Mean and Scale-Factor Modeling of Under- and Over-dispersed Grouped Binary Data

David M. Smith, Malcolm J. Faddy

May 27, 2024

Abstract

This vignette relates to version 3.0 of the R package **BinaryEPPM**. The inclusion of a vignette is part of the update from version 2.3, and describes using it in determining maximum likelihood estimates of the parameters of extended Poisson process models involving binary variables. These flexible models provide a family that can handle unlimited under-dispersion, but limited over-dispersion, with respect to the binomial distribution, with that distribution being a special case. Within **BinaryEPPM**, models with the mean and scale-factor related to covariates are constructed conforming to a generalized linear model formulation. Combining such under-dispersed models with over-dispersed models provides a general form of residual distribution for modeling grouped binary data. Use of the package is illustrated by application to several data-sets.

Keywords: binomial distribution, under-dispersion, over-dispersion, extended Poisson process models, double generalized linear 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. Smith and Faddy (2016) was concerned with generalizations of the Poisson distribution to deal with over- and under-dispersion. Similar generalizations of the binomial distribution exist, which are another special case of the modeling described in Faddy (1997). Covariate dependence can be incorporated via a re-parameterization using approximate forms of the mean and variance. The supplementary material for Faddy and Smith (2012) contained R (R Core Team, 2024) code illustrating the fitting of these models. This code has been extended and generalized to have inputs and outputs akin to those of the generalized linear model (GLM) function `glm` from the packages **stats** and **betareg** (Cribari-Neto and Zeileis, 2010; Grün et al., 2012). The distributions considered can be placed within either a single or double GLM framework. Sáez-Castillo et al. (2023) describes a double GLM model formulation for related count data.

There are many distributional models available for over-dispersed binary data, for example as listed in R package **fitODBOD** Mahendran and Wijekoon (2024). There are fewer available for under-dispersed discrete (count or binary) data. With regard to count data, this is as noted by Huang (2023) who references Sellers and Morris (2017), Zeviani et al. (2014). Although focused on the Conway-Maxwell-Poisson (CMP) distribution applied to count data under- or over dispersed with respect to the Poisson distribution, Sellers

et al. (2017) include data from Bailey (1990) as an example of under-dispersed binomial data. Although a number of the probability distributions for handling over-dispersion can admit some under-dispersion, where the residual variance is less than that corresponding to a binomial distribution, these may be rather too limited for them to be considered general models for under-dispersed data. The EPPM extended binomial complements these models by modeling under-dispersion (and limited over-dispersion). Both the mean and scale-factor (or variance) can be formulated in terms of associated covariates. Observed under-dispersed binary data can be modeled using the EPPM extended binomial distribution, leading to better fitting models, model checking diagnostics, and assessment of the precision of any estimated quantities.

Data sets can be entered as either a `data.frame` with two columns for the dependent binary variable i.e., as `r/n` where `r` is the number of successes and `n` the number of trials; or as a `list` of frequency distributions (as `list`) of counts of values of `r` with `n` given by the length of the list. Illustrations of both forms of data input are given in the examples. A range of link functions has been included. These represent ones often used in areas such as bioassay.

2 Models

2.1 Extended Poisson process model

The models described in Faddy (1997) can be summarised as describing probability distributions on $0, 1, 2, \cdots, n$ in terms of the vector of probabilities

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

where $\boldsymbol{Q}$ is an $(n+1)\times(n+1)$ bi-diagonal matrix consisting of (Poisson process) rate parameters $\lambda_i(>0)$ for $i \;=\; 0,\ 1,\ \ldots,\ n-1$ on the upper diagonal; and $-\lambda_i$ for $i \;=\; 0,\ 1,\ \ldots,\ n$ (with $\lambda_n \;=\; 0$) on the diagonal. A function of linearly decreasing λ_i's

$$\lambda_i \;=\; a(n \;-\; i),\ \text{for } i = 0, 1, 2, \ldots, n \ \text{ with } a > 0 \tag{2}$$

gives rise to the binomial distribution with probability $p \;=\; 1-\exp(-a)$. If covariates, $\boldsymbol{x}$ say, influence the response then having $\log(a) \;=\; \boldsymbol{x}^T\boldsymbol{\beta}$ (the usual linear predictor) in this binomial special case corresponds to generalized linear modeling with a complementary log-log link function (Dobson and Barnett, 2008, Chapter 7). The complementary log-log link function arises naturally from this extended Poisson process modeling, although other link functions (such as logistic) relevant to binary data could be used and have been included. Faddy and Smith (2008) considered a generalization of Equation 2

$$\lambda_i \;=\; a(n \;-\; i)^b,\ \text{with } b > 0 \tag{3}$$

resulting in distributions analogous to those from correlated binomial modeling (Kupper and Haseman, 1978) with concave sequences of λ_i's ($0 < b < 1$) corresponding to positive correlations and over-dispersion, and convex sequences ($b > 1$) to negative correlations and under-dispersion. Here approximations for the mean and variance of these distributions from Faddy (1997) are used to re-parameterize them in terms of the probability of a success p_s in a single Bernoulli trial and scale-factor f_s for the variance of the number of successes in n trials as in Equations 4 and 5.

$$p_s = \frac{\text{mean}}{n} \;\approx\; 1 \;-\; \left\{1 \;-\; an^{b-1}(1 \;-\; b)\right\}^{\frac{1}{1-b}} \tag{4}$$

$$\text{and } f_s = \frac{\text{variance}}{np_s(1 \;-\; p_s)} \;\approx\; \frac{(1 \;-\; p_s)^{2b-1} \;-\; 1}{p_s(1 \;-\; 2b)} \tag{5}$$

with substantial under-dispersion possible for large b ($f_s \to 0$ as $b \to \infty$) while over-dispersion is limited by $f_s < \frac{1}{1-p_s}$ (the value for $b \to 0$). Since the complementary probability distribution of the number of

failures will have (approximately) $f_s < \frac{1}{p_s}$ over-dispersion is effectively limited by $f_s < \max\left(\frac{1}{1-\mathrm{p_s}}, \frac{1}{\mathrm{p_s}}\right)$ with this modeling. Although technically the scale-factor cannot exceed n, this is unlikely to be a practical limitation. A simple log link can be used for covariate dependence; i.e., $\log(f_s) = \boldsymbol{x}^\top \boldsymbol{\gamma}$.

