Journal of Statistical Software

March 2016, Volume 69, Issue 6. *doi: 10.18637/jss.v069.i06*

Mean and Variance Modeling of Under- and Overdispersed Count Data

David M. Smith
Truven Health Analytics

Malcolm J. Faddy
Queensland University of Technology

Abstract

This article describes the R package **CountsEPPM** and its use in determining maximum likelihood estimates of the parameters of extended Poisson process models. These provide a Poisson process based family of flexible models that can handle both underdispersion and overdispersion in observed count data, with the negative binomial and Poisson distributions being special cases. Within **CountsEPPM** models with mean and variance related to covariates are constructed to match a generalized linear model formulation. Use of the package is illustrated by application to several published datasets.

Keywords: Poisson distribution, underdispersion, overdispersion, negative binomial distribution, extended Poisson process models.

1. Introduction

Modeling using extended Poisson process models (EPPMs) was originally developed in Faddy (1997), where the construction of discrete probability distributions having very general dispersion properties was described. The Poisson and negative binomial distributions are special cases of this modeling which includes both underdispersion and overdispersion relative to the Poisson, with the negative binomial having the most extreme level of overdispersion within the EPPM family. Faddy and Smith (2008) incorporated covariate dependence in the mean via a reparameterization using an approximate form of the mean; and Faddy and Smith (2011) extended this to incorporate covariate dependence in the dispersion, this being achieved by a reparameterization using an approximate form of the variance. The supplementary material for Faddy and Smith (2011) contained R code illustrating fitting these models. This R code has been extended and generalized to have inputs and outputs more akin to those of a generalized linear model (GLM) as in the R function `glm` and the R function `betareg` (Cribari-Neto and Zeileis 2010, Grün, Kosmidis, and Zeileis 2012). Both Hilbe (2011) and Hilbe (2014) have

comments about a software package for EPPMs being developed in the R system (R Core Team 2016); the package **CountsEPPM** (Smith and Faddy 2016) whose use is described in this article is that software. This article relates to version 2.1 of the package submitted to the Comprehensive R Archive Network (CRAN) in 2016. The important differences between versions 1.0 and 2.0, 2.1 involve the use of arguments `formula` and `data` when running the function `CountsEPPM`.

2. Models

2.1. Extended Poisson process models (EPPMs)

Only a summary of these models is given. More details, in particular of their theoretical background and development, can be found in Faddy (1997). The defining equation is:

$$\boldsymbol{p} \;=\; (1\ 0\ \cdots\ 0)\exp(\boldsymbol{Q}) \tag{1}$$

where $\boldsymbol{p}$ is a row vector of probabilities and $\boldsymbol{Q}$ is the matrix:

$$\boldsymbol{Q} \;=\; \begin{bmatrix} -\lambda_0 & \lambda_0 & 0 & \cdots & 0 \\ 0 & -\lambda_1 & \lambda_1 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & \cdots & -\lambda_n \end{bmatrix}$$

with the λ parameters being rates of an extended Poisson process, and $\exp(\boldsymbol{Q})$ is the matrix exponential function. Constant λ's here correspond to the simple Poisson process with the distribution of the number of events in a time interval of length t being Poisson with mean λt. The extension allowing the λ's to depend on i, the cumulative number of events occurring, was first suggested in Faddy (1997). The probability p_i of obtaining a count of size i ($i = 0, 1, \ldots, n$) in a time interval of unit length is the $(i+1)$th element of $\boldsymbol{p}$, so discrete probability distributions can be constructed from a sequence of λ's.

A three parameter function for the λ_i sequence:

$$\lambda_i \;=\; a(b \;+\; i)^c, \text{ for } i = 0, 1, 2, \ldots, \text{ where } a > 0,\ b > 0 \text{ and } c \leq 1 \tag{2}$$

will be used to produce discrete distributions including the Poisson ($c = 0$) and negative binomial ($c = 1$) as special cases with the probability mass function for the negative binomial distribution being in the form:

$$\mathrm{P}\{r = i\} \;=\; \left(\begin{array}{c} b+i-1 \\ b-1 \end{array} \right) \{1 \;-\; \exp(-a)\}^i \{\exp(-a)\}^b, \quad \text{for } i = 0, 1, \ldots \infty.$$

Generally, the parameter $c > 0$ in Equation 2 results in distributions overdispersed relative to the Poisson distribution, and $c < 0$ distributions underdispersed relative to the Poisson. The mean and variance of distributions from EPPMs defined by Equation 2 generally have to be determined numerically directly from the probability distribution of Equation 1 using a suitable truncation (n); however, approximations are available as in Faddy (1997), Faddy

and Smith (2008), Faddy and Smith (2011):

$$\text{mean} \approx m = b\left[\left(1+\frac{a(1-c)}{b^{1-c}}\right)^{\frac{1}{1-c}} - 1\right] \tag{3}$$

$$\text{variance} \approx v = \frac{b\left(\frac{m}{b}+1\right)\left[1-\left(\frac{m}{b}+1\right)^{(2c-1)}\right]}{1-2c} \; . \tag{4}$$

The mean approximation can be used to obtain a in Equation 2 as:

$$a = \frac{(m+b)^{1-c}-b^{1-c}}{1-c} \quad \left[= \log\left(1+\frac{m}{b}\right) \text{ for } c=1\right] \; . \tag{5}$$

Covariates $\boldsymbol{x}$ can then be incorporated using a log link:

$$\log(m) = \text{linear predictor}(\boldsymbol{x}) \; . \tag{6}$$

Covariate dependence in the variance can be incorporated by having v as a function of another linear predictor and then solving the equation:

$$\frac{v}{m+b} = \frac{\exp\{(2c-1)\log(m/b+1)\}-1}{2c-1} \tag{7}$$

for c in terms of v, m and b. Such covariate dependence in the variance permits the modeling of datasets where some subsets exhibit underdispersion and others overdispersion. As the right-hand-side of Equation 7 is a convex increasing function of c, Newton-Raphson iteration (initial value $c_0 = 1$) will converge to the solution. If a solution $c > 1$ is indicated, that parameterization is inadmissible since the resulting probability distribution will be improper, so the parameter space is bounded by $c \leq 1$. The above approximations are only used to derive the reparameterization of Equations 5 and 7; exact calculation of means and variances are done numerically from the probability distribution of Equation 1 to minimize the effect of this approximation. This means in practice that for $c \neq 0, 1$ the actual relationship between log(mean) and the covariates $\boldsymbol{x}$ is not quite linear, but Toscas and Faddy (2003) noted that in their example any departure from linearity was almost imperceptible. Use of exact means (and variances) in this way overcomes, to a great extent, the bias in the approximations noted by Grunwald, Bruce, Jiang, Strand, and Rabinovitch (2011).

The parameter b in Equation 2 is something of a nuisance parameter which is often poorly estimated, and better estimated as $\log(b)$ (Faddy and Smith 2011), so the default model is specified by the two parameters a and c in Equation 2 dependent on covariates with b always a single scalar-valued constant. This model fits well to data overdispersed with respect to the Poisson distribution but Faddy and Smith (2011) found that poor convergence to the maximum can occur for underdispersed data if the parameter b becomes large and the hessian at an apparent maximum is poorly conditioned; in such circumstances a limiting form of the λ_i's of Equation 2 for $b \to \infty$ with $a = \alpha b^{-b/\beta}$ and $c = b\beta$, and a special case of Equation 2 is used (Faddy and Smith 2011):

$$\lambda_i = \alpha \exp(\beta i), \; i = 0, 1, 2, \ldots, \tag{8}$$

with $\beta < 0$, corresponding to $c < 0$ in (2) and underdispersion; the approximate mean and variance are given by:

$$m = \frac{-\log(1 - \alpha\beta)}{\beta} \tag{9}$$

$$\text{and } v = \frac{\exp(2\beta m) - 1}{2\beta} \tag{10}$$

with solutions of Equations 9 and 10 achieving the reparameterization in terms of the approximate mean and variance.

