Mean and Scale-Factor Modeling of Under- and Overdispersed Count Data

David M. Smith, Malcolm J. Faddy

January 08, 2024

Abstract

This vignette describes an updated version of 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 scale-factor 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

This vignette is for version 3.1 of **CountsEPPM**. The updates from version 3.0 are minor. Version 3.0 is a revised and added-to version of Smith and Faddy (2016) which related to version 2.1. The important differences between previous versions and 3.1, 3.0 are a focus on mean and scale-factor models with variance models dropped, the addition of generic (S3) methods, using only `optim` for optimization i.e., no use of `nlm`, and offsets included in the formulae. Readers should refer to Smith and Faddy (2016) for details of the actual modelling which involves Markov birth processes. This vignette was constructed as a static pdf file using (R.rsp) Bengtsson (2022).

The models using extended Poisson process models (EPPMs) were 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; 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 et al., 2012). Both Hilbe (2011) and Hilbe (2014) have commented about a software package for EPPMs being developed in the R system (R Core Team, 2023); the package **CountsEPPM** Smith and Faddy (2018) whose use is described in this vignette is that software.

2 Description of the functions

The main function of the package, also named `CountsEPPM`, is focused on models with two covariate dependences linked to the mean and scale-factor. The input into the function is a formula involving a single response variable and one or two formulae related to the mean and scale-factor 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 the response variable (mean, scale-factor) and linear predictor of covariates is log; the log of parameter b is also used but the parameter c of Equation 2 of Smith and Faddy (2016) is untransformed. The full three parameter version of Equation 2 of Smith and Faddy (2016) has been labeled the Faddy distribution as in Grunwald et al. (2011). Because of possible issues with the parameter b, variants of the models where b is fixed have been included in the lists of models. This enables profile log-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 the R function `optim` (options used being the simplex method of Nelder and Mead (1967) or `BFGS` using numerical derivatives). A facility to change options for `optim` through use of the argument `control` is included. The elements of this argument are the options for `optim` as described in R Core Team (2023). The default values set within `CountsEPPM` are `fnscale = -1`, `trace = 0`, `maxit = 1000` for `optim`. Although for most data sets the two options of `optim` 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 simplex method, is robust to discontinuities in the log-likelihood surface. However, it is slow to converge. In contrast the function `BFGS` 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 `BFGS` makes use of derivatives in the actual model fitting, but both options calculate a hessian matrix from the derivatives to produce standard errors for the parameter estimates. The calculation of the numerical derivatives and hessians use the functions `grad` and `hessian` from the package **numDeriv** (Gilbert and Varadhan, 2019). The derivatives are more accurately calculated using **numDeriv** (Gilbert and Varadhan, 2019) than using alternative central difference approximations, resulting in better model fitting and better conditioned hessians. However, as stated on Nash (2014, p. 131), a longer time is taken. 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 conclusion of parameter estimation to evaluate whether maximum likelihood estimates have been attained. The code for the main analysis function is

```
CountsEPPM(formula, data, subset = NULL, na.action = NULL, weights = NULL,
  model.type = "mean and scale-factor", model.name = "general", link = "log",
  initial = NULL, ltvalue = NA, utvalue = NA, method = "Nelder-Mead",
  control = NULL, fixed.b=NA)
```

with details of the arguments given in Table 1 together with defaults if any. As in earlier versions, `data` can be either a `list` or a `data.frame`. The response variable in `formula` is a vector if a `data.frame` is input, or a `list` if a `list` is input. The response variables `mean.obs` and `variance.obs` are constructed within `CountsEPPM` prior to being used to fit models. Package **Formula** Zeileis and Croissant (2010) is used to extract model information from the `formula` input to `CountsEPPM`. To avoid repeated extractions within subordinate functions, extraction of model information used in the model fitting, such as `covariates.matrix.mean`, is only done once within `CountsEPPM`. In version 3.0 a set of S3 generic extractor functions for objects of class `"CountsEPPM"` was added. The set is similar to that of Table 1 of `betareg` (Cribari-Neto and Zeileis, 2010). `CountsEPPM` returns an object of class `"CountsEPPM"` summarizing the model fit, the components of which are given in Table 2.

Argument	Description	Default
`formula`	a single response variable & paired formulae Zeileis and Croissant (2010)	
`data`	`data.frame` or `list`	
`subset`	subsetting commands	NULL
`na.action`	action taken for NAs in data	NULL
`weights`	vector if `data` is a `data.frame` a `list` if `data` is a `list` attributes `normalization`, `norm.to.n`	vector of ones list of lists of ones both NULL

`model.type`	`"mean only"`
	`"mean and scale-factor"`
		`"mean and scale-factor"`

if `model.type = "mean only"` (only a in Equation 2 of Smith and Faddy (2016) modeled)		

`model.name`	`"Poisson"`
	`"negative binomial"`
	`"negative binomial fixed b"`
	`"Faddy distribution"` Equation 2 of Smith and Faddy (2016)
	`"Faddy distribution fixed b"`	

if `model.type = "mean and scale-factor"` (both modeled)		
`model.name`	`"general"` as Equations 3 and 4 `"general fixed b"` `"limiting"` as Equations 9 and 10 of Smith and Faddy (2016)	`"general"`
`link`	the glm link function for mean count only log allowed	`"log"`
`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`

`method`	`"Nelder-Mead"`
	`"BFGS"` attribute `"grad.method"` which is `"simple"` or `"Richardson"`
		`"Nelder-Mead"`
		attribute `"simple"`

`control`	list of control parameters `optim`	see text for more details
`fixed.b`	value b is fixed at	`NA`

Table 1: Arguments of `CountsEPPM`.

