Models for overdispersed count data

1503064

May 3, 2018

1 Abstract

The Poisson distribution is first the port of call when modelling count data. A key property of the Poisson distribution is that the variance is equal to the mean. When we fit a Poisson model to count data, therefore, we should expect to see that the conditional variance is equal to the conditional mean. If this property does not hold, we have extradispersion; most commonly, the conditional variance is greater than the conditional mean, in which case we have overdispersion. True overdispersion may undermine the model entirely, making its standard errors incorrect and any conclusions drawn from it erroneous. Apparent overdispersion may be due to misspecification of the model (through omission of a significant predictor or interaction term), or it may be because the model family is inappropriate. In the latter case, there are several alternative families of model which can be investigated to find one more suitable. We consider quasipoisson and negative binomial models including the hurdle variant, and explore the use of rootograms for count model fit diagnostics via simulation and application to a real world data set.

2 Introduction

What is count data? For the purposes of this article, *count data* is taken to be a collection of realisations of a random variable taking only non-negative integer values, arising from observing a sequence of events or from enumerating items falling into a set of categories. Theoretically there is no upper bound, although clearly in practice any sample has one. This kind of data arises in many natural contexts; the specific example used in this article is counts of "likes" accrued by posts on Facebook.

How do we model count data? The Poisson model is the underlying distribution for most count models. The simple Poisson model, however, is usually inadequate to model real data[1]. We consider various extensions to the Poisson here, including quasipoisson, negative binomial and hurdle negative binomial models.

What is overdispersion? In order to legitimately model data as coming from a Poisson distribution, we need the mean and variance of the response, conditional on the predictors, to be equal (equidispersion). If this property does not hold for a given set of count data, the data are Poisson extradispersed. If the conditional variance is greater than the mean, the data are overdispersed; if less, the data are underdispersed. In practice, overdisperson is by far the more common case[1]. In general, extradispersion is measured against the distribution; for example, data is overdispersed with respect to a given distribution if its conditional variance is greater than expected for data drawn from that distribution.

What are the effects of overdispersion? If a model is overdispersed, *we cannot rely on the model to appropriately inform us about the response*[1]. Any conclusion based on the coefficients or standard errors of an overdispersed model is highly suspect.

What causes overdispersion? Hilbe[1] highlights the difference between *apparent* and *true* overdispersion.

Apparent overdispersion can occur when the model is incorrectly specified (through omitting important explanatory predictors or interaction terms) or when the data contains outliers or non-random missing values. This is briefly explored via simulation below.

True overdispersion occurs when those factors have been considered, yet the assumptions of the underlying distribution are still violated. There are many possible context-specific reasons for this.

For example, McCullagh and Nelder [3] point out that in behavioural studies involving animals, incidents tend to occur in clusters, violating the assumption of independence. It is easy to see how this general principle applies to our Facebook data set as each like exposes more users to the post via Facebook's engagement-driven post sharing algorithm. Another highlighted possibility is that the data come from a Poisson process observed over an interval of random length; in our case, note that the inner workings of the Facebook algorithm are mysterious and it is reasonable to assume there are effects which can't easily be measured influencing how long posts remain sufficiently visible to users to attract likes - in other words, if we assume that there is a primary time interval during which each post receives likes, the length of this interval is not constant across posts.

Zero counts A common way in which data deviates from a Poisson model is an excess or absence of zero counts. When there are excessive zeros, the data may contain two types of zero: true zeros, which occur when a valid count is taken and the value is zero, and excess zeros, where the count is necessarily equal to zero due to a separate process. For example, in a count of customers entering a bank per hour we should distinguish between the zeros which occur when the building is open (true zeros), and those when the building is closed (excess zeros).

Alternatively, the way the count is taken may preclude the presence of zeros at all; Hilbe gives the example of the length (in days) of a hospital stay.

Zero-inflated, zero-truncated models and hurdle models have been developed to assist in these situations.

Assessing Overdispersion There are a variety of means of assessing overdispersion in Poisson models. Dispersion statistics can be derived from the model via residuals or comparison of log likelihood with the saturated model; there is a brief exploration of the behaviour of these statistics via simulation below, followed by application to our data. Kleiber and Zeileis[2] propose the rootogram as a graphical means of assessing extradispersion. Examples are given below of rootograms from well- and poorly- fitting simulated models, and then rootograms are produced for our data to assess the fit of a range of candidate models.

3 Main Body

3.1 Overview of techniques

3.1.1 Simulated Data

The simulation process here was inspired by Hilbe[1]:

Coefficients $\beta_0, \beta_1, \ldots, \beta_p$ were specified, and then for each of n simulated observations, the p predictor variables were sampled from a Uniform distribution (i.e. $x_{i,1}, \ldots, x_{i,p}$ are realisations of the random variables $X_1, \ldots, X_p \sim \text{Uniform}(0,1)$, for $i = 1, \ldots, n$); from this we produce n mean values taking $\mu_i = e^{\beta_0 + \sum_{i=1}^{p} x_i \beta_i}$ for $i = 1, \ldots, n$. The simulated responses are then realisations of random variables with expectation μ_i; e.g. y_i as a realisation of $Y_i \sim Poisson(\mu_i), i = 1, \ldots, n$.