When fitting mean and variance models of Equations 3 and 4 (or Equations 9 and 10) use of a log link function to relate mean and variance to the linear predictors means that the scale factor (variance/mean) can be modeled rather than the variance by simply including log(mean) as an offset in the linear predictor for the variance. An option to do this is included.

An alternative formulation only involving the mean model is available Faddy and Smith (2008), where only the proportionality constant a in Equation 2 (or α in Equation 8) is dependent on the covariates $\boldsymbol{x}$; this does not actually quantify the mean but is based on increasing (decreasing) values of a leading to increased (decreased) rates λ_i's of the extended Poisson process and hence increased (decreased) counts on average.

2.2. Models other than EPPMs

Other distributional models exist that can handle under- or overdispersed count data and for specific data sets these may fit better than the EPPMs. This is more likely for overdispersed data because there are many alternative models. The most commonly used of these alternatives for overdispersed data is the negative binomial. Hilbe (2011) is a comprehensive overview of the various forms and manifestations of the negative binomial with R code included. Ridout and Besbeas (2004) describe various models for handling count data that show underdispersion with respect to the Poisson distribution, focusing on exponentially weighted Poisson distributions but also looking at others including a less general EPPM corresponding to Equation 2 with $b = 1$ which they label as a changing birth rate (CBR) model. Sáez-Castillo and Conde-Sánchez (2013) describe hyper-Poisson regression models for count data that can handle under- and overdispersion with both mean and dispersion dependent on covariates.

3. Description of the functions

The main function of the package is also named `CountsEPPM` and is focused on models with two covariate dependences linked to the mean and variance. The input into the function is a formula involving a single response variable and one or two formulae related to the mean and variance models. Although the input formula involves a single response variable, the actual model fitting has a list of frequency distributions `list.counts` in place of the response variable, which is either input or constructed from the input data according to whether a `list` or a `data.frame` is input. For all models the GLM link function between response variable (mean, variance) and linear predictor of covariates is log; the log of parameter b is also used but the parameter c of Equation 2 is untransformed. The full three parameter version of Equation 2 has been labeled the Faddy distribution as in Grunwald *et al.* (2011). Because of possible issues with the parameter b, in version 2.0 variants of the models where b is fixed

have been included in the lists of models. This enables profile likelihoods to be produced for this parameter.

Nash (2014) is a recent reference on optimization using R functions and contains information on, and insights into, the methods used. All models are fitted to the data using maximum likelihood, the optimization method used being either the R function `optim` (simplex method of Nelder and Mead 1967), or the R function `nlm` (a Newton method using derivatives). A facility to change options for these two functions through use of the argument `control` was introduced into version 2.0. The elements of this argument are the options for `optim` or `nlm` as described in R Core Team (2016). The default values set within `CountsEPPM` are `fnscale = -1`, `trace = 0`, `maxit = 1000` for `optim`, and for `nlm` they are `fscale = 1`, `print.level = 0`, `stepmax = 1`, `gradtol = 1e-8`, `steptol = 1e-10`, `iterlim = 500`. Although for most data sets the two functions `optim` and `nlm` give similar results in terms of log(likelihood) and parameter estimates, etc., some results may be a little different depending on particular features of the data set. The function `optim`, being an implementation of the simplex method, is robust to discontinuities in the log(likelihood) surface. However, it is slow to converge. In contrast the function `nlm` makes use of derivatives, in this case numerical derivatives, resulting in faster convergence, but there is a reliance on the log(likelihood) surface being smooth, i.e., no sudden changes in derivative values. Only `nlm` makes use of derivatives in the actual model fitting, but both functions have options to calculate a hessian matrix which can be used to produce standard errors for the parameter estimates. These function options were used in version 1.0 but not in version 2.0 where the calculation of the numerical derivatives and hessians use the functions `grad` and `hessian` from the package **numDeriv** (Gilbert and Varadhan 2015). This replacement was made because the derivatives are more accurately calculated resulting in better model fitting and better conditioned hessians. However, as Nash (2014, p. 131) states, **numDeriv** (Gilbert and Varadhan 2015) takes a longer time than alternative central difference approximations. The occurrence of `NA` in the vector of standard errors is an indication of problems with the model fitting, possibly caused by an inappropriate model. This is particularly so when all estimates of parameter standard errors are `NA` which results when the hessian matrix can not be be inverted due to its determinant being zero or it being otherwise ill-conditioned. Having derivatives available means that they can be reviewed, together with the hessian, at the final parameter estimates to evaluate whether maximum likelihood estimates have been attained.

In versions 2.0 and 2.1 `data` can be either a `list` or a `data.frame` whereas in version 1.0 it could only be a `list`. Also, in versions 2.0 and 2.1 the response variable in `formula` is a vector if a `data.frame` is input or a `list` if a `list` is input. In version 1.0 it was required to use the names `mean.obs` and `variance.obs` for the response variables in `formula`. In versions 2.0 and 2.1the response variables `mean.obs` and `variance.obs` are constructed within `CountsEPPM` prior to being used to fit models. The structure of the model fitting within `CountsEPPM` is unchanged from version 1.0. The R package **Formula** of Zeileis and Croissant (2010) is used to extract model information from the `formula` input to `CountsEPPM`. To avoid repeated extractions within subordinate functions, the extraction of model information used in the model fitting, such as `covariates.matrix.mean`, is only done once within `CountsEPPM`. Function arguments for offsets for the mean and variance, as well as arguments for the lower and upper truncation values for truncated distributions have been included. In version 2.1 the names of two functions have been changed to `summary.CountsEPPM` and `distribution.CountsEPPM` from their version 2.0 names. Also, `summary.CountsEPPM` is now an S3 method and a `logLik`

Argument	Description	Default
`formula`	a single response variable & paired formulae Zeileis and Croissant (2010)	
`data`	`data.frame` or `list`, i.e., response variable list of frequency distributions	
`model.type`	`"mean only"`	`"mean and variance"`
	`"mean and variance"`	
if `model.type = "mean only"` *(only a in Equation 2 modeled)*		
`model`	`"Poisson"`	
	`"negative binomial"`	
	`"negative binomial fixed b"`	
	`"Faddy distribution"` Equation 2	
	`"Faddy distribution fixed b"`	
if `model.type = "mean and variance"` *(both modeled)*		
`model`	`"general"` as Equations 3 and 4	`"general"`
	`"general fixed b"`	
	`"limiting"` as Equations 9 and 10	
`offset`	one or two element list of offset vectors	vectors of 0's
`initial`	vector of initial values for parameters, means first followed by the variances and/or parameters c, $\log(b)$	Poisson `glm` output augmented by 0's for other parameters
`ltvalue`	lower truncation value (excluded)	`NA`
`utvalue`	upper truncation value (excluded)	`NA`
`optimization.method`	`"optim"` or `"nlm"`	`"optim"`
`control`	list of control parameters, `"optim"` or `"nlm"`	see text for more detail
`scale.factor.model`	`"yes"` if scale factor modeled `"mean and variance"` models only	`"no"`
`fixed.b`	value b is fixed at	`NA`

Table 1: Arguments of `CountsEPPM`.

method has also been added. In addition, the object returned by `CountsEPPM` has been assigned a class of `"CountsEPPM"`, and the base package **stats** imported, which enables use of functions such as `AIC`.

