Communications in Information and Systems
Volume 20, Number 1, 23–44, 2020

Identifying zero-inflated distributions with a new R package iZID

Lei Wang*, Hani Aldirawi*, and Jie Yang†

Count data with a large portion of zeros arise naturally in many scientific disciplines. When conducting one-sample Kolmogorov-Smirnov (KS) test for count data, the estimated p-value is biased due to plugging in sample estimates of unknown parameters. As a consequence, the result of a KS test could be too conservative. In the newly developed R package "iZID" for zero-inflated count data, we use bootstrapped Monte Carlo estimates to overcome the bias issue in estimating p-values, as well as bootstrapped likelihood ratio tests for zero-inflated model selection. Our new package also provides miscellaneous functions to simulate zero-inflated count data and calculate maximum likelihood estimates of unknown parameters. Compared with other R packages available so far, our package covers more types of zero-inflated distributions and provides adjusted p-value estimates after incorporating the influence of unknown model parameters. To facilitate the potential users, in this paper we provide detailed descriptions of functions in "iZID" and illustrate the use of them with executable R code.

AMS 2000 subject classifications: Primary 62G10, 62F10; secondary 62F40.
Keywords and phrases: Count data, hurdle model, Kolmogorov-Smirnov test, model selection, zero-inflated distribution.

1. Introduction

Sparse or zero-inflated count data arise naturally from a rich variety of scientific disciplines including microbiome [1], insurance claim [2, 3], security [4], health care [5], and more. Researchers often need to explore for the most appropriate probabilistic model which fits the zero-inflated data the best. In the statistical literature, one-sample Kolmogorov-Smirnov (KS) test provides a universal tool for testing if the random sample follows a specific continuous distribution with known parameters. More specifically,

*These two authors contributed to this paper equally.
†Supported in part by NSF grant DMS-1924859.

the one-sample KS test calculates the statistic $D_N = \sup_x |F_N(x) - F_0(x)|$, where $F_N(x) = N^{-1} \sum_{i=1}^{N} I_{(-\infty,x]}(x_i)$ is the empirical distribution function, $F_0(x)$ is the cumulative distribution function under the null hypothesis, and $\{x_1, x_2, \cdots, x_N\}$ is a given random sample of size N. Dimitrova *et al.* [6] developed an R package "KSgeneral" to compute the p-value of a KS test given a specified distribution function $F_0(x)$, which can be continuous, discrete, or mixed, but with known parameters. In practice, however, the distribution parameters are typically unknown. It is known that plugging in sample estimates of unknown parameters tends to overestimate the p-value of a KS test and consequently leads to a more conservative decision [7, 8].

To overcome the bias issue induced by plugging in estimated parameters, Aldirawi *et al.* [9] proposed a bootstrapped Monte Carlo procedure to estimate the p-value of a KS test for discrete probabilistic models. In the circumstance that more than one models pass the KS tests, Aldirawi *et al.* [9] proposed a bootstrapped procedure for estimating the p-values of the likelihood ratio tests for pairwise comparisons of candidate models. Based on [9], we develop a new R package named "iZID" for identifying Zero-Inflated and Hurdle Distributions, available from the Comprehensive R Archive Network (CRAN, https://CRAN.R-project.org/package=iZID). For user's convenience, we cover regular Poisson, negative binomial, beta binomial, and beta negative binomial distributions as well. Using "iZID", the p-value is estimated by counting the number of random samples whose KS test statistics are greater than the KS statistic derived from the original data. Since the random samples are generated using the maximum likelihood estimates obtained from the bootstrapped or original data, the resulting p-value is automatically adjusted for the influence of plugging in sample estimates.

The rest of this paper is structured as follows. In Section 2, we review commonly used probabilistic distributions for modeling count data, including four regular distributions and their zero-inflated and hurdle versions. We also briefly review existing R packages for analyzing zero-inflated data in the last subsection of Section 2. In Section 3, we dissect the package "iZID" and present the syntax of major functions. In Section 4, we illustrate the use of "iZID" with executable R code and examples. We summarize in Section 5.

2. Commonly used probabilistic distributions for modeling count data

In the statistics literature, Poisson, negative binomial, and their zero-inflated and hurdle versions have been commonly used in modeling count data [1]. Nevertheless, Aldirawi *et al.* [9] suggested that zero-inflated or hurdle models

of beta binomial and beta negative binomial distributions might be more appropriate for modeling sparse omics data. In this section, besides commonly used zero-inflated Poisson (ZIP) and zero-inflated negative binomial (ZINB) distributions, we also review beta binomial (BB), beta negative binomial (BNB), zero-inflated beta binomial (ZIBB), zero-inflated beta negative binomial (ZIBNB), beta binomial hurdle (BBH), and beta negative binomial hurdle (BNBH) distributions, which are more flexible by attaching a beta prior distribution.

2.1. Standard probabilistic distributions for modeling counts

(I) Poisson model If X follows a Poisson distribution with mean $\lambda > 0$, then $P(X = k) = \frac{\lambda^k}{k!}\exp(-\lambda)$, where $k = 0, 1, 2, \cdots$. The log-likelihood function of λ given a random sample $\{x_1, \ldots, x_N\}$ of size N can be written as

$$l(\lambda) = \sum_{i=1}^{N} x_i \cdot \log\lambda - N\lambda - \sum_{i=1}^{N} \log\Gamma(x_i + 1)$$

where the gamma function $\Gamma(x + 1) = x!$.