Component	Description
`data.type`	`"data.frame"` or `"list"`
`list.data`	data as a `"list"` of frequency distributions
`call`	the call to **CountsEPPM**
`formula`	the `formula` input
`model.type`	`"mean only"` or `"mean and scale-factor"`
`model.name`	`"Poisson"`, `"negative binomial"`, `"negative binomial fixed b"`, `"Faddy distribution"` (Equation 2), `"Faddy distribution fixed b"`, `"general"` (Equations 3 & 4), `"general fixed b"`, `"limiting"` (Equations 9 & 10). Equation numbers of Smith and Faddy (2016)
`link`	the glm link function for mean count
`covariates.matrix.mean`	matrix of covariates for the mean
`covariates.matrix.scalef`	matrix of covariates for the scale-factor
`offset.mean`	offset vector for the mean
`offset.scalef`	offset vector for the scale-factor
`coefficients`	the estimated coefficients
`loglik`	the final log-likelihood value
`vcov`	the estimated variance/covariance matrix
`n` needed for `lmtest`	the number of observations
`nobs` needed for `stats`	the number of observations
`df.null`	null model degrees of freedom
`df.residual`	residual degrees of freedom
`ltvalue`	lower truncation value (excluded)
`utvalue`	upper truncation value (excluded)
`fixed.b`	value b is fixed at
`vnmax`	a vector of maximum counts in each of the grouped data vectors
`weights`	a vector or list of weights
`converged`	whether converged
`iterations`	number of iterations
`method`	`"Nelder-Mead"` or `"BFGS"`
`start`	initial estimates input
`optim`	final estimates of coefficients
`control`	control parameters of `optim`
`fitted.values`	fitted values of mean count
`y`	observed values of mean count
`terms`	model terms

Table 2: Components of object returned by `CountsEPPM`.

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 scale-factor is also being modeled the initial estimates of the parameters of the scale-factor linear predictor are set to 1.0 recognising that for the Poisson distribution the variance equals the mean. The initial value of $\log(b)$ of Equation 2 of Smith and Faddy (2016) is set to zero. The matrix exponential function used for calculating the probabilities of Equation 1 of Smith and Faddy (2016) is that of the package **expm** of Maechler et al. (2023) which depends on the package **Matrix** of Bates et al. (2023).

Function	Description
`print()`	a simple printed display
`summary()`	standard regression output (coefficient estimates, standard errors, partial Wald tests); returns an object of class `summary.BinaryEPPM` containing the relevant summary statistics (which has a `print` method)
`coef()`	extract coefficients of model (full, mean, or scale-factor components), a single vector of all coefficients by default
`vcov()`	associated covariance matrix (with matching names)
`predict()`	predictions (response, linear predictor mean, linear predictor scale-factor, scale-factor, mean, variance, distribution probabilities, distribution parameters) for existing and new data
`fitted()`	fitted means for observed data
`residuals()`	extract residuals (deviance, Pearson, response, standardized deviance, standardized Pearson residuals), defaulting to standardized Pearson residuals
`terms()`	extract terms of model components
`model.matrix()`	extract model matrix of model components
`model.frame()`	extract full original model frame
`logLik()`	extract fitted log-likelihood
`plot()`	diagnostic plots of residuals, predictions, leverages, etc.
`hatvalues()`	hat values (diagonal of hat matrix)
`cooks.distance()`	Cook's distance
`gleverage()`	generalized leverage
`waldtest()`	Wald tests of model parameters
`coeftest()`	partial Wald tests of coefficients
`lrtest()`	likelihood ratio tests of model parameters
`AIC()`	compute information criteria (AIC, BIC, ...)

Table 3: Generic Functions for Use with Objects of Class `CountsEPPM`.

Table 3 gives details of a set of S3 generic extractor functions for objects of class `"CountsEPPM"`. The set is similar to that of Table 1 of Cribari-Neto and Zeileis (2010) related to package **betareg**, except there are no functions `estfun`, `bread` or `linear.hypothesis`. Also, `gleverage` and `cooks.distance` are variants of the functions `glm.diag` and `glm.diag.plots` from package **boot** Canty and Ripley (2022) rather than **betareg**. The first four blocks refer to functions specific to **CountsEPPM**. The last block contains generic functions, the default versions of which work because of the information supplied by the functions of the first four blocks. Package **lmtest** Zeileis and Hothorn (2002) needs to be loaded to use `coeftest` and `lrtest`. Function `AIC` comes from **stats** which is a default package loaded when R is started. In Table 2 both `n` and `nobs` are included, so that functions from both packages **lmtest** and **stats** can use the object returned.

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., `$vnmax` is a vector of the maximum observed counts. If probabilities for counts greater than these maximums are wanted, the values in `output.fn$vnmax` can be increased in value and `predict` with `type="distribution"` run to obtain these probabilities.

3 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 the log-likelihood surface possible, particularly with respect to the parameter b. It is recommended that analyses be run more than once using different initial estimates and optimization methods.