As iteration is involved in the model fitting, initial estimates of the parameters are needed. These can optionally be provided in the vector `initial`. Within `CountsEPPM`, if `initial` is unset, a Poisson model is fitted using `glm` and the estimates from that fit are used to provide estimates for the parameters of the mean linear predictor. If the variance is also being modeled the initial estimates of the parameters of the variance linear predictor are set to the same values as those for a mean model recognising that for the Poisson distribution the variance equals the mean. The initial value of $\log(b)$ is set to zero. The matrix exponential function used for calculating the probabilities of Equation 1 is that of the package **expm** of Goulet, Dutang, Maechler, Firth, Shapira, and Stadelmann (2015) which depends on the package **Matrix** of Bates and Maechler (2016). The code for the main analysis function is

Component	Description
model.type	"mean only"
	"mean and variance"
model	"Poisson"
	"negative binomial"
	"negative binomial fixed b"
	"Faddy distribution" Equation 2
	"Faddy distribution fixed b"
	"general" Equations 3 and 4
	"general fixed b"
	"limiting" Equations 9 and 10
covariates.matrix.mean	matrix of covariates for the mean
covariates.matrix.variance	matrix of covariates for the variance
offset.mean	offset vector for the mean
offset.variance	offset vector for the variance
ltvalue	lower truncation value (excluded)
utvalue	upper truncation value (excluded)
scale.factor.model	"yes" if scale factor not variance modeled; "mean and variance" models only
fixed.b	value b is fixed at
estses	a data frame with columns (name, estimates, se)
vnmax	a vector of maximum counts in each of the grouped data vectors
loglikelihood	the loglikelihood value
mean.obs	a vector of the mean values of the grouped data vectors
variance.obs	a vector of the variance values of the grouped data vectors

Table 2: Components of object returned by CountsEPPM.

```
CountsEPPM(formula, data, model.type, model, offset, initial, ltvalue,
  utvalue, optimization.method, control, scale.factor.mode, fixed.b)
```

with details of the arguments given in Table 1 together with defaults if any.

CountsEPPM prints out statements as to the type of data input; the optimization method used with its return code and associated message; together with the number of function calls (optim) or iterations (nlm); and warning messages. Further details of optim and nlm, their return codes and associated messages are available in R Core Team (2016). CountsEPPM also returns an object of class "CountsEPPM" summarizing the model fit, the components of which are given in Table 2.

Two generic (S3) methods and a function are available. The summary method prints out a summary of model information, estimates of parameters with their standard errors, log-likelihood, etc., in GLM-like form. The logLik method simply extracts the log(likelihood) object from the object output by CountsEPPM, enabling Akaike's information criterion (AIC) and other log(likelihood) based to be easily used, i.e., by functions such as AIC . The function distribution.CountsEPPM produces an object consisting of the fitted means, variances and, optionally, total probabilities and/or parameters of the Faddy distributions. Both methods

Argument	Description	Default
`output.fn`	output object from `CountsEPPM`	
`print.probabilities`	`"yes"` if desired to print out probabilities	`"no"`
`FDparameters`	`"yes"` if desired to print out the parameters of the Faddy distribution Equation 2	`"no"`

Table 3: Arguments of `distribution.CountsEPPM`.

Components	Description
`means`	a data frame consisting of paired means and variances where `.obs` refers to observed where `.prob` refers to fitted probability distributions where `.par` refers to approximating Equations 3 and 4 or 9 and 10 and total fitted probabilities for each vector of grouped counts
`probabilities`	probabilities for each grouped count vector
`FDparameters`	parameters of the Faddy distribution Equation 2

Table 4: Components of object returned by `distribution.CountsEPPM`.

and the function have the output object from `CountsEPPM`, named `output.fn` here, as an input object.

Details of the input arguments of `distribution.CountsEPPM` are given in Table 3 together with defaults. Table 4 gives details of the components of the object returned by `distribution.CountsEPPM`.

As the vectors of frequency distributions are only required to be of length the maximum observed count value +1, this is how they are set up. However, the fitted models can have probability masses at counts greater than these maximum counts. A component of the output object from `CountsEPPM`, i.e., `output.fn$vnmax` is a vector of the maximum observed counts. The object returned by `distribution.CountsEPPM` includes the total probabilities for the fitted models. If the total probabilities over these ranges for the fitted models indicate inadequate cover, i.e., totals appreciably less than 1, which impacts on the exact means and variances calculated, the values in `output.fn$vnmax` can be increased and `distribution.CountsEPPM` run again to improve the calculated means, variances and cover.

4. Examples

The examples illustrate various ways in which **CountsEPPM** can be used to produce informatative analyses. For the first three examples `data` is a `list` where the dependent variable of counts is a `list` of vectors of frequency distributions, whereas for the last two `data` is a `data frame`. Significant time savings can be made by using the `list` form of input where applicable. The fitting of models and estimation of their parameters can be sensitive to the initial estimates and method of estimation chosen, with flatness of likelihood surface possible, particularly with respect to parameter b. It is recommended that analyses be run more than once using different initial estimates and optimization methods.

4.1. Number of young at varying effluent concentrations data

These Ceriodaphnia dubia data were used as an example by Faddy and Smith (2011). Ceriodaphnia dubia are water fleas used to test the impact of effluents on water quality. The data, originally from Bailer and Oris (1997), are counts of young at varying effluent concentrations, and are in `list` of frequencies and variables form. The defaults for `model.type` of `"mean and variance"`, and `model` of `"general"`, are used. The code given below is for three runs using `optim` followed by one using `nlm` with this last run being primarily to obtain the derivatives (gradients) at the final estimates. Although during these runs the estimate of $\log(b)$ changed sign, its standard error is relatively large and the estimates of the other parameters had relatively small changes in value (maximum change of 0.4 in the intercept of of log(variance).

```
R> data("ceriodaphnia.group")
R> names.parameters <- c("log(mean) Intercept", "log(mean) dose",
+    "log(mean) dose^2", "log(variance) Intercept", "log(variance) dose",
+    "log(variance) dose^2", "log(b)")
R> for (i in 1:3) { if (i == 1) {
+    output.fn <- CountsEPPM(number.young ~ 1 + vdose + vdose2 |
+      1 + vdose + vdose2, data = ceriodaphnia.group,
+      control = list(maxit = 2000))
+    output.fn$estses[[1]] <- names.parameters
+  } else {
+    initial <- output.fn$estses[[2]]
+    names(initial) <- output.fn$estses[[1]]
+    output.fn <-  CountsEPPM(number.young ~ 1 + vdose + vdose2 |
+      1 + vdose + vdose2, data = ceriodaphnia.group, initial = initial,
+      control = list(maxit = 2000))
+  }}
R> CountsEPPM(number.young ~ 1 + vdose + vdose2 | 1 + vdose + vdose2,
+    data = ceriodaphnia.group, initial = initial,
+    optimization.method = "nlm", control = list(print.level = 1))
```

with the final run using `nlm` giving a log(likelihood) value of −152.1356 after 50 function calls; a return code of `return code 2 successive iterates within tolerance, current iterate is probably solution` ; and gradient vector as follows.

```
Gradient:
[1]  2.310945e-05  8.823889e-07  9.203419e-03 -1.598838e-03  7.425108e-04
[6] -3.418002e-04 -1.266544e-03
```