(II) Negative binomial model Let $p \in (0, 1)$ be the probability of success in a sequence of independent Bernoulli trials and X be the number of failures observed before the r^{th} success ($r \geq 1$). Then $P(X = k) = \binom{k+r-1}{k} p^r (1-p)^k$, $k = 0, 1, 2, \cdots$. The log-likelihood function of (r, p) given $\{x_1, \ldots, x_N\}$ can be written as

$$\begin{aligned} l(r,p) &= \sum_{i=1}^{N} \log\Gamma(x_i + r) - N\log\Gamma(r) + rN\log p + \sum_{i=1}^{N} x_i \cdot \log(1-p) \\ &\quad - \sum_{i=1}^{N} \log\Gamma(x_i + 1) \end{aligned}$$

(III) Beta binomial model First let X be a binomial random variable with parameters $n \geq 1$ and $p \in (0, 1)$. Further let p have a beta prior distribution with parameters $\alpha_1 > 0$ and $\alpha_2 > 0$. Then X follows a beta binomial distribution with probability mass function $P(X = k) = \binom{n}{k} \frac{\text{Beta}(k+\alpha_1, n-k+\alpha_2)}{\text{Beta}(\alpha_1, \alpha_2)}$, where $\text{Beta}(a, b) = \frac{\Gamma(a)\Gamma(b)}{\Gamma(a+b)}$. The log-likelihood function of parameters (n, α_1, α_2) given $\{x_1, \ldots, x_N\}$ can be written as

$$l(n, \alpha_1, \alpha_2) = N\log\Gamma(n+1) + \sum_{i=1}^{N} \log\Gamma(x_i + \alpha_1) + \sum_{i=1}^{N} \log\Gamma(n - x_i + \alpha_2)$$

$$+ \quad N\log\Gamma(\alpha_1+\alpha_2) - \sum_{i=1}^{N}\log\Gamma(x_i+1) - \sum_{i=1}^{N}\log\Gamma(n-x_i+1)$$
$$- \quad \log\Gamma(n+\alpha_1+\alpha_2) - N\log\Gamma(\alpha_1) - N\log\Gamma(\alpha_2)$$

(IV) Beta negative binomial model First let X be a negative binomial random variable with parameters $r \geq 1$ and $p \in (0,1)$. Further let p have a beta prior distribution with parameters $\alpha_1 > 0$ and $\alpha_2 > 0$. Then X follows a beta negative binomial distribution with probability mass function $P(X=k) = \frac{\Gamma(r+k)\mathrm{Beta}(\alpha_1+r,\alpha_2+k)}{k!\Gamma(r)\mathrm{Beta}(\alpha_1,\alpha_2)}$. The log-likelihood function of parameters (r,α_1,α_2) given $\{x_1,\ldots,x_N\}$ can be written as

$$\begin{aligned} l(r,\alpha_1,\alpha_2) &= \sum_{i=1}^{N}\log\Gamma(x_i+r) + N\log\Gamma(\alpha_1+r) + \sum_{i=1}^{N}\log\Gamma(x_i+\alpha_2) \\ &+ N\log\Gamma(\alpha_1+\alpha_2) - \sum_{i=1}^{N}\log\Gamma(x_i+1) - N\log\Gamma(r) \\ &- \sum_{i=1}^{N}\log\Gamma(x_i+r+\alpha_1+\alpha_2) - N\log\Gamma(\alpha_1) - N\log\Gamma(\alpha_2) \end{aligned}$$

The parameter estimates for these models are obtained by maximizing the corresponding log-likelihood function, known as the maximum likelihood estimates (mle). The optimization procedures in "iZID" are implemented using the R function `optim`.

2.2. Zero-inflated probabilistic distributions for modeling counts

Zero-inflated distributions provide a flexible way to model data with an excess of zeros. Let $P_{\boldsymbol{\theta}}$ stand for the probability mass function (pmf) of a standard probabilistic model with parameters $\boldsymbol{\theta}$ and let $\phi \in [0,1]$ be the weight parameter of excess zeros, then the pmf of the corresponding zero-inflated distribution is

$$P_{\mathrm{ZI}}(k;\phi,\boldsymbol{\theta}) = \phi\mathbf{1}_{\{k=0\}} + (1-\phi)P_{\boldsymbol{\theta}}(k) \tag{1}$$

Given a random sample $\{X_1,\ldots,X_N\}$ from model (1), we aim to find the maximum likelihood estimate $\hat{\phi}$ for ϕ and $\hat{\boldsymbol{\theta}}$ for $\boldsymbol{\theta}$. Without any loss of generality, we rearrange $X_1,\ldots,X_N$ such that $X_i \neq 0$ for $i=1,\ldots,m$ and

$X_{m+1} = \cdots = X_N = 0$. In other words, $m = \#\{i : X_i \neq 0\}$ is the number of nonzero observations. Then the likelihood function of $(\phi, \boldsymbol{\theta})$ is

$$L(\phi, \boldsymbol{\theta}) = [\phi + P_{\boldsymbol{\theta}}(0)(1-\phi)]^{N-m} (1-\phi)^m [1 - P_{\boldsymbol{\theta}}(0)]^m \prod_{i=1}^{m} P_{\mathrm{tr}}(X_i; \boldsymbol{\theta}) \tag{2}$$

where $P_{\mathrm{tr}}(k; \boldsymbol{\theta}) = P_{\boldsymbol{\theta}}(k)/[1 - P_{\boldsymbol{\theta}}(0)]$, $k \neq 0$ is the zero-truncated pmf.

Since $L(\phi, \boldsymbol{\theta})$ in (2) is not separable for ϕ and $\boldsymbol{\theta}$, in "iZID" package we adopt the reparametrization method proposed in Aldirawi and Yang [10] for estimating ϕ and $\boldsymbol{\theta}$.

