Journal of Statistical Software

MMMMMM YYYY, Volume VV, Issue II. *doi: 10.18637/jss.v000.i00*

hurdlr: An R Package for Zero-inflated and Extremely Over-dispersed Count Data

Earvin Balderama
Loyola University Chicago

Taylor Trippe
Loyola University Chicago

Abstract

The abstract of the article.

Keywords: keywords, comma-separated, not capitalized, Java.

1. Introduction

Count data is one of the most common and important forms of data encountered in every field of study. While some count data can be modeled easily with binomial or poisson likelihoods, many times the data is gathered such that a disproportionate amount of zeros appear relative to non-zeros. For these, a class of models called zero-inflated models were created with the reasoning that the counts came from two separate data-generating mechanisms; one that generated counts according to some discrete probability distribution (such as Poisson), and another that generated the excess amount of zeros.

In this paper, we present an R package dedicated to modeling zero-inflated count data, with an emphasis on developing new methods to account for both zero-inflation and extreme over-dispersion simultaneously. The models in this package utilize Bayesian estimation techniques. Parameter estimation under a Bayesian framework is suitable because of the model's hierarchical structure. Model parameters were given uninformative priors and updated one-at-a-time using a Markov chain Monte Carlo (MCMC) algorithm.

2. Models for count data

2.1. Zero-inflated models

Zero-inflated models are widely used for count data exhibiting a significantly higher amount of zeros that would otherwise be observed given some typical count distribution. These are simply two-component mixture models that can separate the excess zeros in the data from the underlying (parametric) count distribution. A common choice for the count distribution is Poisson, but is typically replaced with negative binomial if there is over-dispersion in the data.

2.2. Hurdle models

Hurdle models are similar to zero-inflated models in that they also account for the high occurrence of zeros in data. The difference is that for hurdle models, *all* zero-count observations are assumed to have been generated by a mechanism separate of the non-zeros. We begin by trying to fit the negative binomial hurdle model, with the following distributional components:

$$h_1(y_i|m_i, r) = \begin{cases} p_i & \text{if } y_i = 0, \\ (1 - p_i) \cdot f(y_i|m_i, r) & \text{if } y_i > 0, \end{cases} \tag{1}$$

where for each observation i, $0 \leq p_i \leq 1$ and $f(y_i|m_i, r)$ is the negative binomial probability function, with mean m_i and overdispersion parameter r, truncated at 0.

3. Double-hurdle models

In some cases, few but very large count values causes the data to be so extremely overdispersed that a simple negative binomial hurdle model proves to be inadequate.

Properly fitting such an extremely right-skewed distribution is the goal of the double-hurdle model. In this model, a second hurdle is created to represent the transition from a distribution of "typical" counts to a distribution of larger counts. Just as in the hurdle model, we assume separate data-generating processes for each of the components. The first hurdle represents the divide between the zero-counts and the "typical" non-zero counts. Larger counts can be thought of as coming from a heavy-tailed distribution g if the second hurdle is crossed. The double-hurdle model has the following form for observation i:

$$h_2(y_i|\boldsymbol{\theta}) = \begin{cases} p_i & \text{if } y_i = 0, \\ (1 - q_i)(1 - p_i) f(y_i|m_i, r) & \text{if } 1 \leq y_i < \mu, \\ q_i(1 - p_i) g(y_i|\mu, \sigma, \xi) & \text{if } y_i \geq \mu, \end{cases} \tag{2}$$

where $\boldsymbol{\theta}$ is the (possibly) vector-valued set of model parameters, p_i is the probability that observation i is governed by a zero-generating process, and q_i is the probability that observation i is drawn from the large-count distribution conditional on it being non-zero. The typical-count component is left-truncated at 0 and right-truncated at μ to maintain separability in the likelihood then normalized to ensure that each component is a density.