Details of the model can be printed out including the parameters of the Faddy distribution.

```
R> summary.CountsEPPM(output.fn)
R> output.distribution <- CountsEPPM.distribution(output.fn,
+    output.FDparameters = "yes")
R> print(output.distribution$FDparameters)
```

```
Model type: mean and variance
Model     : general
Link for mean         : log
Link for variance     : log
variance model fitted
Parameter estimates and se's
                   name    Estimate  Std. Error     z value        Pr(>|z|)
    log(mean) Intercept  3.14042179 0.086742941 36.20377363 0.000000e+00 ***
         log(mean) dose  0.17346783 0.030550787  5.67801504 1.362667e-08 ***
       log(mean) dose^2 -0.01961752 0.002522857 -7.77591264 7.549517e-15 ***
log(variance) Intercept  4.43774936 0.484210655  9.16491471 0.000000e+00 ***
     log(variance) dose -0.49214832 0.232549396 -2.11631733 3.431783e-02 *
   log(variance) dose^2  0.02770728 0.016657750  1.66332680 9.624704e-02 .
                log(b) -0.14123841 2.673364438 -0.05283171 9.578660e-01

 Signif. codes: "***" 0.001 "**" 0.01 "*" 0.05 "." 0.1

 Log-likelihood: -152.1356 on 7 Df

      out.va    out.vb     out.vc
1   7.650481 0.8673864  0.5185416
2  18.485093 0.8673864  0.1830752
3  56.574580 0.8673864 -0.2035973
4 417.549439 0.8673864 -0.9186527
5   6.796195 0.8673864  0.2157152
```

The above parameter estimates agree with those in Faddy and Smith (2011) to two decimal places, except for $\log(b)$ which is relatively poorly estimated.

4.2. Lüning et al. data

These data are from Lüning, Sheridan, Ytterborn, and Gullberg (1966) and are in the form of a `list` of frequencies and variables. The number of trials are stated in Lüning *et al.* (1966) to be both lower and upper truncated at 4 and 11 respectively, so the data are for counts of 5 to 10. Default initial values are used for fitting the default `general` model of Equations 3 and 4. The scale factor model was used because it fitted better than the variance model.

```
R> data("Luningetal.litters")
R> output.fn  <- CountsEPPM(number.trials ~ 0 + fdose |  0 + fdose,
+    Luningetal.litters, ltvalue = 4, utvalue = 11,
+    control = list(maxit = 2000), scale.factor.model=  "yes")
R> CountsEPPM.summary(output.fn)
```

The return code from **CountsEPPM** is `convergence 0 successful` after 1070 function calls, and a warning message `In sqrt(diag(varcov)) :  NaNs produced`. Output details of the final model follow.

```
Model type: mean and variance
Model     : general
```

```
Link for mean         : log
Link for scale factor : log
scale factor model model fitted
Parameter estimates and se's
     name  Estimate   Std. Error      z value Pr(>|z|)
   fdose0  1.917189 0.0001375107 13942.102616        0 ***
 fdose300  1.832115 0.0099444421   184.235060        0 ***
 fdose600  1.723982 0.0187317975    92.035065        0 ***
   fdose0 -1.446450 0.0741870280   -19.497350        0 ***
 fdose300 -1.365485 0.0927950574   -14.715065        0 ***
 fdose600 -1.209275 0.1391476952    -8.690585        0 ***
   log(b) 19.395048          NaN          NaN      NaN

 Signif. codes: '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1

 Log-likelihood: -2653.815 on 7 Df
```

The value of $b = \exp(19.395048)$ is large suggesting the `limiting` distribution of Equations 9 and 10 is worth considering.

```
R> output.fn <- CountsEPPM(number.trials ~ 0 + fdose |  0 + fdose,
+    Luningetal.litters,model = "limiting", ltvalue = 4, utvalue = 11,
+    control=list(maxit = 1000), scale.factor.model = "yes")
```

This results in a return code of `convergence 0 successful` after 543 function calls. Details of the model can be printed out including the parameters of the `limiting` distribution.

```
R> summary.CountsEPPM(output.fn)
R> output.distribution <- distribution.CountsEPPM(output.fn,
+    output.FDparameters = "yes")
R> print(output.distribution$FDparameters)

Model type: mean and variance
Model     : limiting
Link for mean         : log
Link for scale factor : log
scale factor model model fitted
Parameter estimates and se's
     name  Estimate  Std. Error    z value Pr(>|z|)
   fdose0  1.916985 0.007739159 247.699406        0 ***
 fdose300  1.832032 0.009943131 184.251062        0 ***
 fdose600  1.724253 0.018659171  92.407788        0 ***
   fdose0 -1.443848 0.074518751 -19.375637        0 ***
 fdose300 -1.366263 0.092768274 -14.727692        0 ***
 fdose600 -1.210751 0.138809204  -8.722413        0 ***

 Signif. codes: "***" 0.001 "**" 0.01 "*" 0.05 "." 0.1
```

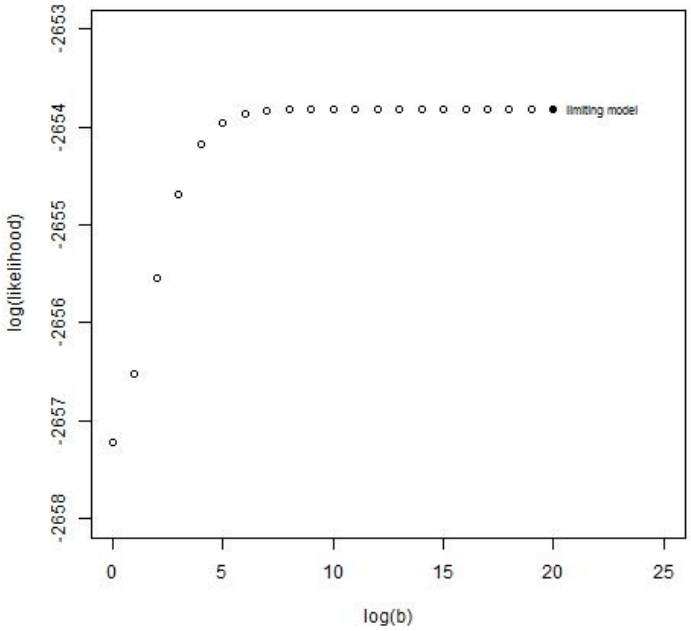

Figure 1: log(likelihood) for fixed values of parameter log(b).

```
 Log-likelihood: -2653.816 on 6 Df

  out.valpha  out.vbeta
1   22.98881 -0.3067164
2   18.91382 -0.3070540
3   13.95323 -0.2872702
```