2.3. Hurdle probabilistic distributions for modeling counts

Hurdle or zero-altered models provide another way to deal with data containing many zeros. While zero-inflated models add extra probability $\phi[1-P_{\boldsymbol{\theta}}(0)]$ to the occurrence of zeros, hurdle models set ϕ as the probability of zeros, which can be more or less than $P_{\boldsymbol{\theta}}(0)$ determined by the standard models. The pmf of the corresponding hurdle model is

$$P_{\mathrm{ZA}}(k; \phi, \boldsymbol{\theta}) = \phi \mathbf{1}_{\{k=0\}} + (1-\phi) P_{\mathrm{tr}}(k; \boldsymbol{\theta}) \tag{3}$$

with the zero-truncated pmf $P_{\mathrm{tr}}(k; \boldsymbol{\theta})$. Note that the only difference between the hurdle model (3) and the corresponding zero-inflated model (1) is that, the hurdle model P_{ZA} is built from the zero-truncated pmf P_{tr} instead of the original baseline pmf $P_{\boldsymbol{\theta}}$ in P_{ZI}.

The likelihood function of model (3) is

$$L(\phi, \boldsymbol{\theta}) = \phi^{N-m} (1-\phi)^m \prod_{i=1}^{m} P_{\mathrm{tr}}(X_i; \boldsymbol{\theta}) \tag{4}$$

Since $L(\phi, \boldsymbol{\theta})$ in (4) is separable for ϕ and $\boldsymbol{\theta}$, it is straightforward to obtain $\hat{\phi} = 1 - m/N$. We follow Aldirawi and Yang [10] to estimate $\boldsymbol{\theta}$ over nonzero observations.

2.4. Existing R packages for analyzing zero-inflated data

Several packages are currently available from the Comprehensive R Archive Network (CRAN) for analyzing zero-inflated data, including "bzinb" (Cho *et al.* [11]), "hurdlr" (Balderama *et al.* [12]), "mazeinda" (Albasi [13]),

"mhurdle" (Croissant *et al.* [14]), "rbtt" (Waudby-Smith *et al.* [15]), "ZIBBSeqDiscovery" (Hu *et al.* [16]), "ZIBseq" (Peng *et al.* [17]), "zic" (Jochmann [18]), "ZIM" (Yang *et al.* [19]) and "ziphsmm" (Xu *et al.* [20]), etc.

Package "bzinb" provides tools for random sample generation, maximum likelihood estimation and log likelihood computation for bivariate zero-inflated negative binomial and Poisson distributions. With "hurdlr", users are able to fit hurdle or zero-inflated negative binomial and Poisson regression models using Bayesian strategy. Package "mazeinda" is tailored to compute and test the significance of pairwise monotonic association for count data with any degree of zero-inflation. The creation of "mhurdle" is inspired by the households' expenditure survey data where many zeros exist in predictors recording the cost of some goods or activities. The function `mhurdle` in package "mhurdle" enables the estimation of a large class of regression models with up to three hurdles, which allows that zero observations in predictors occur by up to three structural reasons. Package "rbtt" tries to tackle the inflation of type I error in two-sample t-tests comparing two groups of zero-inflated data via robust bootstrapped test. Package "ZIBBSeqDiscovery" models the relationship between the count data and some covariates of interest by zero-inflated beta-binomial models. Package "ZIBseq" regresses the counts on categorical clinical conditions in zero-inflated beta models. Package "zic" outputs the Bayesian estimate of zero-inflated count models while assuming that the parameters follow certain prior distributions. Package "ZIM" enables both observation-driven and parameter-driven modeling for time series with excess zeros. Package "ziphsmm" analyses longitudinal continuous-time data via zero-inflated Poisson hidden (semi-)Markov models.

Except for packages "mazeinda" and "rbtt", the rest fit count data to specific models. To the best of our knowledge, our package "iZID" is the first one to conduct KS test for count data with p-values adjusted for the influence of sample estimate of unknown parameters. Example 2.1 below shows that our function `dis.kstest` is more reliable than the basic R function `ks.test` in estimating p-values. For more comparisons, please see Section 4.2.

Example 2.1. In this experiment, we simulate $N = 100$ random numbers from a zero-inflated negative binomial (ZINB) distribution with parameters $\phi = 0.6, r = 2, p = 0.01$. The maximum likelihood estimates $\hat{\phi} = 0.590, \hat{r} = 2.06, \hat{p} = 0.011$ are fairly accurate. Nevertheless, if one wants to test if the original sample from a ZINB distribution by simulating another random sample using the estimated parameter values, the classical R function `ks.test` rejects ZINB model with p-value 0.01 and a warning message. If we use our function `dis.kstest` in package "iZID", the adjusted p-value

is 0.12 which passes the ZINB model. For readers' reference, we provide the R code and output below:

```
> set.seed(343)
> nsimu=100
> x=sample.zi(N=nsimu, phi=0.6, distri = "nb", r=2, p=0.01)
> mle=nb.zihmle(x, r=5, p=0.5, type="zi")
> mle
            r         p       phi    loglik
[1,] 2.058397 0.0112907 0.5899598 -316.7666
> y=sample.zi(N=nsimu, phi=mle[3], distri = "nb", r=mle[1],
p=mle[2])
> ks.test(x,y)

Two-sample Kolmogorov-Smirnov test

data:  x and y
D = 0.23, p-value = 0.01008
alternative hypothesis: two-sided

Warning message:
In ks.test(x, y): p-value will be approximate in the presence
of ties
> dis.kstest(x, nsim=200, bootstrap = TRUE,
distri = "zinb")$pvalue
[1] 0.12
```

3. Architecture of the package "iZID"

The package "iZID" contains four main functions: `dis.kstest`, `model.lrt`, `sample.zi` and `sample.h`. Function `dis.kstest` computes bootstrapped or Monte Carlo p-value of one-sample KS test under a specific discrete distribution. Function `model.lrt` implements a likelihood ratio test to select between two candidate models, in the case that more than one models have p-values greater than the pre-specified significance level. Functions `sample.zi` and `sample.h` are random sample generators, where the former outputs random deviates of zero-inflated models and the latter generates random counts from hurdle models. This package also provides some miscellaneous functions to calculate maximum likelihood estimates and the corresponding log likelihood value for a large set of models modeling count data. To accelerate the calculation process, we parallelize the computation of bootstrapped Monte Carlo estimates using R package "doParallel" [21] and "foreach" [22].