Balderama et al 2016 considered the Generalized Pareto distribution (GPD) to describe the third component of the count data. If $y \sim \mathsf{GPD}(\mu, \sigma, \xi)$, then y has density function

$$g(y|\mu, \sigma, \xi) = \frac{\sigma^{1/\xi}}{\Big(\sigma + \xi \times (y - \mu)\Big)^{\frac{1}{\xi}+1}} \tag{3}$$

for $y \geq \mu$ and $\xi \neq 0$. Since we are dealing with count data, we use a discretized version of the GPD distribution function, defined by taking the difference between the cumulative densities at $y + 0.5$ and $y - 0.5$.

Notice that when $\mu = \infty$, (2) simplifies to (1). Also notice that setting $\mu = 1$ essentially removes the second component, thus all non-zero counts are modeled under the large-count distribution. Balderama et al 2016 defined this as the GPD-hurdle model, a new single-hurdle alternative to the negative binomial distribution for extremely right-skewed count data.

4. Implementing hurdlr

4.1. Zero-inflated modeling

The **hurdlr** package supports zero-inflated modeling with the functions `zero_nb` and `zero_poisson` which, depending on usage, applies a negative binomial or Poisson distribution to count data with an excess of zeros. Function usage is denoted:

```
zero_nb(y, x, size, a, b, mu.start, beta.prior.mean, beta.prior.sd, iters, burn,
nthin, plots, progress.bar)
```

or

```
zero_poisson(y, x, a, b, lam.start, beta.prior.mean, beta.prior.sd, iters, burn,
nthin, plots, progress.bar)
```

with the arguments defined as follows.

`y` numeric response vector.

`x` numeric predictor matrix.

`size` size (r) parameter for NB(r, μ) likelihood distributions.

`a` shape parameter for Gamma(a, b) prior distributions.

`b` rate parameter for Gamma(a, b) prior distributions.

`size` size (r) parameter for NB(r, μ) likelihood distributions.

`mu.start, lam.start` initial value of distributional mean for NB(r, μ) or Pois(λ) likelihood, respectively.

`beta.prior.mean` mean (μ) for Normal(μ, σ^2) prior distributions for coefficient estimation.

`beta.prior.sd` standard deviation (σ) for Normal(μ, σ^2) prior distributions for coefficient estimation.

`iters` number of iterations for the Markov chain algorithm.

`burn` numeric burn-in length.

`nthin` numeric thinning rate.

`plots` logical operator. `TRUE` to print plots.

`progress.bar` logical operator. `TRUE` to print progress bar.

Both `zero_nb` and `zero_poisson` functions output a list with the elements:

4.2. Hurdle modeling

The **hurdlr** package allows for hurdle modeling through the `hurdle` function. The `hurdle` function supports both single- and double-hurdle models as well as a variety of distributions for both "typical" distribution f(y|m, r) and the distribution for "extreme" values g(y|μ, σ, ξ)

Forms of usage

A `hurdle` function has two forms of usage; one to fit a single-hurdle model, truncated at zero, and the other to fit a double-hurdle model with an additional hurdle at some 'extreme' threshold ψ. The two usages are, respectively:

```
hurdle(y, x, Inf, dist, control, iters, burn, nthin, plots, progress.bar)
```

and

```
hurdle(y, x, hurdle, dist, dist.2, control, iters, burn, nthin, plots, progress.bar)
```

with the arguments defined as follows.

`y` numeric response vector.

`x` optional numeric predictor matrix.

`hurdle` numeric threshold (ψ) for 'extreme' observations of two-hurdle models. `Inf` for one-hurdle models.

`dist` character specification of response distribution. The function supports a choice of Poisson, negative binomial, log-normal, and generalized Pareto distributions.

`dist.2` character specification of response distribution for 'extreme' observations of two-hurdle models. The function supports a choice of Poisson, negative binomial, log-normal, and generalized Pareto distributions. `"none"` for one-hurdle models.

`control` list of parameters for controlling the fitting process, specified by a `hurdle_control()` assignment.