It can be seen that the `limiting` model fits much the same as the `general` model. To further explore the appropriateness of the limiting model a profile likelihood was constructed for a range of values of parameter b from the version of the `general fixed b` model of Equations 3 and 4 with a plot of the resulting log(likelihoods) against log(b) being produced. In Figure 1 there is a clear trending of the log(likelihood) values toward the value of the limiting model. The code used to produce this plot follows.

```
R> vlogfixed.b <- 0:19
R> vloglikelihood <-  rep(0, 20)
R> for (i in 1:20) {
+    if (i == 1) {
+      output.fn <- CountsEPPM(number.trials ~ 0 + fdose |  0 + fdose,
+        Luningetal.litters, model = "general fixed b", ltvalue = 4,
+        utvalue = 11, fixed.b = exp(vlogfixed.b[i]),
+        scale.factor.model = "yes")
+      output.fn$estses[[1]] <- c("mean dose 0", "mean dose 300",
+        "mean dose 600", "scale factor dose 0", "scale factor 300",
+        "scale factor 600")
```

```
+    } else {
+      output.fn <- CountsEPPM(number.trials ~ 0 + fdose |  0 + fdose,
+        Luningetal.litters, model = "general fixed b", initial = initial,
+        ltvalue = 4, utvalue = 11, fixed.b = exp(vlogfixed.b[i]),
+        scale.factor.model = "yes")
+    }
+    initial <- output.fn$estses[[2]]
+    names(initial) <- output.fn$estses[[1]]
+    vloglikelihood[i] <- output.fn$loglikelihood
+  }
R> plot(vlogfixed.b, vloglikelihood, xlim = c(-10, 10),
+    ylim = c(-2658, -2653), xlab = "log(b)", ylab = "log(likelihood)",
+    main = "Profile likelihood for log(b) Luning litters data")
R> points(6, output.fn$loglikelihood, pch = 16)
R> text(6.1, output.fn$loglikelihood, "limiting model", pos = 4,
+    offset = 0.5, cex = 0.7)
```

4.3. Number of attempts at feeding of herons

These data are originally from Zhu, Eickhoff, and Kaiser (2003) and are in the form of a `list` of frequencies and variables. Faddy and Smith (2005) described an alternative modeling approach to that of Zhu *et al.* (2003) constructing a bivariate EPPM for both count and binary data. Here a univariate EPPM for the numbers of trials (attempts at foraging) of 20 adult and 20 immature green-backed herons is considered. The first model fitted was a negative binomial using the default initial values.

```
R> data("herons.group")
R> output.fn  <- CountsEPPM(number.attempts ~ 0 + group, herons.group,
+    model.type = "mean only", model = "negative binomial")
R> output.fn$estses[[1]] <- c("Adult mean", "Immature mean", "log(b)")
R> summary.CountsEPPM(output.fn)
```

`CountsEPPM` returns a code of `convergence 0 successful` after 110 function calls. Details of the model given by `summary.CountsEPPM` follow.

```
Model type: mean only
Model     : negative binomial
Link for mean         : log
Parameter estimates and se's
          name  Estimate Std. Error  z value     Pr(>|z|)
    Adult mean 0.5623042  0.1549961 3.627861 0.0002857787 ***
 Immature mean 0.4758576  0.1643896 2.894695 0.0037952703 **
        log(b) 0.5082486  0.2679285 1.896956 0.0578337823 .

 Signif. codes: "***" 0.001 "**" 0.01 "*" 0.05 "." 0.1

 Log-likelihood: -120.2042 on 3 Df
```

A Faddy distribution is now fitted.

```
R> output.fn <- CountsEPPM(number.attempts ~ 0 + group, herons.group,
+    model.type = "mean only", model = "Faddy distribution")
R> output.fn$estses[[1]] <- c("Adult mean", "Immature mean", "c", "log(b)")
R> summary.CountsEPPM(output.fn)
```

`CountsEPPM` returns a code of `convergence 0 successful` after 335 function calls. Details of the model given by `summary.CountsEPPM` follow.

```
Model type: mean only
Model     : Faddy distribution
Link for mean         : log
Parameter estimates and se's
          name  Estimate   Std. Error      z value    Pr(>|z|)
    Adult mean 0.5628182 1.549617e-01     3.631982 0.000281253 ***
 Immature mean 0.4765216 1.643571e-01     2.899306 0.003739898 **
             c 0.9999997 6.301249e-05 15869.864382 0.000000000 ***
        log(b) 0.5070286 2.679235e-01     1.892438 0.058432676 .

 Signif. codes: "***" 0.001 "**" 0.01 "*" 0.05 "." 0.1

 Log-likelihood: -120.2042 on 4 Df
```

As can be seen, the Faddy distribution fit is nearly identical to that of the negative binomial. The function `distribution.CountsEPPM` can be used to evaluate how close to 1 the total probabilities are for the Faddy distribution (negative binomial).

```
R> output.distribution <- distribution.CountsEPPM(output.fn,
+    output.probabilities = "yes")
R> print(output.distribution)

$means
  mean.obs variance.obs mean.prob variance.prob totalprob
1     7.95     51.62895  7.069142      30.10360 0.9710490
2     6.65     34.76579  6.309652      26.63537 0.9888142

$probabilities
$probabilities[[1]]
 [1] 0.054207702 0.074451022 0.081919897 0.082680037 0.079683626 0.074619482
 [7] 0.068518504 0.062025215 0.055542261 0.049315532 0.043487554 0.038132360
[13] 0.033278821 0.028926520 0.025056608 0.021639253 0.018638723 0.016016821
[19] 0.013735148 0.011756573 0.010046116 0.008571446 0.007303109 0.006214567
[25] 0.005282124

$probabilities[[2]]
 [1] 0.068980469 0.091648873 0.097552224 0.095244467 0.088797245 0.080440349
```

```
 [7] 0.071453071 0.062570931 0.054202486 0.046555466 0.039713965 0.033687072
[13] 0.028439950 0.023913789 0.020038526 0.016740833 0.013948975 0.011595614
[19] 0.009619269 0.007964909 0.006583996 0.005434214 0.004479007 0.003687025
[25] 0.003031553 0.002489939
```

The closeness of the total probabilities to 1 can be improved by increasing the maximum values for the grouped counts. A print out of the parameters of the Faddy distribution is also produced.

```
R> output.fn$vnmax <- output.fn$vnmax + 60
R> output.distribution <- distribution.CountsEPPM(output.fn,
+    output.FDparameters = "yes")

R> print(output.distribution)

$means
  mean.obs variance.obs mean.prob variance.prob totalprob
1     7.95     51.62895  7.948019      45.99085 0.9999993
2     6.65     34.76579  6.649904      33.28333 1.0000000

$FDparameters
    out.va  out.vb    out.vc
1 1.755613 1.66035 0.9999997
2 1.610463 1.66035 0.9999997
```

4.4. Titanic survivors

To illustrate the inclusion of offsets, data of passenger survival from the 1912 Titanic shipping disaster are used. The data are in data frame form as given in Table 9.37 of Hilbe (2011), i.e., the numbers surviving out of the number of cases (passengers) within different age, sex, and class categories. The individual data for all 1316 passengers is available from package **msme** (Hilbe and Robinson 2014). Hilbe (2011, p. 265–268) analyzes the numbers surviving as count data with an offset of the log of the number of cases for the mean, and fits a negative binomial with variance function $v = m + \alpha m^2$ which equates to the variance function of Equation 4 with $\alpha = \frac{1}{b}$. Both mean and scale factor need to be offset by the log of the number of cases. A series of models were fitted, i.e., a negative binomial with b fixed at the value from Hilbe (2011) of $b = 9.615385$; a negative binomial; a Faddy distribution; a general mean and variance model with only an intercept for the variance; and a general mean and scale factor model with only an intercept for the scale factor. The general mean and scale factor model with only an intercept for the scale factor was found to have the largest log(likelihood). Details of it and its fitting follow.

```
R> data("Titanic.survivors.case")
R> lncases <- log(Titanic.survivors.case$cases)
R> for (i in 1:3) { if (i == 1) {
+    output.fn <- CountsEPPM(survive ~ age + sex + class,
```

```
+      Titanic.survivors.case, offset = list(lncases, lncases),
+      control = list(maxit = 4000), scale.factor.model = "yes")
+    output.fn$estses[[1]] <- c("Intercept mean", "age adult", "sex male",
+      "class 2nd class"," class 3rd class", "Intercept scale factor",
+      "log(b)")
+  } else {
+    initial <- output.fn$estses[[2]]
+    names(initial) <- output.fn$estses[[1]]
+    output.fn <- CountsEPPM(survive ~ age + sex + class,
+      Titanic.survivors.case, offset = list(lncases, lncases),
+      initial = initial, control = list(maxit = 4000),
+      scale.factor.model = "yes")
+  }}
R> summary.CountsEPPM(output.fn)
```