3.1. Estimate p-value of one-sample KS test

To estimate the p-value of a KS test given a pre-specified distribution as null hypothesis, the user may call the function `dis.kstest` with the syntax:

```
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)
```

`x`	Independent non-negative integers which stands for counts. Can be a vector or a matrix.
`nsim`	Number of bootstrapped samples generated for computing maximum likelihood estimate of unknown parameters.
`distri`	The distribution under null hypothesis. Currently, standard Poisson, negative binomial, beta binomial, beta negative binomial distributions as well as their zero-inflated and hurdle versions are available in the package. Accordingly, `distri` can be set to be one of {Poisson, nb, bb, bnb, zip, zinb, zibb, zibnb, ph, nbh, bbh, bnbh}. Note that users do not need to provide an estimate for unknown parameters. Instead, `dis.kstest` automatically carries out the task.
`r, p`	Optional arguments for assigning initial values of unknown parameters of standard, zero-inflated and hurdle negative binomial distributions.
`alpha(1,2) and n`	Optional arguments for assigning initial values of unknown parameters of standard, zero-inflated and hurdle beta binomial distributions.
`alpha(1,2) and r`	Optional arguments for assigning initial values of unknown parameters of standard, zero-inflated and hurdle beta negative binomial distributions.
`lowerbound`	The lower searching bound.
`upperbound`	The upper searching bound.

Details:

- `dis.kstest` will be initialized with naive sample estimates if initial values are not given. 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.

- The way to calculate p-value of KS test is taken from [9] and illustrated in Algorithm 1. Given a random sample $\mathbf{x} = \{x_1, x_2, \cdots, x_N\}$, `nsim` bootstrapped samples are obtained by resampling $\mathbf{x}$ with replacement if setting `bootstrap=TRUE` (by default). If setting `bootstrap=FALSE`, `nsim` new samples will be simulated with the mle of the original data $\mathbf{x}$, and KS statistics of the new samples will be computed.

Value:
An object of class "dis.kstest" including all the input values, as well as

- `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 $\mathbf{x}$.
- `pvalue`: Monte Carlo p-value of the one-sample KS test.

3.2. Likelihood ratio test for model selection

If the p-values returned by `dis.kstest` are not significant for more than one distributions, a likelihood ratio test can be performed to select a relatively "better" model for the data on hand. The way to call `model.lrt` is as follows:

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

where `d1` and `d2` are two objects of class "dis.kstest" under different distributions. The likelihood ratio test statistic is the difference between log likelihood of the alternative and the null distribution decided by `d2` and `d1`, respectively. The algorithm under the hood follows [9] with the pseudocode given in Algorithm 2. One may simulate `nsim` new samples under the null distribution using `nsim` mles inherited from `d1$mle_new`, and then calculate the differences between log likelihood of new samples under the alternative and the null hypotheses as statistics of the likelihood ratio tests. Function `dis.kstest` returns the proportion of test statistics of new samples that are greater than the statistic of original data $\mathbf{x}$. A small p-value indicates that the data on hand is more likely to come from the alternative distribution. Otherwise, the null distribution shows no significant difference to the alternative one.

Algorithm 1: Pseudocode of `dis.kstest` for discrete KS test

Input : `x`, `nsim`, `bootstrap`, `distri`, `lowerbound`, `upperbound`, `parallel`.
Output: p-value of the KS test.

Assigning initial values of unknown parameters using naive sample estimates or input values;
$mle\_ori \leftarrow$ maximum likelihod estimates of unknown parameters using `x`;
$Dn\_ori \leftarrow$ KS test statistic of `x` under `distri` whose parameters equal mle_ori;
Let mle_new be a matrix with `nsim` columns;
for $j \leftarrow 1$ *to* `nsim` **do**
  if `bootstrap` **then**
    $x\_new \leftarrow$ sampling N observations from `x` with replacement // N is the length of vectorized `x`;
    $mle\_new[, j] \leftarrow$ maximum likelihod estimates of unknown parameters using `x_new`;
  else
    $mle\_new[, j] \leftarrow mle\_ori$;
  end if
end for
Let $D2$ be a vector with `nsim` elements;
for $j \leftarrow 1$ *to* `nsim` **do**
  $x\_new \leftarrow$ a vector of N random deviates generated from `distri` with unknown parameters equal to $mle\_new[, j]$;
  $D2[j] \leftarrow$ KS test statistic of x_new under `distri` whose parameters equal $mle\_new[, j]$;
end for
$pvalue \leftarrow \frac{\#\{D2 > Dn\_ori\}}{nsim}$;
return $pvalue$;

3.3. Generate random samples from zero-inflated and hurdle distributions

Random deviates from standard Poisson and negative binomial distributions can be generated by basic R functions `rpois` and `rnbinom`, respectively. With R package "ExtraDistr" (Wolodzko [23]), functions `rbbinom` and `rbnbinom` are available for standard beta binomial and beta negative binomial distributions, respectively. In addition, there are a few other R packages for generating a dataset from some hurdle distributions. For ex-