Given f_s and p_s Equation 5 can be solved for b using the R root finding function `uniroot`, then Equation 4 can be solved for a leading to

$$\lambda_i = n\left[\frac{1 - (1 - p_s)^{1-b}}{(1 - b)}\right]\left(1 - \frac{i}{n}\right)^b \tag{6}$$

from Equation 3. This parameterization based on approximate forms for the mean and variance results in the exact mean and scale-factor not being described perfectly by their respective link functions of the linear predictors but by some perturbations of these. However, for the examples discussed in the next section the effect of this on moment-based estimates is modest. The covariate coefficients $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$ describing the mean and scale-factor can be estimated by maximum likelihood from data $y_1, y_2, \ldots, y_k$ using the likelihood $p_{y_1}, p_{y_2}, \ldots, p_{y_k}$ from the probabilities in Equation 1. Alternatively, the parameter b in Equation 6 can be estimated as a nuisance parameter if there is no interest in modeling the scale-factor. Exact calculation of the mean, variance and scale-factor can be done using the probabilities in Equation 1.

2.2 Models other than EPPMs

Two often used distributional models utilizing two parameters, rather than one as with the binomial, for describing over-dispersed binary data are the correlated binomial and beta binomial distributions. These two relax different parts of the identical, independently distributed (iid) assumption, with the former allowing the outcomes of successive Bernoulli trials to be correlated; and the latter being a mixed binomial distribution where the success probability p_s is not fixed over the sequence of trials but varies according to a beta distribution. As with the EPPM, both the mean and scale-factor (or variance) can be formulated in terms of associated covariates. This means that all three distributions can be placed within the generalized linear model (GLM) framework, either as a single or double GLM. Sáez-Castillo et al. (2023) describes a double GLM model for count data.

The mean and scale-factor of a simple correlated binomial with correlation ρ between the outcomes of any two trials are np_s and $1 + \rho(n-1)$, with probability mass function as in Kupper and Haseman (1978)

$$\text{Prob(X=x)} = \binom{n}{x} p_s^x (1-p_s)^{n-x}\left\{1 + \frac{\rho}{2p_s(1-p_s)}\left[(x - np_s)^2 + x(2p_s - 1) - np_s^2\right]\right\}.$$

The beta binomial distribution can be parameterized as in Smith (1983) with probability mass function

$$\text{Prob(X=x)} = \binom{n}{x} \frac{\prod_{r=0}^{x-1}(\mu + r\theta)\prod_{r=0}^{n-x-1}(1 - \mu + r\theta)}{\prod_{r=0}^{n-1}(1 + r\theta)},$$

with mean μ $(= np_s)$ and scale-factor $1 + \frac{\theta}{(1+\theta)}(n-1)$ (Hughes and Madden, 1995). Both these distributions do admit some modest levels of under-dispersion. Bounds on the scale-factor can be determined from those given for ρ in Kupper and Haseman (1978) for the correlated binomial, and for θ in Prentice (1986) for the beta binomial.

The EPPM generalization of the binomial distribution complements these two, and potentially other distributions, as it allows quite general levels of under-dispersion but only modest levels of over-dispersion. Therefore a distribution formed by a combination of beta binomial for $f_s > 1$ and EPPM extended binomial for $f_s \leq 1$ will allow for the full range of under- and over-dispersion in observed data. With the mean and

scale-factor being dependent on covariates as discussed in the previous sub-section, continuity is assured by both the EPPM extended binomial and the two other distributions reducing to the simple binomial distribution for $f_s = 1$. Standard likelihood methods would apply as $f_s = 1$ is not on the boundary of the parameter spaces of either of the components forming the residual distribution.

3 Description of the functions

Models with two covariate dependencies linked to p_s and f_s are developed using Equations 1 and 3. The link function between p_s and the linear predictor of covariates is either logit, probit, complementary log-log, cauchit, log, loglog, double exponential, double reciprocal, power logit, or negative complementary log. The last four of these link functions are not available in `glm` or **betareg**. References to them are Ford et al. (1992), Gaudard et al. (1993), and Tibshirani and Ciampi (1983). Only a log link function is used for the scale-factor f_s. Fitting to data is done using maximum likelihood, the optimization method used being one of two of the options available in the R function `optim`, i.e., the simplex method of Nelder and Mead (1967) (`"Nelder-Mead"`), or the `"BFGS"` method which uses first derivatives. The first derivatives used in the latter method, and in calculating the hessian matrix, are numerical ones obtained using the `gradient` function of the R package **numderiv** of Gilbert and Varadhan (2019).

The R package **Formula** of Zeileis and Croissant (2010) is used to extract model information from the `formula` input to `BinaryEPPM`. Offsets are included in the formulae specifications. To avoid repeated extractions within subsidiary functions, extraction of model information such as `covariates.matrix.mean` is only done once. As iteration is involved in the model fitting, initial estimates of the parameters are needed. These can be provided in the vector `initial` with a default, if unset, of initial estimates being produced within `BinaryEPPM` by fitting a binomial model using `glm`. The matrix exponential function used for calculating the probabilities of Equation 1 is from the package **expm** of Maechler et al. (2024). Three pseudo R-squared are available, the first, is the square of the correlation between the observed and predicted GLM linear predictor values; the other two are commonly used in logistic regression, relevant references being Cox and Snell (1989) and Nagelkerke (1991).