3.1 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 scale-factor"`, and `model` of `"general"`, are used. The code given below is for a run using the `method="simplex"` option of `optim` followed by one using `BFGS` with `attribute="Richardson"` with this last run giving 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.

```
> output.fn <- CountsEPPM(number.young ~ 1 + vdose + vdose2 |
+     1 + vdose + vdose2, data = ceriodaphnia.group, control = list(maxit = 4000,
+     reltol = 1e-11))
> names(output.fn$optim$par) <- c("mean Intercept", "mean dose",
+     "mean dose^2", "scale-factor Intercept", "scale-factor dose",
+     "scale-factor dose^2", "log(b)")
> method <- "BFGS"
> attr(method, which = "grad.method") <- "Richardson"
> output.fn <- update(output.fn, initial = output.fn$optim$par,
+     method = method)
> summary(output.fn)

 Dependent variable is a list of frequency distributions for counts
Call:
CountsEPPM(formula = number.young ~ 1 + vdose + vdose2 | 1 + vdose +
    vdose2, data = ceriodaphnia.group, initial = output.fn$optim$par,
    method = method, control = list(maxit = 4000, reltol = 1e-11))
Model type        : mean and scale-factor
Model name        : general
Link scale-factor : log
 Coefficients (model for mean with log link)
t test of coefficients:
                Estimate Std. Error t value  Pr(>|t|)
mean Intercept  3.1406554  0.0869054 36.1388 < 2.2e-16 ***
mean dose       0.1733831  0.0306760  5.6521 1.170e-06 ***
mean dose^2    -0.0196109  0.0025417 -7.7156 1.203e-09 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
 Coefficients (model for scale-factor with log link)
t test of coefficients:
                       Estimate Std. Error t value Pr(>|t|)
scale-factor Intercept  1.297690   0.446589  2.9058 0.005768 **
scale-factor dose      -0.665740   0.219718 -3.0300 0.004129 **
scale-factor dose^2     0.047334   0.015831  2.9899 0.004603 **
log(b)                 -0.144725   2.825796 -0.0512 0.959391
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
 Type of estimator: ML (maximum likelihood)
 Log-likelihood: -152.1356 on 7 Df
 Number of iterations: 54 of optim method BFGS gradient method Richardson
 final gradients of parameters
[1]  0.0004941553 -0.0037358680 -0.0146512799  0.0002093457 -0.0003585277
[6]  0.0037383454  0.0008115459
 return code 0 successful
```

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. Further details of the model can be printed out such as the parameters of the Faddy distribution as well as the predicted values of the means, variances and scale-factors.

```
> print(predict(output.fn, type = "distribution.parameters"))
      out.va    out.vb     out.vc
1   7.661050 0.8652604  0.5179848
2  18.489075 0.8652604  0.1830096
3  56.471380 0.8652604 -0.2029656
4 414.300384 0.8652604 -0.9160960
5   6.799103 0.8652604  0.2157105
> predictions <- data.frame(mean = predict(output.fn, type = "mean"),
+     variance = predict(output.fn, type = "variance"), scale.factor = predict(output.fn,
+         type = "scale.factor"))
> print(predictions)
       mean variance scale.factor
1 23.119013 84.63475    3.6608289
2 28.895566 41.96015    1.4521312
3 32.817583 23.81847    0.7257839
4 31.761267 11.51949    0.3626899
5  9.428487 13.67522    1.4504148
```

To illustrate use of the `newdata` argument of `predict` a new `data.frame` of the second and third rows is constructed and used with `predict`.

```
> newdata <- data.frame(intercept = rep(1, 2), vdose = ceriodaphnia.group$vdose[2:3],
+     vdose2 = ceriodaphnia.group$vdose2[2:3], vnmax = c(35,
+         44))
> predictions <- data.frame(mean = predict(output.fn, newdata = newdata,
+     type = "mean"), variance = predict(output.fn, newdata = newdata,
+     type = "variance"), scale.factor = predict(output.fn,
+     newdata = newdata, type = "scale.factor"))
> print(predictions)
      mean variance scale.factor
1 28.89557 41.96015    1.4521312
2 32.81758 23.81847    0.7257839
```

3.2 Lüning et al. data

These data are from Lüning et al. (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 of Smith and Faddy (2016).

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

 Dependent variable is a list of frequency distributions for counts
 distribution truncated below at  4
 distribution truncated above at  11
Call:
CountsEPPM(formula = number.trials ~ 0 + fdose | 0 + fdose, data = Luningetal.litters,
    ltvalue = 4, utvalue = 11, control = list(maxit = 2000))
Model type        : mean and scale-factor
Model name        : general
Link scale-factor : log
 Coefficients (model for mean with log link)
t test of coefficients:
          Estimate Std. Error t value  Pr(>|t|)
fdose0   1.9171890  0.0022050 869.471 < 2.2e-16 ***
fdose300 1.8321149  0.0099383 184.350 < 2.2e-16 ***
fdose600 1.7239822  0.0186856  92.262 < 2.2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
 Coefficients (model for scale-factor with log link)
t test of coefficients:
          Estimate Std. Error  t value  Pr(>|t|)
fdose0   -1.446450   0.074170 -19.5018 < 2.2e-16 ***
fdose300 -1.365485   0.092402 -14.7777 < 2.2e-16 ***
fdose600 -1.209275   0.138516  -8.7302 < 2.2e-16 ***
```

```
log(b)   19.395048        NaN      NaN       NaN
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
 Type of estimator: ML (maximum likelihood)
 Log-likelihood: -2653.815 on 7 Df
 Number of iterations: 1068 of optim method Nelder-Mead
 return code 0 successful
```

A warning message `In sqrt(diag(varcov)) :  NaNs produced` is produced which appears in the (R.rsp) Bengtsson (2022) log file. This message and the value of $b = \exp(19.395048)$ being large suggests that the fitted model corresponds to a negative exponential sequence in the underlying birth process (Equations 9 and 10 of Smith and Faddy (2016)).