Algorithm 2: Pseudocode of `model.lrt` for likelihoood ratio test

Input : `d1`, `d2`.
Output: p-value of the likelihood ratio test.
Initialization:
  $mle\_new \leftarrow d1\$mle\_new$; $N \leftarrow d1\$N$; $distri1 \leftarrow d1\$distri$; $nsim \leftarrow d1\$nsim$; $distri2 \leftarrow d2\$distri$; $t1 \leftarrow$ log likelihood under $distri1$, given original data; $t2 \leftarrow$ log likelihood under $distri2$, given original data; also adopts other values in `d1` and `d2`.
end
$t\_ori \leftarrow (t2 - t1)$;
Let t_new be a vector with `nsim` elements;
for $j \leftarrow 1$ *to* `nsim` **do**
  $x\_new \leftarrow$ a vector of N random deviates generated from $distri1$ with unknown parameter(s) equal to $mle\_new[, j]$;
  $t1 \leftarrow$ log likelihood under $distri1$ with maximum likelihood estimates plugged-in, given x_new;
  $t2 \leftarrow$ log likelihood under $distri2$ with maximum likelihood estimates plugged-in, given x_new;
  $t\_new[j] \leftarrow (t2 - t1)$;
end for
$pvalue \leftarrow \frac{\#\{t\_new > t\_ori\}}{nsim}$;
return $pvalue$;

ample, package "countreg" provides function `hpois` for generating dataset from Poisson hurdle distribution.

In our package "iZID", we allow the use of new distributions including beta binomial and beta negative binomial distributions, and more importantly, their corresponding zero-inflated and hurdle models. Aldirawi *et al.* [9] introduced a procedure grounded upon the central limit theorem to produce random values from zero-inflated and hurdle models. In package "iZID", we implement the procedure to the following two functions:

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

These two functions have exactly the same arguments. Here `N` represents the size of random sample to return. Argument `phi` stands for the value of structural parameter ϕ in zero-inflated and hurdle models, e.g., formulae (1) and (3). The input `distri` currently belongs to the set of four standard

distributions {Poisson, nb, bb, bnb}. For example, by setting `distri=nb`, `sample.zi` and `sample.h` return zero-inflated and hurdle negative binomial distributed random deviates, respectively. Arguments `lambda`, `r`, `p`, `alpha1`, `alpha2` and `n` are parameter values for different distributions, which must be specified. For instance, with `distri=nb`, users need to provide values for `r` and `p`.

3.4. Calculate maximum likelihood estimate and log likelihood

In order to calculate the maximum likelihood estimate as well as the value of log likelihood of the aforementioned four standard distributions and their zero-inflated and hurdle versions, one may simply use the following lines of code with package "iZID":

```
poisson.mle(x)
bb.mle(x,n,alpha1,alpha2,lowerbound = 0.01,
  upperbound = 10000)
nb.mle(x,r,p,lowerbound = 0.01, upperbound = 10000)
bnb.mle(x,r,alpha1,alpha2,lowerbound = 0.01,
  upperbound = 10000)
poisson.zihmle(x,type=c('zi','h'),lowerbound = 0.01,
  upperbound = 10000)
bb.zihmle(x,n,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)
bnb.zihmle(x,r,alpha1,alpha2,type=c('zi','h'),
  lowerbound = 0.01, upperbound = 10000)
```

The first four functions are designed for standard distributions. The rest are for zero-inflated models with setting `type='zi'` and hurdle models with setting `type='h'`. Note that the value of arguments will not be checked within the functions. Thus, results could be misleading with improper inputs. When calling `nb.zihmle` and `bnb.zihmle`, the users may receive warning messages such as "`...cannot obtain mle with the current model type...`" if the optimization procedure by R function `optim` does not converge. In this case, the output will be identical to the maximum likelihood estimates for standard negative binomial or beta negative binomial distribution.

4. Illustration

4.1. Quick start

In order to utilize the "iZID" package, one may use the functions described in Section 3.3 to generate two random samples of size 28 from a zero-inflated negative binomial distribution with parameters $(\phi, r, p) = (0.3, 6, 0.4)$ and a hurdle beta negative binomial distribution with parameters $(\phi, r, \alpha_1, \alpha_2) = (0.6, 6, 3, 7)$, respectively.

```
 library(iZID) ##load the package
 sample.zi(N=28,phi=0.3,distri='nb',r=6,p=0.4)
[1] 8 11 10 12 13 0 6 0 10 7 0 5 8 11 11 5 7 7 6 15 0 5 9
14 10 5 10 12
 sample.h(N=28,phi=0.6,distri='bnb',r=6,alpha1=3,alpha2=7)
[1] 0 14 0 20 0 0 17 0 0 0 18 0 0 0 0 0 36 19 14 20 0 24 0
7 2 10 0 0
```

One may test if the maximum likelihood estimates of parameters are close to the truth.

```
 temp1=sample.zi(N=300,phi=0.3,distri='poisson',lambda=5)
 poisson.zihmle(temp1,type='zi')
      lambda     phi     loglik
[1,] 5.058126  0.2955213  -640.1416
```

From the above output, the estimates of λ and ϕ approximate the true values. In the circumstances when the underlying distribution of data `temp1` is unknown, one may fit other models as follows:

```
 nb.zihmle(temp1,type='zi',r=3,p=0.5)
        r         p         phi     loglik
[1,] 340.5231 0.9853779 0.295327 -640.1886
 bb.zihmle(temp1,type='zi',n=3,alpha1=3,alpha2=5)
      n     alpha1   alpha2   phi    loglik
[1,] 637.37  28.23   178.30   0.3   7120.57
 bnb.zihmle(temp1,type='zi',r=3,alpha1=3,alpha2=5)
        r      alpha1    alpha2   phi  loglik
[1,] 10000   1614.465  541.4786  0   30367934
Warning message:
In bnb.zihmle(temp1, type = ''zi'', r = 3, alpha1 = 3, alpha2
 = 5, : cannot obtain mle with the current model type, the
output estimate is derived from general beta negative
binomial distribution.
```