The final run gives a log(likelihood) value of -39.39971 after 1656 function calls; a return code and associated message `convergence 0 successful`; and parameter estimates as follows.

```
Model type: mean and variance
Model     : general
Link for mean         : log
Link for scale factor : log
non zero offsets in linear predictor for mean
non zero offsets in linear predictor for variance
scale factor model model fitted
Parameter estimates and se's
            name    Estimate Std. Error    z value     Pr(>|z|)
  Intercept mean  0.04735906 0.08379422  0.5651830 5.719493e-01
       age adult -0.04338691 0.13331625 -0.3254435 7.448454e-01
        sex male -0.17744857 0.12945297 -1.3707570 1.704507e-01
 class 2nd class -0.03139636 0.11332828 -0.2770391 7.817501e-01
 class 3rd class -0.90083645 0.14494642 -6.2149617 5.133718e-10 ***
 Intercept scale -4.04422601 0.52919046 -7.6422882 2.131628e-14 ***
          log(b) -7.31471067 3.56695791 -2.0506860 4.029753e-02 *

 Signif. codes: "***" 0.001 "**" 0.01 "*" 0.05 "." 0.1

 Log-likelihood: -39.39971 on 7 Df
```

```
R> initial <- output.fn$estses[[2]]
R> names(initial) <- output.fn$estses[[1]]
R> output.fn  <- CountsEPPM(survive ~ age + sex + class,
+    Titanic.survivors.case, offset = list(lncases, lncases),
+    initial = initial, optimization.method = "nlm",
+    control = list(print.level = 1, iterlim = 2000),
+    scale.factor.model = "yes")
```

This run using `nlm` returns a log(likelihood) of −39.39971, the same as earlier, after 16 function calls; a return code with associated message `return code 2 successive iterates within tolerance, current iterate is probably solution`, and final gradients as below.

```
 iteration = 16
Parameter:
[1]  0.04736868 -0.04327433 -0.17740448 -0.03143576 -0.90061482 -4.04403805
[7] -7.31473594
Function Value
[1] 39.39971
Gradient:
[1]  1.228015e-03 -1.392142e-03 -3.324362e-04  3.316573e-03  1.904605e-04
[6]  2.696203e-05  5.533135e-06
```

The parameter estimates together with their standard errors and the parameters of the Faddy distribution can now be printed out.

```
R> summary.CountsEPPM(output.fn)
R> output.distribution <- distribution.CountsEPPM(output.fn,
+    output.FDparameters = "yes")
R> print(output.distribution$FDparameters)

Model type: mean and variance
Model     : general
Link for mean         : log
Link for scale factor : log
non zero offsets in linear predictor for mean
non zero offsets in linear predictor for variance
scale factor model model fitted
Parameter estimates and se's
            name    Estimate Std. Error    z value     Pr(>|z|)
  Intercept mean  0.04736868 0.08379509  0.5652919 5.718753e-01
       age adult -0.04327433 0.13333560 -0.3245519 7.455202e-01
        sex male -0.17740448 0.12945331 -1.3704128 1.705581e-01
 class 2nd class -0.03143576 0.11332093 -0.2774047 7.814694e-01
 class 3rd class -0.90061482 0.14495745 -6.2129598 5.199576e-10 ***
 Intercept scale -4.04403805 0.52977730 -7.6334680 2.287059e-14 ***
          log(b) -7.31473594 3.56921464 -2.0493965 4.042336e-02 *

 Signif. codes: "***" 0.001 "**" 0.01 "*" 0.05 "." 0.1

 Log-likelihood: -39.39971 on 7 Df

         out.va       out.vb      out.vc
1     0.1388207 0.0006656571 -28.0462487
2    45.8756985 0.0006656571   0.3034689
3  1567.5093366 0.0006656571  -5.2064925
```

```
4    40.8129107 0.0006656571   0.3401716
5   388.8886921 0.0006656571  -1.6945827
6    46.9272744 0.0006656571   0.1934622
7   326.8768298 0.0006656571  -2.0936515
8    39.8708414 0.0006656571   0.3330455
9    27.5126488 0.0006656571  -0.4203089
10   24.9972831 0.0006656571   0.3305837
11   20.4592424 0.0006656571  -0.0943559
12   28.1875747 0.0006656571   0.4629002
```

The fit of the general mean and scale factor model is better than that of Hilbe (2011, p. 268), the log(likelihood) values being −39.400 and −43.719 respectively although the former has a one additional parameter.

4.5. Take over bids

These data, originally from Cameron and Johansson (1997), are used as example data in Cameron and Trivedi (2013) as well as in Sáez-Castillo and Conde-Sánchez (2013). The `takeover.bids.case` came from the website associated with Cameron and Trivedi (2013). The dependent variable `NUMBIDS` is the number of bids received by the firm targetted for takeover after the initial bid. Prior to analysis the binary variables were recast as factors and the continuous variables scaled to have zero mean and unit standard deviation. It was found that the scaling of the continuous variables improved the model fitting. The scaling starts by dropping unused variables from the data set and then taking out the count variable.

```
R> data("takeover.bids.case")
R> takeover.bids.case <- transform(takeover.bids.case, TAKEOVER = NULL,
+    CONSTANT = NULL)
R> NUMBIDS <- takeover.bids.case$NUMBIDS
```

The variables that are factors are then extracted and set up as factors.

```
R> LEGLREST <- factor(takeover.bids.case$LEGLREST, levels = c(0,1))
R> REALREST <- factor(takeover.bids.case$REALREST, levels = c(0,1))
R> FINREST  <- factor(takeover.bids.case$FINREST, levels = c(0,1))
R> REGULATN <- factor(takeover.bids.case$REGULATN, levels = c(0,1))
R> WHTKNGHT <- factor(takeover.bids.case$WHTKNGHT, levels = c(0,1))
```

The count variable and factors are then dropped from the data frame and means and standard deviations calculated for rescaling.

```
R> takeover.bids.case <- transform(takeover.bids.case, NUMBIDS = NULL,
+    LEGLREST = NULL, REALREST = NULL, FINREST = NULL, REGULATN = NULL,
+    WHTKNGHT = NULL)
R> cont.var.mean <- lapply(as.list(takeover.bids.case), mean)
R> cont.var.std <- lapply(as.list(takeover.bids.case), sd)
```

The continuous variables are scaled and the count and factors put back into the data frame.

```
R> takeover.bids.case <- as.data.frame(scale(takeover.bids.case,
+    center = TRUE, scale = TRUE))
R> takeover.bids.case <- data.frame(NUMBIDS, takeover.bids.case, LEGLREST,
+    REALREST,  FINREST, REGULATN, WHTKNGHT)
```