`iters` number of iterations for the Markov chain algorithm.

`burn` numeric burn-in length.

`nthin` numeric thinning rate.

`plots` logical operator. `TRUE` to print plots.

`progress.bar` logical operator. `TRUE` to print progress bar.

Additional control parameters

Many of a `hurdle` function's fitting parameters are defined through a `hurdle_control` list. This includes distributional priors, hyperparameters, and MCMC tuning parameters. The `hurdle_control` has a usage of:

```
hurdle_control(a, b, size, beta.prior.mean, beta.prior.sd, beta.tune, pars.tune,
lam.start, mu.start, sigma.start, xi.start)
```

with the arguments defined as follows.

`a` shape parameter for Gamma(a, b) prior distributions.

`b` rate parameter for Gamma(a, b) prior distributions.

`size` size (r) parameter for NB(r, μ) likelihood distributions.

`beta.prior.mean` mean (μ) for Normal(μ, σ^2) prior distributions for coefficient estimation.

`beta.prior.sd` standard deviation (σ) for Normal(μ, σ^2) prior distributions for coefficient estimation.

`beta.tune` MCMC tuning for regression coefficient estimation.

`pars.tune` MCMC tuning for parameter estimation.

`lam.start, mu.start, sigma.start, xi.start` initial value(s) for parameter(s) of 'extreme' observations distribution.

The output of the `hurdle` function is a list with the elements:

4.3. An example of use

[seabird data info]

The following code illustrates the use of `hurdle` to model the seabird data of species Sooty Shearwater. As indicated by Table 1, this data is both zero-inflated and exceptionally over-dispersed, and may best be fit using a GPD-hurdle model. The negative binomial distribution is specified for the "typical" observed counts, and "extreme" observations are modeled by the discrete generalized Pareto distribution. The hurdle to delineate "extreme" values is set as the 90th percentile of the non-zero counts (in this case, 14). The MCMC algorithm is run for 25,000 iterations with a 5,000 burn-in and thinning interval of 5. Uninformative priors and tuning parameters are used for all pieces of the model.

Avian Counts: Sooty Shearwater	
Count	**Frequency**
0	33503
1 - 10	412
11 - 100	88
101 - 500	16
501 +	3

Table 1: Frequency seabird counts for the species Sooty Shearwater. Non-zero counts have a range of 1 to 700 as well as a mean and median of 9.55 and 2, respectively. The resulting distribution is extremely right-skewed.

How do I input/output code???

```
#Load in seabird (species "sosh")
load(url("http://math.luc.edu/~ebalderama/stat388bayes_resources/sb/data/sosh.RData"))

#Set a value for the second hurdle
hurd <- quantile(Y[Y!=0], prob = 0.9)

#Set control parameters
control <- hurdle_control(a = 1, b = 1, size = 1, beta.prior.mean = 0, beta.prior.sd = 10,

#Run function for a double-hurdle model
d.hurdle <- hurdle(y = Y, x = X[,-1], hurd = hurd, dist = "nb", dist.2 = "gpd", control =
```

5. Discussion

References

Affiliation:

Earvin Balderama
Department of Mathematics and Statistics
Loyola University Chicago
1032 W Sheridan Rd.,
Chicago, IL 60660, USA
E-mail: ebalderama@luc.edu
URL: http://math.luc.edu/~ebalderama/

Taylor Trippe
Department of Mathematics and Statistics
Loyola University Chicago
1032 W Sheridan Rd.,
Chicago, IL 60660, USA
E-mail: ebalderama@luc.edu
URL: http://math.luc.edu/~ebalderama/

Journal of Statistical Software http://www.jstatsoft.org/
published by the Foundation for Open Access Statistics http://www.foastat.org/
MMMMMM YYYY, Volume VV, Issue II *Submitted:* yyyy-mm-dd
doi:10.18637/jss.v000.i00 *Accepted:* yyyy-mm-dd