Note that the log likelihood of beta binomial distribution for data `temp1` exceeds that of zero-inflated Poisson distribution, though the latter is the true underlying model. It suggests the need of conducting KS tests to identify an "appropriate" model before estimating model parameters. Without specifying any initial guess on parameters, the procedure of obtaining p-values works as follows:

```
 dis.kstest(temp1,nsim=100,bootstrap=TRUE,
   distri='Poisson')$pvalue
[1] 0
 dis.kstest(temp1,nsim=100,bootstrap=TRUE,
   distri='nb')$pvalue
[1] 0
 dis.kstest(temp1,nsim=100,bootstrap=TRUE,
   distri='bb')$pvalue
[1] 0
 dis.kstest(temp1,nsim=100,bootstrap=TRUE,
   distri='bnb')$pvalue
[1] 0
 dis.kstest(temp1,nsim=100,bootstrap=TRUE,
   distri='zip')$pvalue
[1] 0.97
 dis.kstest(temp1,nsim=100,bootstrap=TRUE,
   distri='zinb')$pvalue
[1] 0.97
 dis.kstest(temp1,nsim=100,bootstrap=TRUE,
   distri='zibb')$pvalue
[1] 0
 dis.kstest(temp1,nsim=100,bootstrap=TRUE,
   distri='zibnb')$pvalue
[1] 0
Warning message:
In bnb.zihmle(x, r, alpha1, alpha2, type = ``zi'') :
cannot obtain mle with the current model type, the output
estimate is derived from general beta negative binomial
distribution.

 dis.kstest(temp1,nsim=100,bootstrap=TRUE,
   distri='ph')$pvalue
[1] 0.98
 dis.kstest(temp1,nsim=100,bootstrap=TRUE,
   distri='nbh')$pvalue
```

```
[1] 0.94
 dis.kstest(temp1,nsim=100,bootstrap=TRUE,
   distri='bbh')$pvalue
[1] 0
 dis.kstest(temp1,nsim=100,bootstrap=TRUE,
   distri='bnbh')$pvalue
[1] 0
Warning message:
In bnb.zihmle(x, r, alpha1, alpha2, type = ''h'') :
cannot obtain mle with the current model type, the output
estimate is derived from general beta negative binomial
distribution.
```

The divergence of empirical distribution of `temp1` from zero-inflated Poisson and negative binomial distributions and their hurdle versions is not significant with p-values close to 1. Since a zero-inflated model and its hurdle version are closely related, we are more interested in distinguishing two types of distributions, say, zero-inflated Poisson or negative binomial, which can be done by using the function `model.lrt`. Define the two "dis.kstest" objects returned from zero-inflated Poisson and negative binomial as "d1" and "d2", respectively.

```
 model.lrt(d1,d2)
[1] 0.5
```

With the current sample size of data `temp1`, the likelihood ratio test, which is the most powerful test, does not tell the difference between zero-inflated Poisson and negative binomial distribution. In this case, a larger sample size would be needed.

4.2. Comparison with R package "KSgeneral"

Package "KSgeneral" [6] supports the computation of p-value for discrete KS test, assuming that parameter values in the null distribution are already known. To conduct a KS test via "KSgeneral", we need to substitute the unknown parameters with their maximum likelihood estimates. Suppose that a random sample $\{X_1, \ldots, X_{1000}\}$ is generated from a zero-inflated negative binomial distribution with parameters $(\phi, r, p) = (0.7, 5, 0.6)$ using the function `sample.zi` as below:

```
library(iZID)
library(extraDistr)
library(KSgeneral)
```

```
set.seed(10086)
x=sample.zi(N=1000,phi=0.7,distri='nb',p=0.6,r=5)
table(x)
x
 0   1  2  3  4  5  6  7 8 10 11 12
726 52 58 59 40 24 22 10 4  1  2  2
## some naive initial estimates of unknown parameters
r=max(x)
p=sum(x>0)/length(x)
n=max(x)+1
alpha1=abs(mean(x)*(mean(x)*(1-mean(x))/var(x)-1))
alpha2=abs((1-mean(x))*(mean(x)*(1-mean(x))/var(x)-1))
```

To test if the simulated data follows from zero-inflated negative binomial distribution:

```
## maximum likelihood estimates of unknown parameters
temp1=nb.zihmle(x,type='zi',r=r,p=p)
temp1
         r         p       phi     loglik
[1,] 5.477278 0.6428991 0.6992482 -1127.7
y1=stepfun(0:max(x), c(0, temp1[3]+(1-temp1[3])*pnbinom(
   0:max(x),size=ceiling(temp1[1]),p=temp1[2])))
## conduct discrete KS test with function disc_ks_test
## in ''KSgeneral''
disc_ks_test(x=x, y=y1, exact=T, tol=1e-08)$p
[1] 0.6051321
## conduct discrete KS test with function dis.kstest
## in ''iZID''
dis.kstest(x,nsim=100,bootstrap=TRUE,distri='zinb',r=r,
   p=p)$pvalue
[1] 0.27
```