```
> output.fn <- update(output.fn, model.name = "limiting")
> summary(output.fn)

 Dependent variable is a list of frequency distributions for counts
 distribution truncated below at  4
 distribution truncated above at  11
Call:
CountsEPPM(formula = number.trials ~ 0 + fdose | 0 + fdose, data = Luningetal.litters,
    model.name = "limiting", ltvalue = 4, utvalue = 11, control = list(maxit = 2000))
Model type        : mean and scale-factor
Model name        : limiting
Link scale-factor : log
 Coefficients (model for mean with log link)
t test of coefficients:
          Estimate Std. Error t value  Pr(>|t|)
fdose0   1.9169670  0.0077381 247.730 < 2.2e-16 ***
fdose300 1.8320442  0.0099421 184.271 < 2.2e-16 ***
fdose600 1.7242914  0.0186506  92.453 < 2.2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
 Coefficients (model for scale-factor with log link)
t test of coefficients:
          Estimate Std. Error  t value  Pr(>|t|)
fdose0   -1.444080   0.074503 -19.3828 < 2.2e-16 ***
fdose300 -1.366303   0.092760 -14.7294 < 2.2e-16 ***
fdose600 -1.210711   0.138765  -8.7249 < 2.2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
 Type of estimator: ML (maximum likelihood)
 Log-likelihood: -2653.816 on 6 Df
 Number of iterations: 533 of optim method Nelder-Mead
 return code 0 successful
```

The parameters of the limiting model can be printed out

```
> lm.loglik <- output.fn$loglik
> print(lm.loglik)
```

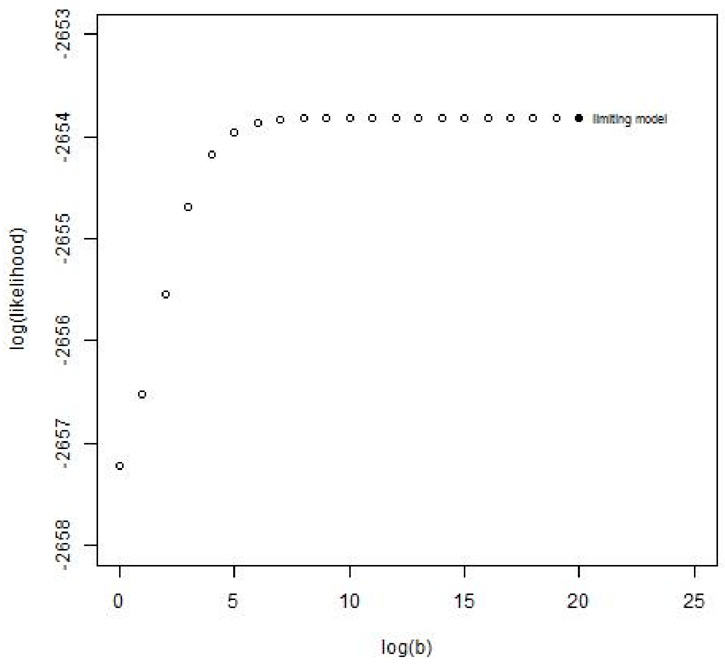

Figure 1: log-likelihood for fixed values of parameter log(b).

```
[1] -2653.816
> predict(output.fn, type = "distribution.parameters")
  out.valpha  out.vbeta
1   22.99626 -0.3067979
2   18.91508 -0.3070639
3   13.95311 -0.2872457
```

showing much the same results as the previous fit of 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 of Smith and Faddy (2016) 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 Figure 1 follows.

```
> vfixed.b <- c(0:19)
> vloglikelihood <- rep(0, 20)
> vloglikelihood <- sapply(1:20, function(i) {
+     if (i == 1) {
+         output.fn <- CountsEPPM(number.trials ~ 0 + fdose |
+             0 + fdose, Luningetal.litters, model.name = "general fixed b",
+             ltvalue = 4, utvalue = 11, fixed.b = exp(vfixed.b[i]))
+     }     else {
+         output.fn <- CountsEPPM(number.trials ~ 0 + fdose |
+             0 + fdose, Luningetal.litters, model.name = "general fixed b",
+             ltvalue = 4, utvalue = 11, initial = output.fn$optim$par,
+             fixed.b = exp(vfixed.b[i]))
+     }
+     vloglikelihood[i] <- output.fn$loglik
+ })
> plot(vfixed.b, vloglikelihood, xlim = c(0, 25), ylim = c(-2658,
+     -2653), main = "Profile likelihood for log(b) Luning litters data",
+     xlab = "log(b)", ylab = "log(likelihood)")
> points(20, lm.loglik, pch = 16)
> text(20.1, lm.loglik, "limiting model", pos = 4, offset = 0.5,
+     cex = 0.7)
```

3.3 Number of attempts at feeding of herons

These data are originally from Zhu et al. (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 (number of attempts) and grouped (number of successful attempts) 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.

```
> output.fn.one <- CountsEPPM(number.attempts ~ 0 + group,
+     herons.group, model.type = "mean only", model.name = "negative binomial")
> names(output.fn.one$optim$par) <- c("Adult mean", "Immature mean",
+     "log(b)")
> print(summary(output.fn.one))

 Dependent variable is a list of frequency distributions for counts
Call:
CountsEPPM(formula = number.attempts ~ 0 + group, data = herons.group,
    model.type = "mean only", model.name = "negative binomial")
Model type        : mean only
Model name        : negative binomial
 Coefficients (model for mean with log link)
t test of coefficients:
              Estimate Std. Error t value  Pr(>|t|)
group Adult     0.56230    0.15500  3.6279 0.0008573 ***
group Immature  0.47586    0.16439  2.8947 0.0063315 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
 Coefficients (model for scale-factor with log link)
t test of coefficients:
       Estimate Std. Error t value Pr(>|t|)
log(b)  0.50825    0.26793   1.897  0.06566 .
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
 Type of estimator: ML (maximum likelihood)
 Log-likelihood: -120.2042 on 3 Df
 Number of iterations: 108 of optim method Nelder-Mead
 return code 0 successful
```

The second model fitted was a more general Faddy distribution again using the default initial values.