The arguments of `BinaryEPPM` are

```
BinaryEPPM(formula, data, subset = NULL, na.action = NULL,
  weights = NULL, model.type = "p only",
  model.name = "EPPM extended binomial", link = "cloglog",
  initial = NULL, method = "Nelder-Mead",
  pseudo.r.squared = "square of correlation", control = NULL)
```

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

Argument	Description	Default
`formula`	paired formulae as in Zeileis and Croissant (2010)	
`data`	a `data.frame` or a `list`	
`subset`	subsetting commands	`NULL`
`na.action`	action taken for NAs in `data`	`NULL`
`weights`	vector if `data` is a `data.frame`	vector of ones
	a `list` if `data` is a `list`	list of lists of ones
	attributes `normalization`, `norm.to.n`	both `NULL`
`model.type`	`"p only"` (only p_s in Equation 4 modeled)	`"p only"`
	`"p and scale-factor"` (p_s and f_s modeled)	
`model.name`	`"binomial"` (`"p only"`)	`"EPPM extended binomial"`
	`"beta binomial"`	
	`"correlated binomial"`	
	`"EPPM extended binomial"`	
`link`	the GLM link function for p_s	`"cloglog"`
	`"logit"` `"probit"` `"cloglog"`	
	`"cauchit"` `"log"` `"loglog"`	
	`"doubexp"` `"doubrecip"`	
	`"negcomplog"`	
	`"powerlogit"` attribute `"power"`	`"power"` = 1
`initial`	parameter initial values vector	`glm` fit of binomial
`method`	`"Nelder-Mead"`	`"Nelder-Mead"`
	`"BFGS"` attribute `"grad.method"`	attribute `"simple"`
	which is `"simple"` or `"Richardson"`	
`pseudo.r.squared`	`"square of correlation"`	`"square of correlation"`
	`"R squared"`	
	`"max-rescaled R squared"`	
`control`	list of control parameters	see text for more detail

Table 1: Arguments of `BinaryEPPM`.

The dependent variable is either a column, or columns, where `data` is a `data.frame`; or a `list` within `data` where it is a `list`. For the latter, the response variable `list` is one of frequency distributions. Several of the example data sets are available in both forms to illustrate how to deploy them. Table 2 gives details of the fitted model object of class '`BinaryEPPM`' returned. It is a list similar to those of objects with classes '`glm`' and '`betareg`' returned by calls to `glm` and **betareg**.

Component	Description
`data.type`	`data.frame` or `list`
`list.data`	data as a `list` of frequency distributions
`call`	the call to `BinaryEPPM`
`formula`	the `formula` input
`model.type`	`"p only"` or `"p and scale-factor"`
`model.name`	as in Table 1 according to value of `model.type`
`link`	the GLM link function for p_s
`covariates.matrix.p`	matrix of covariates for p_s
`covariates.matrix.scalef`	matrix of covariates for scale-factor
`offset.p`	offset vector for p_s
`offset.scalef`	offset vector for scale-factor
`coefficients`	the estimated coefficients
`loglik`	the final log likelihood
`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
`vnmax`	a vector of number of trials
`weights`	a vector of weights
`converged`	whether converged
`iterations`	number of iterations
`method`	`"Nelder-Mead"` or `"BFGS"`
`pseudo.r.squared`	pseudo R squared value
`start`	initial estimates input
`optim`	final estimates of coefficients
`control`	control parameters of `optim`
`fitted.values`	fitted values of p_s
`y`	observed values of p_s
`terms`	model terms

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

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 precision components), a single vector of all coefficients by default
`vcov()`	associated covariance matrix (with matching names)
`predict()`	predictions (response, linear predictor p_s, linear predictor scale-factor, p_s, scale-factor, scale-factor limits, 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 '`BinaryEPPM`'.

Table 3 gives details of a set of S3 generic extractor functions for objects of class '`BinaryEPPM`'. 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, 2024) rather than **betareg**. The first four blocks refer to functions specific to **BinaryEPPM**. 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. The limits on the values of θ (beta binomial) or ρ (correlated binomial) can be obtained from function `predict` with argument `type = "distribution.parameters"`. For given values of n and p_s tables of limits can be constructed using the subsidiary function `Model.BCBinProb` of **BinaryEPPM**.

4 Examples

To run the examples as shown the package **lmtest** needs to be installed and loaded.

4.1 Data of Author word counts

These data of the number of articles in five and ten word samples are from Bailey (1990), and are referred to in Sellers et al. (2017) related to under-dispersion. Both forms of the data are available with `data("wordcount.grouped", package = "BinaryEPPM")` and `data("wordcount.case", package = "BinaryEPPM")` representing `list` and `data.frame` respectively. Sellers et al. (2017) considers only the Macaulay data of the ten word sample. The following code extracts these data in frequency distribution form from `data("wordcount.grouped", package = "BinaryEPPM")`, fits the EPPM extended binomial distribution to the `""p and scale-factor""` model; and prints out estimates, predictions and log-likelihood.

```
> Macaulay10.grouped <- list(number.words = wordcount.grouped$number.words[2])
> output.fn <- BinaryEPPM(data = Macaulay10.grouped, number.words ~
+     1 | 1, link = "logit", model.type = "p and scale-factor",
+     model.name = "EPPM extended binomial")
> print(data.frame(mean = predict(output.fn, type = "mean"),
+     variance = predict(output.fn, type = "variance"), p = predict(output.fn,
+         type = "p"), scale.factor = predict(output.fn, type = "scale.factor"),
+     lower = c(predict(output.fn, type = "scale.factor.limits")[["lower"]]),
+     upper = c(predict(output.fn, type = "scale.factor.limits")[["upper"]])),
+     row.names = FALSE)
     mean  variance         p scale.factor lower    upper
 1.049696 0.6475566 0.1049696    0.6892495     0 1.112236
> cat("\n", "log-likelihood ", logLik(output.fn), "\n")
 log-likelihood  -117.6615
```

The resulting log-likelihood is very close to those reported in Table 3 of Sellers et al. (2017) for the binomial variant of the *sum of Conway-Maxwell-Poissons* (sCMP) class of distributions. This binomial variant is the two parameter CMP distribution when a third (integer) parameter $m = 1$. The reported values are -118.319, -117.327, -117.331, -118.521 for $m = 1, 2, 3, 4$ respectively.