From the results above, there is no significant evidence showing that the simulated data comes from distributions other than ZINB. However, a more realistic scenario is that we may also testify other null distributions such as ZIBB, ZIP or ZIBNB.

```
## when the null distribution is ZIBB
temp1=bb.zihmle(x,type='zi',n=n,alpha1=alpha1,
   alpha2=alpha2)
```

```
y1=stepfun(0:max(x), c(0, temp1[4]+(1-temp1[4])*pbbinom(
   0:max(x),size=round(temp1[1]), alpha=temp1[2],
   beta=temp1[3])))
disc_ks_test(x=x, y=y1, exact=T, tol=1e-08)$p
[1] 1
dis.kstest(x,bootstrap=TRUE,distri='zibb',n=n,
   alpha1=alpha1,alpha2=alpha2)$pvalue
[1] 0
## when the null distribution is ZIP
temp1=poisson.zihmle(x,type='zi')
y1=stepfun(0:max(x), c(0, temp1[2]+(1-temp1[2])*ppois(
   0:max(x),lambda=temp1[1])))
disc_ks_test(x=x, y=y1, exact=T, tol=1e-08)$p
[1] 0.4722135
dis.kstest(x,nsim=100,bootstrap=TRUE,distri='zip')$pvalue
[1] 0.47
## when the null distribution is ZIBNB
temp1=bnb.zihmle(x,type='zi',r=r,alpha1=alpha1,
   alpha2=alpha2)
y1=stepfun(0:max(x), c(0, temp1[4]+(1-temp1[4])*pbnbinom(
   0:max(x),size=round(temp1[1]),alpha=temp1[2],
   beta=temp1[3])))
disc_ks_test(x=x, y=y1, exact=T, tol=1e-08)$p
[1] 1
dis.kstest(x,bootstrap=TRUE,distri='zibnb',r=r,
   alpha1=alpha1,alpha2=alpha2)$pvalue
[1] 0
```

Neither function `disc_ks_test` in package "KSgeneral" nor our function `dis.kstest` could distinguish between ZINB and ZIP distributions with the current sample size. As for ZIBB and ZIBNB distributions, the p-value 1 obtained by `disc_ks_test` is apparently misleading, while our `dis.kstest` correctly rejects the two null hypothese with p-values equal to 0.

4.3. A real data example

In this subsection, we use the real dataset "dataCar" from R package "insuranceData" for illustration. The data consists of 67,856 one-year vehicle insurance policies issued in 2014–2015. The variable `number of claims` is a sparse count variable. The goal is to identify the distribution of the variable. Table 1 shows the numbers of claims as well as percentages.

Table 1: Number of claims in dataset "dataCar"

Occurrence	Frequency	Percentage
0	63,232	**93.18%**
1	4,333	6.39%
2	271	0.40%
3	18	0.03%
4	2	0.00%
Total	67,856	100.00%

To check if the data follows any specific discrete distribution, we use `dis.kstest` in our package. A large p-value implies that the data may follow the pre-specified discrete distribution. The following R codes show how to test if the variable `number of claims` follows Poisson, negative binomial, ZIP, or ZINB distribution.

```
library(insuranceData)
library(car)
data(dataCar)
attach(dataCar)
X=dataCar[,4] #Number of claims variable

   dis.kstest(X,nsim=200,bootstrap=TRUE,
      distri='Poisson')$pvalue
   [1] 0.035
   dis.kstest(X,nsim=200,bootstrap=TRUE,distri='nb')$pvalue
   [1] 0
   dis.kstest(X,nsim=200,bootstrap=TRUE,distri='zip')$pvalue
   [1] 0.955
   dis.kstest(X,nsim=200,bootstrap=TRUE,distri='zinb')$pvalue
   [1] 0
   dis.kstest(X,nsim=200,bootstrap=TRUE,distri='zip')$mle_ori
        lambda       phi    loglik
   [1,] 0.1324475  0.4506756  -18052.2
```

The above output implies that the data follows ZIP distribution with estimated parameters $\hat{\phi} = 0.451$, and $\hat{\lambda} = 0.132$. To confirm this conclusion, we simulate a random sample from the ZIP distribution with $\phi = 0.451$, and $\lambda = 0.132$ as follows:

```
   Y=sample.zi(N=length(X),phi=0.4506756,distri='Poisson',
      lambda=0.1324475)
```

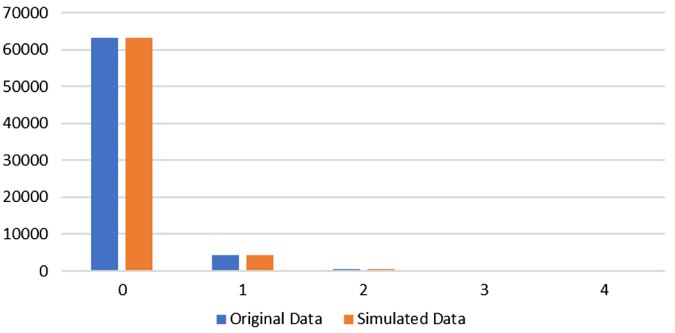

Figure 1: Chart plot to compare the real dataset "dataCar" and the simulated data using the function `sample.zi` with parameters estimated from the real data.

Using R function `table`, we can see that the distributions of the original data X and the simulated data Y match each other very well (see also Figure 1).

```
table(X)
    0    1    2    3  4
63232 4333  271   18  2
table(Y)
    0    1    2    3  4
63172 4371  298   14  1
```

5. Conclusion

In this paper, we introduce a new R package "iZID" which provides the bootstrapped Monte Carlo estimates of p-values of discrete KS tests, as well as a function `model.lrt` to perform a likelihood ratio test when two or more distributions pass the KS test. Besides, "iZID" supports the generation of random deviates from zero-inflated distributions as well as hurdle models, and the computation of maximum likelihood estimates of a large class of models. The implementation of functions `dis.kstest` and `model.lrt` are speeded up by parallel computing via packages "foreach" and "doParallel".

Due to the nature of gamma functions, the optimization of the likelihood function of zero-inflated and hurdle beta binomial and beta negative binomial distributions may not converge. In this circumstance, the results of corresponding standard distributions are returned. We plan to further improve and update the functions in the package for obtaining more robust and reliable sample estimates of parameters.