Initial exploration of the data starting from the default values indicated that the scale factor model should be focused on and that is what was done. The analyses run were too extensive to report in detail here. What is given below is the code to run one iteration of the model using optimization method `nlm` with initial values the estimates of the best model obtained. This was done in order to review the gradients at those estimates so as to be able to assess whether a maximum likelihood solution has been obtained.

```
R> initial <- c(-0.566854145, 0.267698554, -0.166646016, 0.679694180,
+    0.810712355, -0.314938580, -0.151897896, 0.932242219, -0.858111289,
+    0.245379572, -0.691209393, -0.104513779, 0.214444684,  0.533925230,
+    0.267838551, -0.159394702, -0.005098182, 0.451388246, -0.937629929,
+    0.264256025, -5.331921990)
R> names(initial) <- c("(Intercept) mean", "LEGLREST mean", "REALREST mean",
+    "FINREST mean", "WHTKNGHT mean", "BIDPREM mean", "INSTHOLD mean",
+    "SIZE mean", "SIZESQ mean", "REGULATN mean",
+    "(Intercept) scale factor", "LEGLREST scale factor",
+    "REALREST scale factor", "FINREST scale factor",
+    "WHTKNGHT scale factor", "BIDPREM scale factor",
+    "INSTHOLD scale factor", "SIZE scale factor", "SIZESQ scale factor",
+    "REGULATN scale factor", "log(b)")
R> output.fn  <- CountsEPPM(NUMBIDS ~ LEGLREST + REALREST + FINREST +
+    WHTKNGHT + BIDPREM + INSTHOLD + SIZE + SIZESQ + REGULATN | LEGLREST +
+    REALREST + FINREST + WHTKNGHT + BIDPREM + INSTHOLD + SIZE + SIZESQ +
+    REGULATN, data = takeover.bids.case, optimization.method = "nlm",
+    control = list(print.level = 1,iterlim = 1),initial = initial,
+    scale.factor.model = "yes")
```

The vector of gradients produced is

```
Gradient:
 [1] -2.002233e-04 -3.207514e-04 -4.232879e-04 -3.991507e-05 -5.402657e-04
 [6] -3.924664e-04 -1.993562e-04  2.021037e-04 -1.481257e-04 -3.720278e-04
[11]  2.211448e-04  3.236941e-04  4.393388e-05  1.276247e-04  7.106807e-04
[16]  3.508913e-04 -3.020174e-03  1.911738e-04 -2.496476e-04  2.400501e-04
[21]  4.035934e-05
```

with a log-likelihood of −156.0563 and an AIC of 354.1125. The estimates and their standard errors can now be rescaled.

```
R> output.fn$estses[[2]][c(1)] <- output.fn$estses[[2]][c(1)] -
+    output.fn$estses[[2]][c(6)] * cont.var.mean[[3]] / cont.var.std[[3]] -
+    output.fn$estses[[2]][c(7)] * cont.var.mean[[4]] / cont.var.std[[4]] -
+    output.fn$estses[[2]][c(8)] * cont.var.mean[[5]] / cont.var.std[[5]] -
```

```
+    output.fn$estses[[2]][c(9)] * cont.var.mean[[6]] / cont.var.std[[6]]
R> output.fn$estses[[2]][c(6:9)] <- output.fn$estses[[2]][c(6:9)] /
+    cont.var.std[[3]]
R> output.fn$estses[[3]][c(6:9)] <- output.fn$estses[[3]][c(6:9)] /
+    cont.var.std[[3]]
```

The rescaled parameter estimates together with their standard errors can now be printed out.

```
R>  summary.CountsEPPM(output.fn)
```

```
             name    Estimate  Std. Error       z value     Pr(>|z|)
  (Intercept) mean  1.67014679 0.271556377    6.15027644 7.734802e-10 ***
     LEGLREST mean  0.26769877 0.176811847    1.51403188 1.300178e-01
     REALREST mean -0.16664573 0.248838278   -0.66969491 5.030523e-01
      FINREST mean  0.67969421 0.316405192    2.14817653 3.169973e-02 *
     WHTKNGHT mean  0.81071272 0.210102099    3.85866074 1.140101e-04 ***
      BIDPREM mean -1.66347994 0.516283213   -3.22202988 1.272859e-03 **
     INSTHOLD mean -0.80231228 0.510310550   -1.57220399 1.159032e-01
         SIZE mean  4.92403095 0.007248136  679.35136094 0.000000e+00 ***
       SIZESQ mean -4.53247727 0.006749513 -671.52660812 0.000000e+00 ***
     REGULATN mean  0.24537982 0.184882280    1.32722197 1.844353e-01
 (Intercept) scale  0.44402884 0.348122020    1.27549772 2.021331e-01
    LEGLREST scale -0.10451400 0.164916290   -0.63373969 5.262507e-01
    REALREST scale  0.21444465 0.194301630    1.10366884 2.697368e-01
     FINREST scale  0.53392514 0.363792790    1.46766280 1.421958e-01
    WHTKNGHT scale  0.26783807 0.220500104    1.21468455 2.244864e-01
     BIDPREM scale -0.84191181 0.608403236   -1.38380561 1.664180e-01
    INSTHOLD scale -0.02691738 0.358605790   -0.07506119 9.401660e-01
        SIZE scale  2.38419730 0.720820330    3.30761661 9.409353e-04 ***
      SIZESQ scale -4.95248825 0.490746123  -10.09175216 0.000000e+00 ***
    REGULATN scale  0.26425586 0.228972940    1.15409210 2.484624e-01
           log(b) -5.33192202 3.720409483   -1.43315461 1.518137e-01
```

The Bayesian information criterion (BIC) can also be calculated using `BIC(output.fn)` resulting in a value of 413.6745. Convergence to the maximum likelihood estimates was slow, requring several runs with starting values given by the results of previous runs; this was due to the large number (21) of parameters being estimated, the flatness of the log(likelihood) surface (exemplified by some large standard errors) and a small estimate of the (nuisance) parameter b (negative value of $\log(b)$). The estimate of b being close to 0 and the underdispersion (scale factor < 1) corresponding to $c < 0$ in Equation 2 means that λ_0 can be very large and thus p_0 very small (Equation 1) contributing to a better fitting model. The log-likelihood value of -156.06 for this model with 21 parameters is slightly larger than that of -157.86 from the Sáez-Castillo and Conde-Sánchez (2013) model with 15 parameters. Because of the six extra parameters associated with only a small increase in log(likelihood) the BIC value of 413.67 is larger than those of the models in Sáez-Castillo and Conde-Sánchez (2013): 398.1, 393.5 and 388.3. Sáez-Castillo and Conde-Sánchez (2013) did not report details of fitting a model with the full set of 10 variables in both linear predictors, suggesting that they could

not achieve convergence of the fitting algorithm for this model. The variables Sáez-Castillo and Conde-Sánchez (2013) do not include in their dispersion model as they are not significant are `LEGLREST`, `WHTKNGHT`, `INSTHOLD`, `SIZE`, `SIZESQ`. There is reasonable agreement between the results reported above and those of Sáez-Castillo and Conde-Sánchez (2013) about the important variables in the mean model; in both, `SIZE` and `SIZESQ` have very large t statistics. However, in the dispersion model the results reported above show that `SIZE` and `SIZESQ` also have very large t statistics, whereas in the Sáez-Castillo and Conde-Sánchez (2013) model they are not included due to being not significant.