A saturated model analysis of the complete data set in frequency distribution form can be performed, and estimates, predictions and log-likelihood printed.

```
> output.fn <- BinaryEPPM(data = wordcount.grouped, number.words ~
+     1 + author + fsize + author * fsize | 1 + author + fsize +
+         author * fsize, model.type = "p and scale-factor",
+     model.name = "EPPM extended binomial")
> print(data.frame(author = wordcount.grouped$author, size = wordcount.grouped$fsize,
+     mean = predict(output.fn, type = "mean"), variance = predict(output.fn,
+         type = "variance"), p = predict(output.fn, type = "p"),
+     scale.factor = predict(output.fn, type = "scale.factor"),
+     lower = c(predict(output.fn, type = "scale.factor.limits")[["lower"]]),
+     upper = c(predict(output.fn, type = "scale.factor.limits")[["upper"]])),
+     row.names = FALSE)
      author size      mean  variance         p scale.factor lower    upper
    Macaulay    5 0.6105219 0.3569735 0.1221044    0.6660270     0 1.123554
    Macaulay   10 1.0507198 0.6526806 0.1050720    0.6941058     0 1.112510
```

```
  Chesterton    5 0.5841647 0.3282146 0.1168329    0.6361796     0 1.113552
  Chesterton   10 1.0899548 0.5359336 0.1089955    0.5518519     0 1.112098
> cat("\n", "log-likelihood ", logLik(output.fn), "\n")
 log-likelihood  -340.8471
```

Beta and correlated binomials can be fitted for comparison.

```
> output.fn.one <- update(output.fn, model.name = "beta binomial")
> print(data.frame(author = wordcount.grouped$author, size = wordcount.grouped$fsize,
+     mean = predict(output.fn.one, type = "mean"), variance = predict(output.fn.one,
+         type = "variance"), p = predict(output.fn.one, type = "p"),
+     scale.factor = predict(output.fn.one, type = "scale.factor"),
+     lower = c(predict(output.fn.one, type = "scale.factor.limits")[["lower"]]),
+     upper = c(predict(output.fn.one, type = "scale.factor.limits")[["upper"]])),
+     row.names = FALSE)
      author size      mean  variance         p scale.factor     lower upper
    Macaulay    5 0.6121484 0.4693571 0.1224297    0.8737047 0.8737047     5
    Macaulay   10 1.0513222 0.8407176 0.1051322    0.8936252 0.8936252    10
  Chesterton    5 0.5880414 0.4560093 0.1176083    0.8788290 0.8788290     5
  Chesterton   10 1.0882283 0.8629756 0.1088228    0.8898452 0.8898452    10
> cat("\n", "log-likelihood ", logLik(output.fn.one), "\n")
 log-likelihood  -351.7929
```

```
> output.fn.two <- update(output.fn, model.name = "correlated binomial")
> print(data.frame(author = wordcount.grouped$author, size = wordcount.grouped$fsize,
+     mean = predict(output.fn.two, type = "mean"), variance = predict(output.fn.two,
+         type = "variance"), p = predict(output.fn.two, type = "p"),
+     scale.factor = predict(output.fn.two, type = "scale.factor"),
+     lower = c(predict(output.fn.two, type = "scale.factor.limits")[["lower"]]),
+     upper = c(predict(output.fn.two, type = "scale.factor.limits")[["upper"]])),
+     row.names = FALSE)
      author size      mean  variance         p scale.factor     lower    upper
    Macaulay    5 0.6118394 0.5070221 0.1223679    0.9442282 0.9442282 2.264453
    Macaulay   10 1.0504936 0.9180692 0.1050494    0.9765240 0.9765240 2.885086
  Chesterton    5 0.5875523 0.4908914 0.1175105    0.9467368 0.9467368 2.249589
  Chesterton   10 1.0868891 0.9451298 0.1086889    0.9756115 0.9756115 2.952252
> cat("\n", "log-likelihood ", logLik(output.fn.two), "\n")
 log-likelihood  -356.9905
```

The results for the beta and correlated binomials show that the lower limits for the scale-factor have been reached and that the EPPM extended binomial is a better fit.

4.2 Data of number of rope spores in a dilution series of potato flour

These dilution series data originate from Finney (1971), where a number of samples ($n = 5$) at each of a series of dilutions of a suspension of potato flour were examined for rope spores. The data are given in Faddy and Smith (2008), Faddy and Smith (2012). Both forms of the data are available with

`data("ropespores.grouped", package = "BinaryEPPM")` and `data("ropespores.case", package = "BinaryEPPM")` representing `list` and `data.frame` respectively. All models fitted have the (approximate) p_s modeled according to the series of dilutions using a `cloglog` link function

$$p_s \;=\; \frac{\text{mean}}{n} \;\approx\; (1 \;-\; \exp(-\exp(\beta_0 \;-\; \log(\text{dilution})))) \;.$$

The preliminary analysis of these data in Faddy and Smith (2008) was based on a binomial distribution from Equation 2 with $\log(a) = \beta_0 - \log(\text{dilution})$, corresponding to the parameter a being proportional to the reciprocal of the dilution, and log(dilution) an offset. Here, $1 \;-\; \exp(-a)$ is the probability of a single sample being fertile for rope spores and $\exp(-a)$ the probability of a single sample being sterile. Fitting a binomial followed by EPPM extended binomial Equation 3 with constant b using the `data.frame` form of input

```
> output.fn <- BinaryEPPM(data = ropespores.case, number.spores/number.tested ~
+     1 + offset(logdilution), model.name = "binomial")
> output.fn.one <- update(output.fn, model.name = "EPPM extended binomial")
> summary(output.fn.one)

 Dependent variable a vector of numerator / denominator.
Call:
BinaryEPPM(formula = number.spores/number.tested ~ 1 + offset(logdilution),
    data = ropespores.case, model.name = "EPPM extended binomial")
Model type        : p only
Model name        : EPPM extended binomial
Link p            : cloglog
non zero offsets in linear predictors
 Coefficients (model for p with cloglog  link)
Coefficient of EPPM b has 1 subtracted from it
so the test is against 1 i.e., a binomial.
t test of coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  1.86624    0.14352 13.0036 1.16e-06 ***
EPPM b       8.49031    6.39010  1.3287   0.2206
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
 Type of estimator: ML (maximum likelihood)
 Log-likelihood: -3.244071 on 2 Df
 Pseudo R-squared: 0.892522 type square of correlation
 Number of iterations: 67 of optim method Nelder-Mead
 return code 0 successful
```

