate and Multivariate Distributions, R package version 1.8.11, https://CRAN.R-project.org/package=extraDistr.
- R. Calaway, Microsoft Corporation, S. Weston, D. Tenenbaum (2017). doParallel: Foreach Parallel Adaptor for the 'parallel' Package, R package version 1.0.11, https://CRAN.R-project.org/package=doParallel.
- R. Calaway, Microsoft, S. Weston (2017). foreach: Provides Foreach Looping Construct for R, R package version 1.4.4, https://CRAN.R-project.org/package=foreach.

See Also

`model.lrt`

Examples

```
set.seed(2001)
temp1=sample.zi(N=300,phi=0.3,distri='poisson',lambda=5)
dis.kstest(temp1,nsim=100,bootstrap=TRUE,distri='Poisson')$pvalue
dis.kstest(temp1,nsim=100,bootstrap=TRUE,distri='nb')$pvalue
dis.kstest(temp1,nsim=100,bootstrap=TRUE,distri='zip')$pvalue
dis.kstest(temp1,nsim=100,bootstrap=TRUE,distri='zinb')$pvalue
```

---

model.lrt *likelihood ratio test for two models*

Description

Conduct likelihood ratio test for comparing two different models.

Usage

```
model.lrt(d1, d2, parallel = FALSE)
```

Arguments

d1	An object of class 'dis.kstest'.
d2	An object of class 'dis.kstest'.
parallel	Whether to use multiple threads to parallelize computation. Default is FALSE. Please aware that it may take longer time to execute the program with parallel=FALSE.

Details

If the pvalue of d1 and d2 are greater than the user-specified significance level, which indicates that the original data x may come from the two distributions in d1 and d2, a likelihood ratio test is desired to choose a more 'possible' distribution based on the current data. NOTE that the x in d1 and d2 must be IDENTICAL! Besides, NOTE that the distri in d1 and d2 must be DIFFERENT!

The distri inherited from d1 is the null distribution and that from d2 is used as the alternative distribution. Following Aldirawi et al. (2019), nsim bootstrapped or simulated samples will be generated according to bootstrap of d1, based on which nsim maximum likelihood estimates of the parameters in null distribution will be calculated. Remember that we have obtained nsim such maximum likelihood estimates while calling function dis.kstest. Thus, the algorithm just adopts the mle_new from d1 to avoid repetitive work. Using the nsim maximum likelihood estimates to generate nsim new samples and calculate nsim corresponding new likelihood ratio test statistics. The output p-value is the proportion of new samples that have statistics greater than the test statistic of the original data x.

As in dis.kstest, the computation is parallelized with the help of packages foreach and doParallel.

With the output p-value smaller than the user-specified significance level, the distri of d2 is more appropriate for modelling x. Otherwise, There is no significant difference between distri of d1 and distri of d2, given the current data.

Value

The p-value of the likelihood ratio test.

Reference

- H. Aldirawi, J. Yang, A. A. Metwally (2019). Identifying Appropriate Probabilistic Models for Sparse Discrete Omics Data, accepted for publication in 2019 IEEE EMBS International Conference on Biomedical & Health Informatics (BHI).
- T. Wolodzko (2019). extraDistr: Additional Univariate and Multivariate Distributions, R package version 1.8.11, https://CRAN.R-project.org/package=extraDistr.
- R. Calaway, Microsoft Corporation, S. Weston, D. Tenenbaum (2017). doParallel: Foreach Parallel Adaptor for the 'parallel' Package, R package version 1.0.11, https://CRAN.R-project.org/package=doParallel.
- R. Calaway, Microsoft, S. Weston (2017). foreach: Provides Foreach Looping Construct for R, R package version 1.4.4, https://CRAN.R-project.org/package=foreach.

See Also

`dis.kstest`

Examples

```
set.seed(2001)
temp1=sample.zi(N=300,phi=0.3,distri='poisson',lambda=5)
d1=dis.kstest(temp1,nsim=100,bootstrap=TRUE,distri='zip')
d2=dis.kstest(temp1,nsim=100,bootstrap=TRUE,distri='zinb')
model.lrt(d1,d2)
```

`OTU` *Bacterial OTUs.*

Description

A dataset containing the OTUs (Operational Taxonomic Units) of 229 bacteria.

Usage

```
OTU
```

Format

The original data set is a skin microbiome from the National Human Genome Research Institute [1,2], which contains 229 bacterial and fungal OTUs. The OTU data attached to this package is a data frame with 229 rows and 354 columns, where each row means one kind of bacteria and each column represents a feature. The first column `x` is the name of the bacteria. The rest of columns are the total number of reads for different features. For example, the second column is the total number of reads for feature 'Center.Back.01', and so forth.