Examination of the estimated λ_i sequences of Equation 2 shows that the λ_0 values are generally very different from the λ_i values for $i \geq 1$, as a consequence of the very small estimate of the parameter b. A more appropriate model for these data might be one that treats the zero counts differently from the non-zero counts; this makes some sense as a zero count would correspond to the initial bid being accepted by the targetted firm, and different circumstances (in the form of different covariate dependence) might be operating. Such a model is not readily constructed from those considered here, and is therefore beyond the scope of the functions developed. Such modeling will be the subject of future research.

5. Concluding remarks

This article has described use of the R package **CountsEPPM** to fit EPPMs to count data that exhibit under- or overdispersion relative to the Poisson distribution. A variety of covariate dependencies and data structures are covered in examples that provide illustrations of the variety of ways in which the package can be used in the analysis of count data.

As described in Faddy and Smith (2005) and Faddy and Smith (2012), the mean and variance of binary data can be modeled using EPPMs in a similar way to that described here for count data. A package to do this is in development and it is intended to submit it when developed, together with supporting data sets as examples, to CRAN, and to produce an article describing its use. Dependent on how the further research goes, similar functions, etc., may be developed for the model (zero and non-zero counts treated differently) mentioned in the last paragraph of the previous section.

References

Bailer AJ, Oris JT (1997). "Estimating Inhibition Concentrations for Different Response Scales Using Generalized Linear Models." *Environmental Toxicology and Chemistry*, **16**(7), 1554–1559. doi:10.1002/etc.5620160732.

Bates D, Maechler M (2016). ***Matrix**: Sparse and Dense Matrix Classes and Methods*. R package version 1.2-4, URL https://CRAN.R-project/package=Matrix.

Cameron AC, Johansson P (1997). "Count Data Regression Using Series Expansions: With Applications." *Journal of Applied Econometrics*, **12**, 203–223. doi:10.1002/(sici)1099-1255(199705)12:3<203::aid-jae446>3.0.co;2-2.

Cameron AC, Trivedi PK (2013). *Regression Analysis of Count Data*. 2nd edition. Cambridge University Press.

Cribari-Neto F, Zeileis A (2010). "Beta Regression in R." *Journal of Statistical Software*, **34**(2), 1–24. doi:10.18637/jss.v034.i02.

Faddy MJ (1997). "Extended Poisson Process Modelling and Analysis of Count Data." *Biometrical Journal*, **39**(4), 431–440. doi:10.1002/bimj.4710390405.

Faddy MJ, Smith DM (2005). "Modelling the Dependence between the Number of Trials and the Success Probability in Binary Trials." *Biometrics*, **61**(4), 1112–1114. doi:10.1111/j.1541-0420.2005.00466.x.

Faddy MJ, Smith DM (2008). "Extended Poisson Process Modelling of Dilution Series Data." *Applied Statistics*, **57**(4), 461–471. doi:10.1111/j.1467-9876.2008.00622.x.

Faddy MJ, Smith DM (2011). "Analysis of Count Data with Covariate Dependence in Both Mean and Variance." *Journal of Applied Statistics*, **38**(12), 2683–2694. doi:10.1080/02664763.2011.567250.

Faddy MJ, Smith DM (2012). "Extended Poisson Process Modeling and Analysis of Grouped Binary Data." *Biometrical Journal*, **54**(3), 426–435. doi:10.1002/bimj.201100214.

Gilbert P, Varadhan R (2015). ***numDeriv****: Accurate Numerical Derivatives*. R package version 2014.2-1, URL https://CRAN.R-project/package=numDeriv.

Goulet V, Dutang C, Maechler M, Firth D, Shapira M, Stadelmann M (2015). ***expm****: Matrix Exponential*. R package version 0.999.1, URL https://CRAN.R-project/package=expm.

Grün B, Kosmidis I, Zeileis A (2012). "Extended Beta Regression in R: Shaken, Stirred, Mixed, and Partitioned." *Journal of Statistical Software*, **48**(11), 1–25. doi:10.18637/jss.v048.i11.

Grunwald GK, Bruce SL, Jiang L, Strand M, Rabinovitch N (2011). "A Statistical Model for Under- or Overdispersed Clustered and Longitudinal Count Data." *Biometrical Journal*, **53**(4), 578–594. doi:10.1002/bimj.201000076.

Hilbe J, Robinson A (2014). ***msme****: Functions and Datasets for 'Methods of Statistical Model Estimation'*. R package version 0.5.1, URL https://CRAN.R-project.org/package=msme.

Hilbe JM (2011). *Negative Binomial Regression*. 2nd edition. Cambridge University Press.

Hilbe JM (2014). *Modeling Count Data*. Cambridge University Press.

Lüning KG, Sheridan W, Ytterborn KH, Gullberg U (1966). "The Relationship Between the Number of Implantations and the Rate of Intra-Uterine Death in Mice." *Mutation Research*, **3**, 444–451. doi:10.1016/0027-5107(66)90054-6.

Nash JC (2014). *Nonlinear Parameter Optimization Using R Tools*. John Wiley & Sons. doi:10.1002/9781118884003.

Nelder JA, Mead R (1967). "A Simplex Method for Function Minimisation." *The Computer Journal*, **7**(4), 308–313. doi:10.1093/comjnl/7.4.308.

R Core Team (2016). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.

Ridout MS, Besbeas P (2004). "An Empirical Model for Underdispersed Count Data." *Statistical Modelling*, **4**(1), 77–89. doi:10.1191/1471082x04st064oa.

Sáez-Castillo AJ, Conde-Sánchez A (2013). "A Hyper-Poisson Regression Model for Overdispersed and Underdispersed Count Data." *Computational Statistics & Data Analysis*, **61**, 148–157. doi:10.1016/j.csda.2012.12.009.

Smith DM, Faddy MJ (2016). ***CountsEPPM**: Fitting of EPPM Models to Count Data*. R package version 2.1, URL https://CRAN.R-project/package=CountsEPPM.

Toscas P, Faddy MJ (2003). "Likelihood-Based Analysis of Longitudinal Count Data Using a Generalized Poisson Model." *Statistical Modelling*, **3**(2), 99–108. doi:10.1191/1471082x03st050oa.

Zeileis A, Croissant Y (2010). "Extended Model Formulas in R: Multiple Parts and Multiple Responses." *Journal of Statistical Software*, **34**(1), 1–13. doi:10.18637/jss.v034.i01.

Zhu J, Eickhoff JC, Kaiser MS (2003). "Modelling the Dependence between Number of Trials and Success Probability in Beta-Binomial-Poisson Mixture Distributions." *Biometrics*, **59**(4), 955–961. doi:10.1111/j.0006-341x.2003.00110.x.

Affiliation:

David M. Smith
Truven Health Analytics
7700 Old Georgetown Road, Suite 650
Bethesda, MD 20814, United States of America
E-mail: david.m.smith@truvenhealth.com

Malcolm J. Faddy
School of Mathematical Sciences
Queensland University of Technology
G.P.O. Box 2434
Brisbane Qld 4001, Australia
E-mail: m.faddy@qut.edu.au

Journal of Statistical Software http://www.jstatsoft.org/
published by the Foundation for Open Access Statistics http://www.foastat.org/
March 2016, Volume 69, Issue 6 *Submitted:* 2013-02-20
doi:10.18637/jss.v069.i06 *Accepted:* 2015-02-11