Likelihood ratio and Wald tests can be performed and AIC values produced to compare the models.

```
> lrtest(output.fn, output.fn.one)
Likelihood ratio test
Model 1: number.spores/number.tested ~ 1 + offset(logdilution)
Model 2: number.spores/number.tested ~ 1 + offset(logdilution)
  #Df  LogLik Df  Chisq Pr(>Chisq)
1   1 -5.5942
2   2 -3.2441  1 4.7003    0.03016 *
---
```

```
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
> waldtest(output.fn, output.fn.one)
Wald test
Model 1: number.spores/number.tested ~ 1 + offset(logdilution)
Model 2: number.spores/number.tested ~ 1 + offset(logdilution)
  Res.Df Df  Chisq Pr(>Chisq)
1      9
2      8  1 2.2057     0.1375
> AIC(output.fn, output.fn.one)
              df      AIC
output.fn      1 13.18843
output.fn.one  2 10.48814
```

The EPPM extended binomial model with constant b is superior to the binomial with significant underdispersion apparent according to the likelihood ratio test, although not according to the Wald test due to considerable asymmetry in the profile log-likelihood as a function of this parameter. The estimates of the other parameter β_0 are rather different due to the formulation of the EPPM extended binomial model in terms of the approximate mean (Equation 4), but this has only a small effect on the actual means of the fitted model.

Residual plots as in Cribari-Neto and Zeileis (2010) can be produced as displayed in Figure 1.

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

The complementary log–log link function is asymmetric about the 50% (ED50) value compared to the symmetric logit link function. To assess how a more general asymmetric link function might perform, the profile likelihood can be optimized for a power logit link function.

```
> output.fn.two <- update(output.fn.one, link = "powerlogit")
> Results <- optim(par = 1, fn = function(par, input.data,
+     ...) {
+     local.link <- "powerlogit"
+     attr(local.link, which = "power") <- par
+     slogL <- update(output.fn.two, link = local.link)$loglik
+     return(slogL)
+ }, input.data = ropespores.case, method = "Brent", lower = 1/3,
+     upper = 3, control = list(fnscale = -1), hessian = TRUE)
> se <- sqrt(-solve(Results$hessian)[1, 1])
> X2 <- round(-2 * (output.fn.one$loglik - Results$value),
+     digits = 4)
> cat(paste("\n", "power", round(Results$par, digits = 4),
+     "se", round(sqrt(-solve(Results$hessian)[1, 1]), digits = 4),
+     "log likelihood", round(Results$value, digits = 4), "\n",
+     sep = " "))
 power 2.5677 se 2.723 log likelihood -2.5956
```

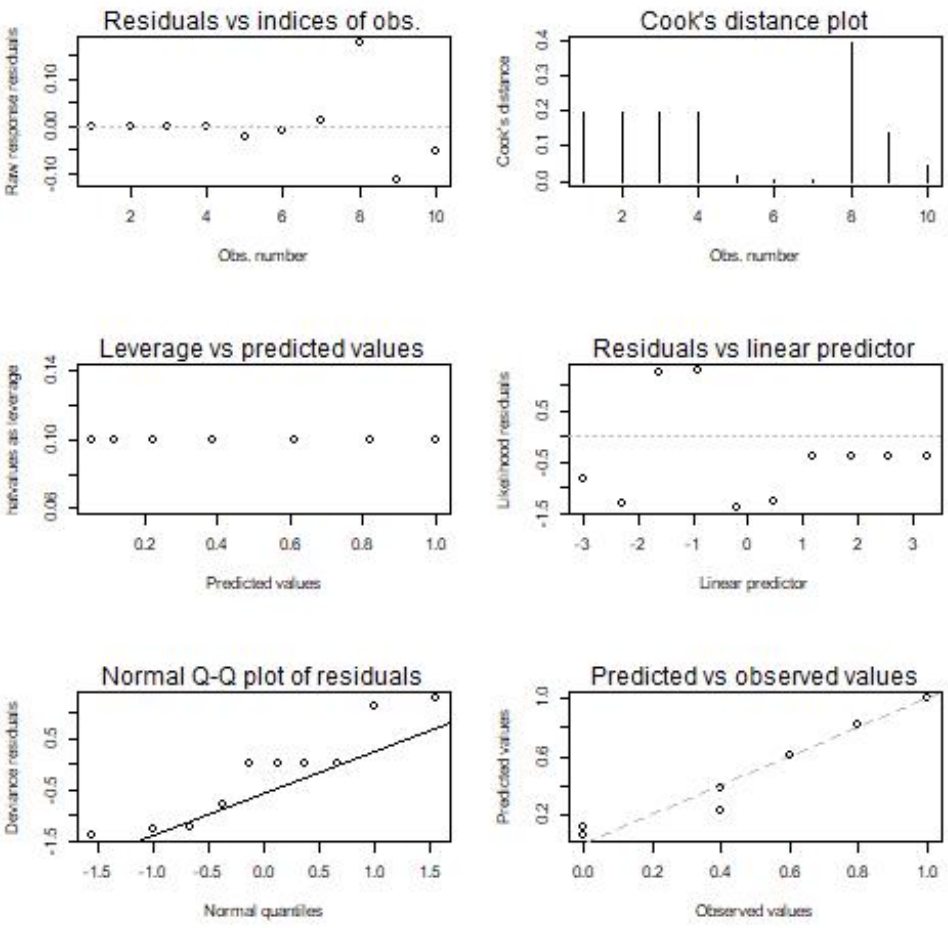

Figure 1: Linear predictor plots.

The difference in log-likelihoods here is insufficient for AIC to favor a model with a power logit link over one with the complementary log-log link.