```
> output.fn.two <- update(output.fn.one, model.name = "Faddy distribution")
> names(output.fn.two$optim$par) <- c("Adult mean", "Immature mean",
+     "c", "log(b)")
> print(summary(output.fn.two))

 Dependent variable is a list of frequency distributions for counts
Call:
CountsEPPM(formula = number.attempts ~ 0 + group, data = herons.group,
    model.type = "mean only", model.name = "Faddy distribution")
Model type        : mean only
Model name        : Faddy distribution
 Coefficients (model for mean with log link)
t test of coefficients:
               Estimate Std. Error t value  Pr(>|t|)
group Adult     0.56282    0.15496  3.6320 0.0008688 ***
group Immature  0.47652    0.16436  2.8993 0.0063351 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
 Coefficients (model for scale-factor with log link)
t test of coefficients:
         Estimate Std. Error    t value Pr(>|t|)
c      1.0000e+00 1.5724e-05 63596.0001   <2e-16 ***
log(b) 5.0703e-01 2.6792e-01     1.8924   0.0665 .
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
 Type of estimator: ML (maximum likelihood)
 Log-likelihood: -120.2042 on 4 Df
 Number of iterations: 331 of optim method Nelder-Mead
 return code 0 successful
```

The estimate of the parameter *c* here is very close to 1.000, the upper limit of the range of permitted values of this parameter which corresponds to a negative binomial model. A Wald test can be performed to compare the coefficients in two models, which as the negative binomial is embedded in the Faddy distribution model, is equivalent to the t test of coefficient `c`. This equivalence is mentioned in the documentation for `waldtest` in the package **lmtest**.

```
> waldtest(output.fn.two, output.fn.one)
Wald test
Model 1: number.attempts ~ 0 + group
Model 2: number.attempts ~ 0 + group
  Res.Df Df Chisq Pr(>Chisq)
1     36
2     37 -1 8.406    0.00374 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

A likelihood ratio test can also be performed to compare the model fits.

```
> lrtest(output.fn.one, output.fn.two)
Likelihood ratio test
```

```
Model 1: number.attempts ~ 0 + group
Model 2: number.attempts ~ 0 + group
  #Df LogLik Df Chisq Pr(>Chisq)
1   3 -120.2
2   4 -120.2  1     0      0.997
```

Print outs of the parameters of the Faddy distribution and the associated distribution for the first group can be produced.

```
> predict(output.fn.two, type = "distribution.parameters")
    out.va  out.vb    out.vc
1 1.755613 1.66035 0.9999997
2 1.610463 1.66035 0.9999997
> predict(output.fn.two, type = "distribution")[1]
[[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
```

By increasing the maximum values for the grouped counts using the following code, the probabilities for the next ten counts in the sequence for the first group can be obtained.

```
> wks <- output.fn.two$vnmax[1] + 2
> wke <- output.fn.two$vnmax[1] + 11
> output.fn.two$vnmax[1] <- output.fn.two$vnmax[1] + 10
> predict(output.fn.two, type = "distribution")[[1]][wks:wke]
 [1] 0.0044847776 0.0038040231 0.0032236418 0.0027294804 0.0023092342
 [6] 0.0019522416 0.0016492910 0.0013924445 0.0011748767 0.0009907317
```

A weighted analysis using the reciprocal of the predicted variances can be performed.

```
> herons.group$weights <- herons.group$number.attempts
> weights <- 1/predict(output.fn.one, type = "variance")
> herons.group$weights <- lapply(1:length(herons.group$weights),
+     function(i) {
+         herons.group$weights[[i]] <- rep(weights[i], length(herons.group$weights[[i]]))
+     })
> attr(herons.group$weights, which = "normalize") <- TRUE
> output.fn <- CountsEPPM(number.attempts ~ 0 + group, herons.group,
+     model.type = "mean only", model.name = "Poisson", weights = herons.group$weights)
> names(output.fn$optim$par) <- c("Adult mean", "Immature mean")
> summary(output.fn)

 Dependent variable is a list of frequency distributions for counts
Call:
CountsEPPM(formula = number.attempts ~ 0 + group, data = herons.group,
    weights = herons.group$weights, model.type = "mean only", model.name = "Poisson")
Model type        : mean only
Model name        : Poisson
 Coefficients (model for mean with log link)
t test of coefficients:
               Estimate Std. Error t value  Pr(>|t|)
group Adult    2.073172   0.086557  23.951 < 2.2e-16 ***
group Immature 1.894617   0.080490  23.538 < 2.2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
 Maximum weighted likelihood regression.
 List of weights used.
 Type of estimator: ML (maximum likelihood)
 Log-likelihood: -168.3474 on 2 Df
 Number of iterations: 45 of optim method Nelder-Mead
 return code 0 successful
```

The same data as `herons.group`, but in `data.frame` form, has been included in the package as `herons.case`. Running the same code with `herons.group` replaced by `herons.case` will produce essentially the same outputs.