Source

- 1. K. Findley, J. Oh, J. Yang, S. Conlan, C. Deming, J.A. Meyer, et al (2013). Human skin fungal diversity, Nature, 498:367–70.
- 2. E.A. Grice, H.H. Kong, S. Conlan, C.B. Deming, J. Davis, A.C. Young, et al (2009). Topographical and temporal diversity of the human skin microbiome, Science, 324:1190–2.

`sample.h` *Generate random deviates from zero-inflated or hurdle models*

Description

Generate random deviates from zero-inflated or hurdle Poisson, negative binomial, beta binomial and beta negative binomial models.

Usage

```
sample.h(N, phi, distri = "poisson", lambda = NA, r = NA, p = NA,
  alpha1 = NA, alpha2 = NA, n = NA)

sample.zi(N, phi, distri = "poisson", lambda = NA, r = NA, p = NA,
  alpha1 = NA, alpha2 = NA, n = NA)
```

Arguments

`N`	The sample size. Should be a positive number. If it is not an integer, N will be automatically rounded up to the smallest integer that no less than N.
`phi`	The structural parameter ϕ, should be a positive value within (0,1).
`distri`	The corresponding standard distribution. Can be one of {'poisson','nb','bb', 'bnb'}, which corresponds to Poisson, negative binomial, beta binomial and beta negative binomial distributions respectively.
`lambda`	A value for the parameter of Poisson distribution. Should be a positive number.
`r`	the number of success before which m failures are observed, where m is a random variable from negative binomial or beta negative binomial distribution. Must be a positive number. If it is not an integer, r will be automatically rounded up to the smallest integer that no less than r.
`p`	The probability of success, should be a positive value within (0,1).
`alpha1`	The first shape parameter of beta distribution. Should be a positive number.
`alpha2`	The second shape parameter of beta distribution. Should be a positive number.
`n`	The number of trials. Must be a positive number. If it is not an integer, n will be automatically rounded up to the smallest integer that no less than n.

Details

By setting `distri=poisson`, `sample.h` and `sample.zi` simulates `N` random deviates from hurdle and zero-inflated Poisson distribution, respectively, and so on forth. For arguments with length larger than 1, only the first element will be used.

Arguments `r` and `p` are for the use of zero-inflated and hurdle negative binomial distributions. `alpha1`, `alpha2` and `n` are for zero-inflated and hurdle beta binomial distributions. `r`, `alpha1` and `alpha2` are used in zero-inflated and hurdle beta negative binomial distributions.

The procedure of generating random deviates follows the work of Aldirawi et al. (2019). The algorithm calls functions for standard distributions to simulate the non-zero samples. Random deviates from standard Poisson and negative binomial distributions can be generated by basic R function `rpois` and `rnbinom`. Functions rbbinom and rbnbinom are available for standard beta binomial and beta negative binomial distributions in R package `extraDistr`.

Value

A vector of length N containing non-negative integers from the zero-inflated or hurdle version of distribution determined by $distri$.

Reference

- H. Aldirawi, J. Yang, A. A. Metwally, Identifying Appropriate Probabilistic Models for Sparse Discrete Omics Data, accepted for publication in 2019 IEEE EMBS International Conference on Biomedical & Health Informatics (BHI) (2019).
- T. Wolodzko, extraDistr: Additional Univariate and Multivariate Distributions, R package version 1.8.11 (2019), https://CRAN.R-project.org/package=extraDistr.

Examples

```
t1=sample.h(N=2000,phi=0.2,distri='Poisson',lambda=5)  ##hurdle poisson random values
t2=sample.h(N=2000,phi=0.2,distri='nb',r=10,p=0.6)   ##hurdle negative binomial
t3=sample.h(N=2000,phi=0.2,distri='bb',alpha1=8,alpha2=9,n=10)   ##hurdle beta binomial
##hurdle beta negative binomial.
t4=sample.h(N=2000,phi=0.2,distri='bnb',r=10,alpha1=8,alpha2=9)

t1=sample.zi(N=2000,phi=0.2,distri='Poisson',lambda=5) ##zero-inflated poisson random values
t2=sample.zi(N=2000,phi=0.2,distri='nb',r=10,p=0.6)   ##zero-inflated negative binomial
t3=sample.zi(N=2000,phi=0.2,distri='bb',alpha1=8,alpha2=9,n=10)  ##zero-inflated beta binomial
##zero-inflated beta negative binomial
t4=sample.zi(N=2000,phi=0.2,distri='bnb',r=10,alpha1=8,alpha2=9)
```

Index