4.3 Frequency of sex combinations in litters of pigs

The title of Brooks et al. (1991) suggests that the data show under-dispersion relative to the binomial distribution. Of the three data sets mentioned, only those for the Yorkshire breed will be used here. The fitting of a binomial distribution to these data with litter size treated as a factor with nine levels suggests that such a model is a satisfactory fit.

```
> output.fn <- BinaryEPPM(data = Yorkshires.litters, model.name = "binomial",
+     number.success ~ 0 + fsize)
> cat(paste("\n", "generalized Pearson goodness of fit statistic",
+     round(sum(residuals(output.fn, type = "pearson")^2),
+         digits = 4), "on", sum(sapply(1:length(Yorkshires.litters$number.success),
+         function(i) {
+             sum(c(Yorkshires.litters$number.success[[i]]))
+         })) - length(attr(Yorkshires.litters$fsize, which = "levels")),
+     "df", "\n", sep = " "))
 generalized Pearson goodness of fit statistic 2614.2181 on 2602 df
```

Fitting binomial and EPPM extended binomial models with the linear predictor associated with probability p_s linearly dependent on litter size as a variable and a constant scale-factor f_s would support this. However, there is an improvement in fit by allowing the linear predictor associated with the scale-factor f_s to also depend on litter size.

```
> output.fn <- BinaryEPPM(data = Yorkshires.litters, model.type = "p only",
+     model.name = "binomial", number.success ~ 1 + vsize)
> output.fn.one <- update(output.fn, model.type = "p and scale-factor",
+     model.name = "EPPM extended binomial", number.success ~
+         1 + vsize | 1)
> output.fn.two <- update(output.fn.one, number.success ~ 1 +
+     vsize | 1 + vsize)
> lrtest(output.fn, output.fn.one, output.fn.two)
Likelihood ratio test
Model 1: number.success ~ 1 + vsize
Model 2: number.success ~ (1 + vsize | 1)
Model 3: number.success ~ (1 + vsize | 1 + vsize)
  #Df  LogLik Df  Chisq Pr(>Chisq)
1   2 -4776.6
2   3 -4776.5  1 0.0726     0.7876
3   4 -4774.6  1 3.8115     0.0509 .
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

A `data.frame` of predicted summary statistics can be printed.

```
> print(data.frame(size = Yorkshires.litters$vsize, mean = predict(output.fn.two,
+     type = "mean"), variance = predict(output.fn.two, type = "variance"),
+     p = predict(output.fn.two, type = "p"), scale.factor = predict(output.fn.two,
+         type = "scale.factor"), lower = c(predict(output.fn.two,
+         type = "scale.factor.limits")[["lower"]]), upper = c(predict(output.fn.two,
+         type = "scale.factor.limits")[["upper"]])), row.names = FALSE)
 size     mean variance         p scale.factor lower    upper
    5 2.526440 1.378628 0.5052879    1.1030256     0 2.035225
    6 3.036085 1.626883 0.5060141    1.0847454     0 2.033201
    7 3.544445 1.859951 0.5063493    1.0630006     0 2.031182
    8 4.051727 2.078820 0.5064658    1.0395836     0 2.029168
    9 4.558034 2.284360 0.5064482    1.0154399     0 2.027159
   10 5.063419 2.477295 0.5063419    0.9910774     0 2.025154
   11 5.567904 2.658235 0.5061731    0.9667782     0 2.023155
   12 6.071499 2.827718 0.5059583    0.9427066     0 2.021160
   13 6.574205 2.986238 0.5057081    0.9189622     0 2.019169
```

The calculation of these exact summary statistics is done using the probabilities in Equation 1 for the EPPM extended binomial, rather than the approximate formulae in Equations 4 and 5. The following code uses `predict` to compare these approximate forms with the above predicted values of p_s and f_s. Figure 2 shows plots of these where the lines represent the approximate (linear) values and the symbols the exact (non linear) values.

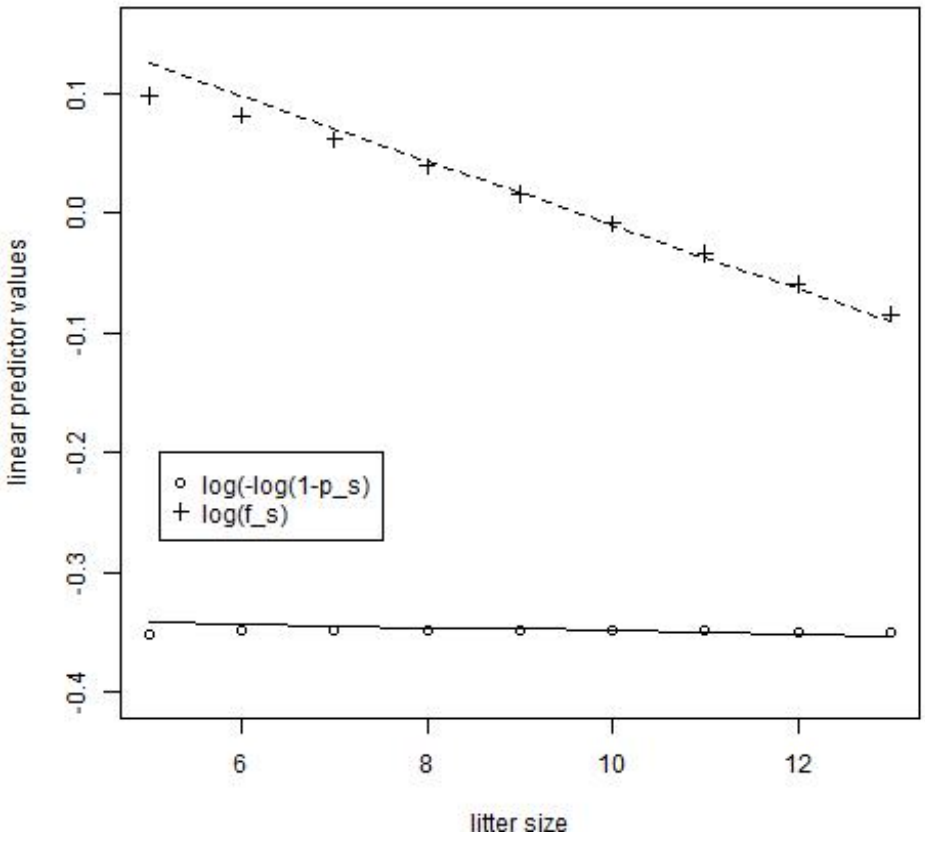

Figure 2: Linear predictor plots.