3.4 Titanic survivors

To illustrate the inclusion of offsets, data of passenger survival from the 1912 sinking of the Titanic 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 (2018). 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 of Smith and Faddy (2016) 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 was fitted: a negative binomial with the parameter b fixed at the value from Hilbe (2011) of $b = 9.615385$; a negative binomial; a Faddy distribution; 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.

```
> lncases <- log(Titanic.survivors.case$cases)
> output.fn <- CountsEPPM(survive ~ age + sex + class + offset(lncases) |
+     1 + offset(lncases), Titanic.survivors.case, control = list(maxit = 2000))
> names(output.fn$optim$par) <- c("Intercept mean", "age adult",
+     "sex male", "class 2nd class", "class 3rd class", "Intercept scale",
+     "log(b)")
> output.fn <- update(output.fn, initial = output.fn$optim$par,
+     method = "BFGS")
> summary(output.fn)

 Dependent variable a vector of counts.
Call:
CountsEPPM(formula = survive ~ age + sex + class + offset(lncases) |
    1 + offset(lncases), data = Titanic.survivors.case, initial = output.fn$optim$par,
    method = "BFGS", control = list(maxit = 2000))
Model type        : mean and scale-factor
Model name        : general
```

```
Link scale-factor : log
non zero offsets in linear predictors
 Coefficients (model for mean with log link)
t test of coefficients:
                 Estimate Std. Error t value Pr(>|t|)
Intercept mean   0.047473   0.083371  0.5694 0.593694
age adult       -0.043302   0.132670 -0.3264 0.757348
sex male        -0.177380   0.129397 -1.3708 0.228762
class 2nd class -0.031658   0.112977 -0.2802 0.790538
class 3rd class -0.900906   0.144728 -6.2248 0.001565 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
 Coefficients (model for scale-factor with log link)
t test of coefficients:
                Estimate Std. Error t value Pr(>|t|)
Intercept scale -4.04620    0.54015 -7.4909  0.00067 ***
log(b)          -7.32906    3.64542 -2.0105  0.10058
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
 Type of estimator: ML (maximum likelihood)
 Log-likelihood: -39.39972 on 7 Df
 Number of iterations: 60 of optim method BFGS gradient method simple
 final gradients of parameters
[1] -0.0205864745 -0.0007941051  0.0041589485  0.0022987724  0.0068945663
[6]  0.0018445196  0.0006846674
 return code 0 successful
```

The parameters of the Faddy distribution can now be printed out.

```
> predict(output.fn, type = "distribution.parameters")
         out.va       out.vb       out.vc
1     0.1393248 0.0006561875 -28.10787857
2    45.9518038 0.0006561875   0.30301645
3  1594.5275880 0.0006561875  -5.21885211
4    40.8714465 0.0006561875   0.33977979
5   392.8734253 0.0006561875  -1.69933911
6    47.0245997 0.0006561875   0.19279231
7   330.3158576 0.0006561875  -2.09927205
8    39.9228614 0.0006561875   0.33264198
9    27.6084984 0.0006561875  -0.42230350
10   25.0221806 0.0006561875   0.33016644
11   20.5064676 0.0006561875  -0.09564431
12   28.2096641 0.0006561875   0.46262788
```

The fit of the general mean and scale-factor model is better than that of Hilbe (2011)(page 268), the log-likelihood values being -39.400 and -43.719 respectively; although the former has one extra parameter it would be preferred according to AIC.

3.5 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 targeted for takeover after the initial bid. As both variables CASE, CONSTANT are equal to 1 throughout they have not been included in package data set. In Smith and Faddy (2016), related to version 2.1 of **CountsEPPM**, the continuous variables were scaled to have zero mean and unit standard deviation prior to analysis, as it was found that the scaling of the continuous variables improved the model fitting. The changes and additions made to version 3.0 make this unnecessary.

```
> method <- "BFGS"
> attr(method, which = "grad.method") <- "Richardson"
> 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,
+     method = method)
> summary(output.fn)

 Dependent variable a vector of counts.
Call:
CountsEPPM(formula = NUMBIDS ~ LEGLREST + REALREST + FINREST + WHTKNGHT +
    BIDPREM + INSTHOLD + SIZE + SIZESQ + REGULATN | LEGLREST + REALREST +
    FINREST + WHTKNGHT + BIDPREM + INSTHOLD + SIZE + SIZESQ + REGULATN,
    data = takeover.bids.case, method = method)
Model type        : mean and scale-factor
Model name        : general
Link scale-factor : log
 Coefficients (model for mean with log link)
t test of coefficients:
              Estimate  Std. Error  t value  Pr(>|t|)
(Intercept)  1.65957009  0.68434361   2.4251 0.0170129 *
LEGLREST     0.26739762  0.18056325   1.4809 0.1416256
REALREST    -0.16824674  0.29408026  -0.5721 0.5684692
FINREST      0.67843758  0.34564743   1.9628 0.0523147 .
WHTKNGHT     0.81295885  0.23202026   3.5038 0.0006753 ***
BIDPREM     -1.66155781  0.54975851  -3.0223 0.0031516 **
INSTHOLD    -0.79387774  0.43031499  -1.8449 0.0678741 .
SIZE         0.30073005  0.01630548  18.4435 < 2.2e-16 ***
SIZESQ      -0.01431148  0.00075854 -18.8671 < 2.2e-16 ***
REGULATN     0.24265669  0.22046981   1.1006 0.2735726
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
 Coefficients (model for scale-factor with log link)
t test of coefficients:
              Estimate  Std. Error t value Pr(>|t|)
(Intercept)  0.43702463  0.71749182  0.6091   0.5438
LEGLREST    -0.10510184  0.18555530 -0.5664   0.5723
REALREST     0.21145038  0.20608799  1.0260   0.3072
FINREST      0.52727653  0.50761873  1.0387   0.3013
WHTKNGHT     0.26344464  0.32522501  0.8100   0.4198
```

```
BIDPREM     -0.83563592  0.92563238 -0.9028   0.3687
INSTHOLD     0.00042329  0.02313890  0.0183   0.9854
SIZE         0.14333750  0.12561935  1.1410   0.2564
SIZESQ      -0.01544509  0.01296636 -1.1912   0.2363
REGULATN     0.26132910  0.36784034  0.7104   0.4790
log(b)      -5.40610469  6.17743959 -0.8751   0.3835
 Type of estimator: ML (maximum likelihood)
 Log-likelihood: -156.059 on 21 Df
 Number of iterations: 270 of optim method BFGS gradient method Richardson
 final gradients of parameters
 [1] -0.0039474537 -0.0071995278 -0.0002617065 -0.0001596616 -0.0022699714
 [6] -0.0062439523 -0.0018273699 -0.0170890232 -0.3991179713 -0.0020442627
[11]  0.0096255662  0.0105503277  0.0012085592  0.0021668738  0.0045170212
[16]  0.0138977697  1.3066759607  0.0142842395  0.0381147798  0.0069109067
[21]  0.0004830249
 return code 0 successful
```