References

[1] A.A. Metwally, H. Aldirawi, J. Yang, *A review on probabilistic models used in microbiome studies*, Communications in Information and Systems (2018), vol. 18, no. 3, 173–191.

[2] J.P. Boucher, M. Denuit, M. Gui, *Number of accidents or number of claims? An approach with zero-inflated Poisson models for panel data*, Journal of Risk and Insurance (2009), vol. 76, no. 4, 821–846.

[3] N. Ismail, H. Zamani, *Estimation of claim count data using negative binomial, generalized Poisson, zero-inflated negative binomial and zero-inflated generalized Poisson regression models*, Casualty Actuarial Society E-Forum (2013), vol. 41, no. 20.

[4] P. Chen, Q. Liu, F. Sun, *Bicycle parking security and built environments*, Transportation Research Part D: Transport and Environment (2018), vol. 62, 169–178.

[5] M.C. Majo, A. van Soest, *The fixed-effects zero-inflated Poisson model with an application to health care utilization*, Tilburg: Econometrics (2011), vol. 2011-083.

[6] D.S. Dimitrova, V.K. Kaishev, S. Tan, *KSgeneral: Computing P-Values of the K-S Test for (Dis)Continuous Null Distribution*, R package version 0.1.1 (2018), https://CRAN.R-project.org/package=KSgeneral.

[7] H.W. Lilliefors, *On the Kolmogorov-Smirnov test for normality with mean and variance unknown*, Journal of the American statistical Association (1967), vol. 62, no. 318, 399–402.

[8] H.W. Lilliefors, *On the Kolmogorov-Smirnov test for the exponential distribution with mean unknown*, Journal of the American Statistical Association (1969), vol. 64, no. 325, 387–389.

[9] H. Aldirawi, J. Yang, A.A. Metwally, *Identifying Appropriate Probabilistic Models for Sparse Discrete Omics Data*, 2019 IEEE EMBS International Conference on Biomedical & Health Informatics (BHI), Chicago, IL, USA, 2019, 1–4.

[10] H. Aldirawi, J. Yang, *Model Selection and Regression Analysis for Zero-altered or Zero-inflated Data*, Statistical Laboratory Technical Report (2019), vol. 2019-01, University of Illinois at Chicago.

[11] H. Cho, C. Liu, J. Park, D. Wu, *bzinb: Bivariate Zero-Inflated Negative Binomial Model Estimator*, R package version 1.0.3 (2019), https://CRAN.R-project.org/package=bzinb.

[12] E. Balderama, T. Trippe, *hurdlr: Zero-Inflated and Hurdle Modelling Using Bayesian Inference*, R package version 0.1 (2017), https://CRAN.R-project.org/package=hurdlr.

[13] A. Albasi, *mazeinda: Monotonic Association on Zero-Inflated Data*, R package version 0.0.1 (2018), https://CRAN.R-project.org/package=mazeinda.

[14] Y. Croissant, F. Carlevaro, S. Hoareau, *mhurdle: Multiple Hurdle Tobit Models*, R package version 1.1-8 (2018), https://CRAN.R-project.org/package=mhurdle.

[15] I. Waudby-Smith, P. Li, *rbtt: Alternative Bootstrap-Based t-Test Aiming to Reduce Type-I Error for Non-Negative, Zero-Inflated Data*, R package version 0.1.0 (2017), https://CRAN.R-project.org/package=rbtt.

[16] T. Hu, Y. Zhou, *ZIBBSeqDiscovery: Zero-Inflated Beta-Binomial Modeling of Microbiome Count Data*, R package version 1.0 (2018), https://CRAN.R-project.org/package=ZIBBSeqDiscovery.

[17] X. Peng, G. Li, Z. Liu, H. Chen, *ZIBseq: Differential Abundance Analysis for Metagenomic Data via Zero-Inflated Beta Regression*, R package version 1.2 (2017), https://CRAN.R-project.org/package=ZIBseq.

[18] M. Jochmann, *zic: Bayesian Inference for Zero-Inflated Count Models*, R package version 0.9.1 (2017), https://CRAN.R-project.org/package=zic.

[19] M. Yang, G. Zamba, J. Cavanaugh, *ZIM: Zero-Inflated Models (ZIM) for Count Time Series with Excess Zeros*, R package version 1.1.0 (2018), https://CRAN.R-project.org/package=ZIM.

[20] Z. Xu, Y. Liu, *ziphsmm: Zero-Inflated Poisson Hidden (Semi-)Markov Models*, R package version 2.0.6 (2018), https://CRAN.R-project.org/package=ziphsmm.

[21] R. Calaway, Microsoft Corporation, S. Weston, D. Tenenbaum, *doParallel: Foreach Parallel Adaptor for the 'parallel' Package*, R package version 1.0.11 (2017), https://CRAN.R-project.org/package=doParallel.

[22] R. Calaway, Microsoft, S. Weston, *foreach: Provides Foreach Looping Construct for R*, R package version 1.4.4 (2017), https://CRAN.R-project.org/package=foreach.

[23] T. Wolodzko, *extraDistr: Additional Univariate and Multivariate Distributions*, R package version 1.8.11 (2019), https://CRAN.R-project.org/package=extraDistr.

Lei Wang
Center for Applied Statistics, School of Statistics
Renmin University of China
Beijing, 100872, China
E-mail address: slimewanglei@163.com

Hani Aldirawi
Department of Mathematics, Statistics, and Computer Science
University of Illinois at Chicago
Chicago, IL 60607, USA
E-mail address: haldir2@uic.edu

Jie Yang
Department of Mathematics, Statistics, and Computer Science
University of Illinois at Chicago
Chicago, IL 60607, USA
E-mail address: jyang06@uic.edu
URL: http://www.math.uic.edu/~jyang06/

Received September 9, 2019