```
> approx.lp.p <- predict(output.fn.two, type = "linear.predictor.p")
> approx.lp.sf <- predict(output.fn.two, type = "linear.predictor.scale.factor")
> exact.lp.p <- log(-log(1 - predict(output.fn.two, type = "p")))
> exact.lp.sf <- log(predict(output.fn.two, type = "scale.factor"))
> plot(x = c(5, 13), y = c(-0.4, 0.15), xlab = "litter size",
+     ylab = "linear predictor values", type = "n")
> lines(x = Yorkshires.litters$vsize, y = approx.lp.p, lty = 1)
> points(x = Yorkshires.litters$vsize, y = exact.lp.p, pch = 1)
> lines(x = Yorkshires.litters$vsize, y = approx.lp.sf, lty = 2)
> points(x = Yorkshires.litters$vsize, y = exact.lp.sf, pch = 3)
> legend(5.1, -0.2, legend = c("log( -log(1 - p_s)", "log(f_s)"),
+     pch = c(1, 3), cex = 1)
```

At least for these data, the approximations are numerically close, but more importantly the exact values show only minor perturbations from linearity. The data from the first five litters sizes show scale-factors greater than one and the data from the last four show scale-factors less than one. The following shows how a combined model with beta binomial for the over-dispersed litter sizes and EPPM extended binomial for the under-dispersed litter sizes can be fitted.

```
> in.par <- c(output.fn.two$coefficients$p.est, output.fn.two$coefficients$scalef.est)
```

```
> Results <- optim(par = in.par, fn = function(in.par, in.data,
+     model.names, subsets, ...) {
+     subset1 <- BinaryEPPM(data = in.data, model.type = "p and scale-factor",
+         model.name = model.names[1], subset = subsets[[1]],
+         initial = in.par, number.success ~ 1 + vsize | 1 +
+             vsize, control = list(maxit = 1))
+     subset2 <- BinaryEPPM(data = in.data, model.type = "p and scale-factor",
+         model.name = model.names[2], subset = subsets[[2]],
+         initial = in.par, number.success ~ 1 + vsize | 1 +
+             vsize, control = list(maxit = 1))
+     slogL <- logLik(subset1) + logLik(subset2)
+     attr(slogL, which = "df") <- attr(logLik(subset1), which = "df") +
+         attr(logLik(subset2), which = "df")
+     attr(slogL, which = "nobs") <- attr(logLik(subset1),
+         which = "nobs") + attr(logLik(subset2), which = "nobs")
+     return(slogL)
+ }, in.data = Yorkshires.litters, model.type = c("p and scale-factor",
+     "p and scale-factor"), model.names = c("beta binomial",
+     "EPPM extended binomial"), subsets = list(c(1:5), c(6:9)),
+     control = list(fnscale = -1), hessian = TRUE)
> cat("\n", "log-likelihood ", logLik(output.fn.two), "\n")
 log-likelihood  -4774.636
```

The resulting parameter estimates together with their standard errors can be printed.

```
> print(data.frame(parameters = c("intercept p", "slope p",
+     "intercept scale factor", "slope scale factor"), Results$par,
+     se = c(sqrt(diag(solve(Results$hessian))))), row.names = FALSE)
             parameters   Results.par          se
            intercept p -0.3443024694 0.015661870
                slope p -0.0005205837 0.001835825
 intercept scale factor  0.2083432559 0.076138072
     slope scale factor -0.0220887354 0.008843641
```

A `data.frame` of predicted summary statistics can be printed.

```
> first.subset <- BinaryEPPM(data = Yorkshires.litters, subset = 1:5,
+     model.type = "p and scale-factor", model.name = "beta binomial",
+     number.success ~ 1 + vsize | 1 + vsize, initial = Results$par,
+     control = list(maxit = 1))
> second.subset <- BinaryEPPM(data = Yorkshires.litters, subset = 6:9,
+     model.type = "p and scale-factor", model.name = "EPPM extended binomial",
+     number.success ~ 1 + vsize | 1 + vsize, initial = Results$par,
+     control = list(maxit = 1))
> print(data.frame(size = Yorkshires.litters$vsize, mean = c(predict(first.subset,
+     type = "mean"), predict(second.subset, type = "mean")),
+     variance = c(predict(first.subset, type = "variance"),
+         predict(second.subset, type = "variance")), p = c(predict(first.subset,
+         type = "p"), predict(second.subset, type = "p")),
+     scale.factor = c(predict(first.subset, type = "scale.factor"),
```

```
+         predict(second.subset, type = "scale.factor")), lower = c(predict(first.subset,
+         type = "scale.factor.limits")[["lower"]], predict(second.subset,
+         type = "scale.factor.limits")[["lower"]]), upper = c(predict(first.subset,
+         type = "scale.factor.limits")[["upper"]], predict(second.subset,
+         type = "scale.factor.limits")[["upper"]])), row.names = FALSE)
 size     mean variance         p scale.factor     lower    upper
    5 2.534078 1.378309 0.5068156    1.1028520 0.4374562 5.000000
    6 3.039805 1.617853 0.5066342    1.0787585 0.4526227 6.000000
    7 3.545169 1.846277 0.5064527    1.0551913 0.4622157 7.000000
    8 4.050170 2.063953 0.5062713    1.0321390 0.4688047 8.000000
    9 4.554809 2.271241 0.5060899    1.0095903 0.4735900 9.000000
   10 5.060765 2.471189 0.5060765    0.9886216 0.0000000 2.023917
   11 5.567677 2.663229 0.5061524    0.9685937 0.0000000 2.023174
   12 6.074282 2.845601 0.5061901    0.9486791 0.0000000 2.022432
   13 6.580583 3.018647 0.5061987    0.9289574 0.0000000 2.021691
```

The predicted p_s and scale factor with its limits for a litter size of 14 can be produced from the fitted model, illustrating use of the `newdata` argument of `predict`.