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 due to the large number (21) of parameters being estimated, the flatness of the log-likelihood surface 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 of Smith and Faddy (2016) means that the probability of a zero count is very small (Equation 1 of Smith and Faddy (2016)) contributing to a better fitting model. The log-likelihood value of -156.06 for this model with 21 parameters is 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 relatively 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 their 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 significant 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) models they are not included due to being not significant. Residual plots as in Cribari-Neto and Zeileis (2010) can be produced as displayed in Figure 2.

```
> layout(matrix(c(1:6), byrow = TRUE, ncol = 2))
> plot(output.fn, which = 1, type = "response")
> plot(output.fn, which = 2, type = "pearson")
> plot(output.fn, which = 3, type = "spearson")
> plot(output.fn, which = 4, type = "likelihood")
> plot(output.fn, which = 5, type = "deviance")
> plot(output.fn, which = 6, type = "sdeviance")
```

Examination of the estimated λ_i sequences of (birth) rate parameters (Equation 2 of Smith and Faddy (2016)) shows that the λ_0 values are generally very different from the λ_i values for $i \geq 1$, as a consequence of the 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 targeted firm, and different circumstances (in the form

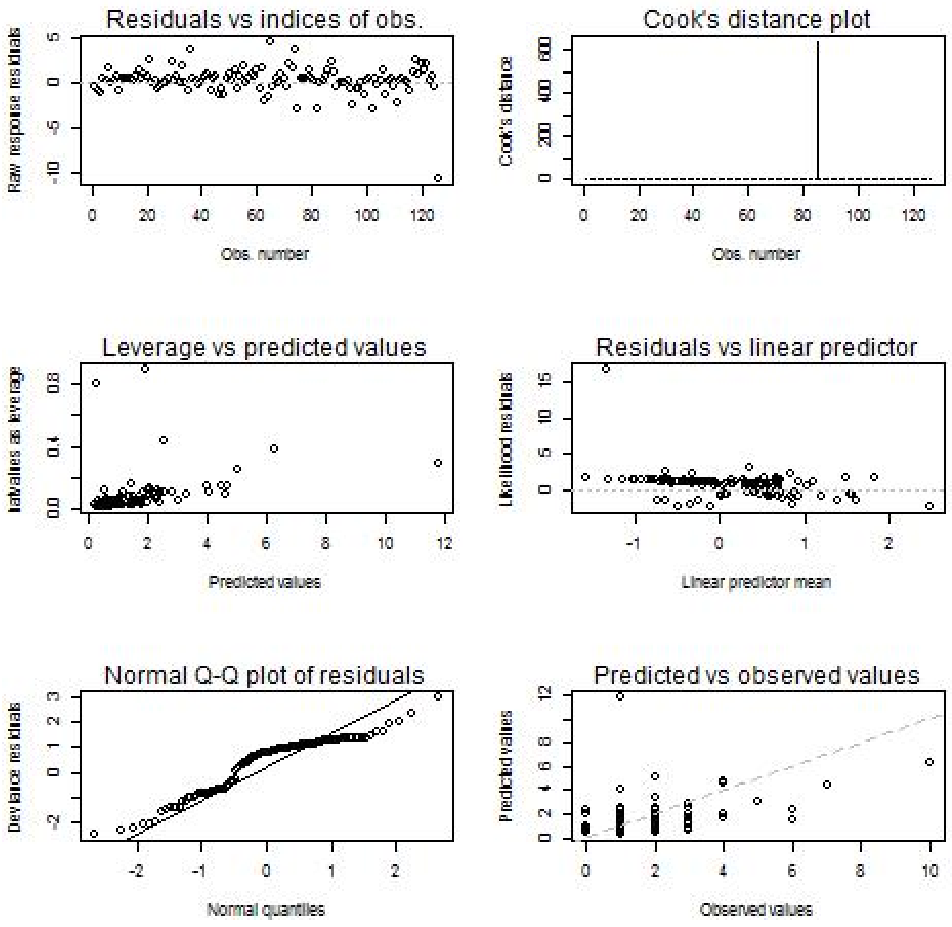

Figure 2: Residual plots.

of different covariate dependence) might be operating. Such a model is not readily constructed from those considered here.

4 Concluding remarks

This vignette has described the use of version 3.1 of the R package **CountsEPPM** to fit EPPMs to count data that exhibit under- or over-dispersion relative to the Poisson distribution. A variety of covariate dependencies and data structures are covered in examples that illustrate the variety of ways in which the package can be used in the analysis of count data. Further updates to **CountsEPPM** are in process.

As described in Faddy and Smith (2005) and Faddy and Smith (2012), the mean and scale-factor of binary data can be modeled using EPPMs in a similar way to that described here for count data. A package **BinaryEPPM** is available in CRAN. This package is being updated to include a vignette describing its use.

References

A. J. Bailer and J. T. Oris. Estimating inhibition concentrations for different response scales using generalized linear models. *Environmental Toxicology and Chemistry*, 16(7):1554–1559, 1997. doi: 10.1002/etc.5620160732.

D. Bates, M. Maechler, M. Jagan, and T. A. Davis. ***Matrix**: Sparse and Dense Matrix Classes and Methods*, 2023. URL `https://CRAN.R-project/package=Matrix`. R package version 1.6-4.

H. Bengtsson. ***R.rsp**: Dynamic Generation of Scientific Reports*, 2022. URL `https://CRAN.R-project.org/package=R.rsp`. R package version 0.45.0.