```
> newdata <- data.frame(vsize = 14, vnmax = c(14), mean.p = Results$par[[1]],
+     mean.scalef = Results$par[[2]])
> print(data.frame(size = newdata$vsize, p = predict(second.subset,
+     newdata, type = "p"), scale.factor = predict(second.subset,
+     newdata, type = "scale.factor"), lower = predict(second.subset,
+     newdata = newdata, type = "scale.factor.limits")[["lower"]],
+     upper = predict(second.subset, newdata = newdata, type = "scale.factor.limits")[["upper"]]),
+     row.names = FALSE)
 size         p scale.factor lower    upper
   14 0.5061843      0.90948     0 2.025047
```

5 Concluding remarks

Although this vignette is focused on under-dispersion, the package **BinaryEPPM** can be used to fit EPPMs to grouped binary data exhibiting both under- and/or over-dispersion relative to the binomial distribution. A variety of covariate dependencies and data structures are covered in examples that provide illustrations of the ways in which **BinaryEPPM** can be used in the analysis of grouped binary data, particularly those showing under-dispersion. It complements the similar modeling in Smith and Faddy (2016) of count data using EPPMs. Package **CountsEPPM** (Smith and Faddy, 2024) is available on the Comprehensive R Archive Network (CRAN) as a contributed package at `https://CRAN.R-project.org/package=CountsEPPM`.

References

B. J. R. Bailey. A model for function word counts. *Applied Statistics*, 39(1):107–1143, 1990.

R. J. Brooks, W. H. James, and E. Gray. Modelling sub-binomial variation in the frequency of sex combinations in litters of pigs. *Biometrics*, 47(2):403–417, 1991. doi: 10.2307/2532134.

A. Canty and B. D. Ripley. ***boot**: Bootstrap R (S-PLUS) Functions*, 2024. R package version 1.3-30.

D. R. Cox and E. Snell. *Analysis of Binary Data*. Chapman & Hall, 2nd edition, 1989.

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.

A. J. Dobson and A. Barnett. *An Introduction to Generalized Linear Models*. Chapman & Hall, 3rd edition, 2008.

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. Extended poisson process modelling of dilution series data. *Journal of the Royal Statistical Society C*, 57(4):461–471, 2008. doi: 10.1111/j.1467-9876.2008.00622.x.

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

D. J. Finney. *Statistical Methods in Biological Assay*. Griffin, 2nd edition, 1971.

I. Ford, B. Torsney, and C. Wu. The use of a canonical form in the construction of locally optimal designs for non–linear problems. *Journal of the Royal Statistical Society B*, 54(2):569–583, 1992. doi: 10.1111/j.2517-6161.1992.tb01897.x.

M. A. Gaudard, M. J. Karson, E. Linder, and S. K. Tse. Efficient designs for estimation in the power logistic quantal response model. *Statistica Sinica*, 3(1):233–243, 1993.

P. Gilbert and R. Varadhan. ***numDeriv****: Accurate Numerical Derivatives*, 2019. URL `http://optimizer.r-forge.r-project.org/`. 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.

A. Huang. On arbitarily underdispersed discrete distributions. *The American Statistician*, 77(1):29–34, 2023. doi: 10.1080/00035.2022.2106305.

G. Hughes and L. V. Madden. Some methods allowing for aggregated patterns of disease incidence in the analysis of data from designed experiments. *Plant Pathology*, 44(6):927–943, 1995. doi: 10.1111/j.1365-3059.1995.tb02651.x.

L. L. Kupper and J. K. Haseman. The use of a correlated binomial model for the analysis of toxicological experiments. *Biometrics*, 34(1):69–76, 1978. doi: 10.2307/2529589.

M. Maechler, C. Dutang, V. Goulet, D. Bates, D. Firth, M. Shapira, and M. Stadelmann. ***expm****: Matrix Exponential*, 2024. URL `https://CRAN.R-project/package=expm`. R package version 0.999-9.

A. Mahendran and P. Wijekoon. ***fitODBOD****: Modeling Over Dispersed Binomial Outcome Data*, 2024. URL `https://github.com/Amalan-ConStat/fitODBOD`. R package version 1.5.1.

N. J. D. Nagelkerke. A note on a general definition of the coefficient of determination. *Biometrika*, 78: 691–692, 1991.

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

R. L. Prentice. Binary regression using an extended beta-binomial distribution, with discussion of correlation induced by covariate measurement errors. *Journal of the American Statistical Association*, 81(394): 321–327, 1986. doi: 10.1080/01621459.1986.10478275.

R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2024. URL `https://www.R-project.org/`.

A. J. Sáez-Castillo, A. Conde-Sánchez, and F. Martínez. Dglmextpois: Advances in dealing with over and under-dispersion in a double glm framework. *The R Journal*, 14:121–140, 2023. ISSN 2073-4859. doi: 10.32614/RJ-2023-002. https://doi.org/10.32614/RJ-2023-002.

K. F. Sellers and D. S. Morris. Underdispersion models: Models that are "under the radar". *Communications in Statistics - Theory and Methods*, 46(24):12075–12086, 2017. doi: 10.1080/03610926.2017.1291976.

K. F. Sellers, A. W. Swift, and K. S. Weems. A flexible distribution class for count data. *Journal of Statistical Distributions and Applications*, 41(12):2616–2626, 2017. doi: 10.1080/02664763.2014.922168.

D. M. Smith. Algorithm as189. maximum likelihood estimation of the parameters of the beta binomial distribution. *Journal of the Royal Statistical Society C*, 32(2):196–204, 1983.

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, 2016. doi: 10.18637/jss.v069.i06.

D. M. Smith and M. J. Faddy. ***CountsEPPM**: Fitting of EPPM Models to Count Data*, 2024. URL `https://CRAN.R-project/package=CountsEPPM`. R package version 3.1-1.

R. J. Tibshirani and A. Ciampi. A family of proportional- and additive-hazards models for survival data. *Biometrics*, 39(1):141–147, 1983. doi: 10.2307/2530814.

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.

W. M. Zeviani, P. J. R. Jr, W. H. Bonat, S. E. Shimakura, and J. A. Muniz. The gamma-count distribution in the analysis of experimental underdispersed data. *Journal of Applied Statistics*, 4(22), 2014. doi: 10.1186/s40488-017-0077-0.