A. C. Cameron and P. Johansson. Count data regression using series expansions: With applications. *Journal of Applied Econometrics*, 12:203–223, 1997. doi: 10.1002/(sici)1099-1255(199705)12:3⟨203::aid-jae446⟩3.0.co;2-2.

A. C. Cameron and P. K. Trivedi. *Regression Analysis of Count Data*. Cambridge University Press, 2nd edition, 2013. URL `https://cameron.econ.ucdavis.edu/racd2/`.

A. Canty and B. D. Ripley. ***boot**: Bootstrap R (S-PLUS) Functions*, 2022. R package version 1.3-28.1.

F. Cribari-Neto and A. Zeileis. Beta regression in R. *Journal of Statistical Software*, 34(2):1–24, 2010. doi: 10.18637/jss.v034.i02. URL `http://www.jstatsoft.org/v34/i02/`.

M. J. Faddy. Extended poisson process modelling and analysis of count data. *Biometrical Journal*, 39(4): 431–440, 1997. doi: 10.1002/bimj.4710390405.

M. J. Faddy and D. M. Smith. Modelling the dependence between the number of trials and the success probability in binary trials. *Biometrics*, 61(4):1112–1114, 2005. doi: 10.1111/j.1541-0420.2005.00466.x.

M. J. Faddy and D. M. Smith. Extended poisson process modelling of dilution series data. *Applied Statistics*, 57(4):461–471, 2008. doi: 10.1111/j.1467-9876.2008.00622.x.

M. J. Faddy and D. M. Smith. Analysis of count data with covariate dependence in both mean and variance. *Journal of Applied Statistics*, 38(12):2683–2694, 2011. doi: 10.1080/02664763.2011.567250.

M. J. Faddy and D. M. Smith. Extended poisson process modeling and analysis of grouped binary data. *Biometrical Journal*, 54(3):426–435, 2012. doi: 10.1002/bimj.201100214.

P. Gilbert and R. Varadhan. ***numDeriv**: Accurate Numerical Derivatives*, 2019. URL `https://CRAN.R-project/package=numDeriv`. R package version 2016.8-1.1.

B. Grün, I. Kosmidis, and A. Zeileis. Extended beta regression in R: Shaken, stirred, mixed, and partitioned. *Journal of Statistical Software*, 48(11):1–25, 2012. doi: 10.18637/jss.v048.i11. URL `http://www.jstatsoft.org/v48/i11/`.

G. K. Grunwald, S. L. Bruce, L. Jiang, M. Strand, and N. Rabinovitch. A statistical model for under- or overdispersed clustered and longitudinal count data. *Biometrical Journal*, 53(4):578–594, 2011. doi: 10.1002/bimj.201000076.

J. Hilbe and A. Robinson. ***msme**: Functions and Datasets for 'Methods of Statistical Model Estimation'*, 2018. URL `https://CRAN.R-project.org/package=msme`. R package version 0.5.3.

J. M. Hilbe. *Negative Binomial Regression*. Cambridge University Press, 2nd edition, 2011.

J. M. Hilbe. *Modeling Count Data*. Cambridge University Press, 2014.

K. G. Lüning, W. Sheridan, K. H. Ytterborn, and U. Gullberg. The relationship between the number of implantations and the rate of intra-uterine death in mice. *Mutation Research*, 3:444–451, 1966. doi: 10.1016/0027-5107(66)90054-6.

M. Maechler, C. Dutang, V. Goulet, D. Bates, D. Firth, M. Shapira, and M. Stadelmann. ***expm**: Matrix Exponential*, 2023. URL `https://CRAN.R-project/package=expm`. R package version 0.999-8.

J. C. Nash. *Nonlinear Parameter Optimization Using R Tools*. John Wiley & Sons, 2014. doi: 10.1002/9781118884003.

J. A. Nelder and R. Mead. A simplex method for function minimisation. *The Computer Journal*, 7(4): 308–313, 1967. doi: 10.1093/comjnl/7.4.308.

R Core Team. *R: LanguageDefinition*. R Foundation for Statistical Computing, Vienna, Austria, 2023. URL `https://www.R-project.org/`.

A. J. Sáez-Castillo and A. Conde-Sánchez. A hyper-poisson regression model for overdispersed and underdispersed count data. *Computational Statistics & Data Analysis*, 61:148–157, 2013. doi: 10.1016/j.csda.2012.12.009.

D. M. Smith and M. J. Faddy. Mean and variance modeling of under- and overdispersed count data. *Journal of Statistical Software*, 69(6):1–23, February 2016. URL `http://www.jstatsoft.org/v69/i06/`.

D. M. Smith and M. J. Faddy. ***CountsEPPM**: Fitting of EPPM Models to Count Data*, 2018. URL `https://CRAN.R-project/package=CountsEPPM`. R package version 3.0.

D. M. Smith and M. J. Faddy. ***BinaryEPPM**: Mean and Variance Modeling of Binary Data*, 2019. URL `https://CRAN.R-project.org/package=BinaryEPPM`. R package version 2.3.

A. Zeileis and Y. Croissant. Extended model formulas in R: Multiple parts and multiple responses. *Journal of Statistical Software*, 34(1):1–13, 2010. doi: 10.18637/jss.v034.i01.

A. Zeileis and T. Hothorn. Diagnostic checking in regression relationships. *R News*, 2(3):7–10, 2002. URL `http://CRAN.R-project.org/doc/Rnews/`.

J. Zhu, J. C. Eickhoff, and M. S. Kaiser. Modelling the dependence between number of trials and success probability in beta-binomial-poisson mixture distributions. *Biometrics*, 59(4):955–961, 2003. doi: 10.1111/j.0006-341x.2003.00110.x.