When simulating a two-step process producing inflated zeros, a probability $\zeta \in (0,1)$ was specified; then for each observation, if $Z_i \sim Uniform(0,1) < \zeta$, the value y_i was set to 0.

Four simulated data sets were generated from a fixed seed, with 5000 observations of data using the following coefficients: $\beta_0 = 2, \beta_1 = 1, \beta_2 = -1, \beta_3 = 0.5$. Each data set also included an extra noise variable which was not used in generating the response (i.e. $\beta_4 = 0$).

Data sets *sim.1, sim.2* were generated using the Poisson distribution, while data sets *sim.3, sim.4* were generated using the Negative Binomial distribution with $\theta = 2$. Data sets *sim1, sim.3* contained no excess zeros; data sets *sim2, sim.4* contained excess zeros with $\zeta = 0.1$.

The R function used to produce the data sets is included in the appendix.

3.1.2 Dispersion Statistics

Two primary numerical measures of dispersion are considered here. McCullagh and Nelder[3] recommend estimation of the dispersion using the Pearson $\mathcal{X}^2$ Statistic:

$$\hat{\sigma}_p^2 = \frac{\mathcal{X}^2}{n-p}$$

Alternatively, a statistic based on the deviance can be used:

$$\hat{\sigma}_d^2 = \frac{D}{n-p},$$

where D is the difference between the saturated model log-likelihood and candidate model log-likelihood. Hilbe[1] states a preference for the Pearson-based dispersion statistic, noting bias in the deviance statistic in some cases. The performance of both statistics is briefly assessed via simulation below, but in both cases, a value below 1 indicates underdispersion, while a value above 1 suggests overdispersion.

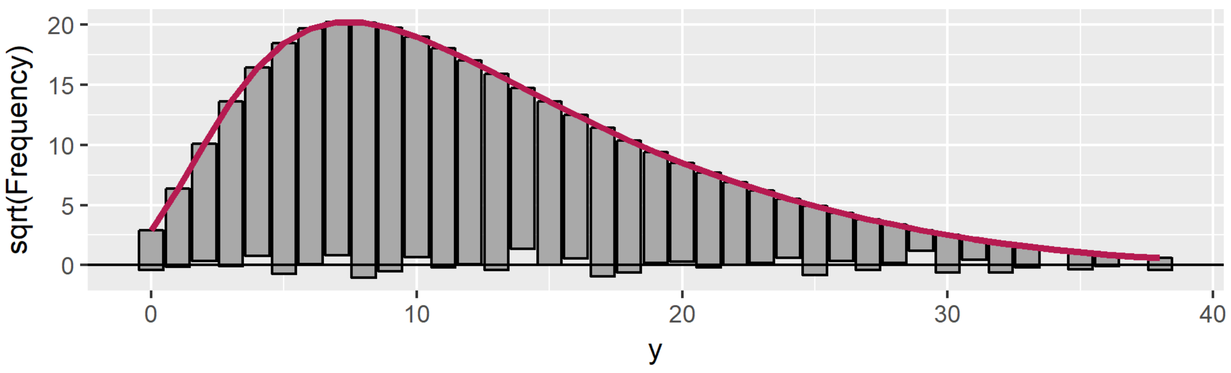

Figure 1: Hanging rootogram produced from Poisson model fitted to true Poisson data.

3.1.3 Rootograms

Rootograms were produced using the *countreg*[5] R package. The hanging style was used as recommended by Kleiber and Zeileis[2]. The red line gives the expected number of observations if the model is well-fit, while the hanging bars give the observed number. The fit can be assessed by looking across the horizontal zero line; each hanging bar should, for a well-fitting model, end close to the line. Bars hanging below the line indicate under-prediction at that value of y, while bars above the line indicate over-prediction. Rootograms produced by different types of poor fit are explored below.

3.2 Assessment of Models

3.2.1 Poisson

We first explore the behaviour of a simple Poisson model; $Y_i \sim Poisson(\mu_i)$, where $\mu_i = e^{\beta_0 + \sum_{j=1}^{p} \beta_i x_{i,j}}$.

First, a Poisson GLM including all variables (actual predictors $x1, x2, x3$ and the noise term $x4$) was fitted to *sim.1*. The fit is excellent, as expected: the estimated coefficients were $\hat{\beta}_0 = 1.99, \hat{\beta}_1 = 0.97, \hat{\beta}_2 = -0.99, \beta_3 = 0.50, \beta_4 = 0.02$; noise term $x4$ was correctly not considered significant. The dispersion statistics were $\hat{\sigma}_p^2 = 0.997$, $\hat{\sigma}_d^2 = 1.021$, indicating equidispersion. The rootogram Figure 1 shows the fit; the hanging lines finish very close to the zero line at almost all values of y.

A second Poisson GLM omitting significant predictors was fitted to *sim.1* with the formula $\mu_i = e^{\beta_0 + \beta_1 x_{i,1}}$. The coefficient estimates were $\hat{\beta}_0 = 1.82, \hat{\beta}_1 = 0.96$; $\hat{\beta}_1$ is close to the true value of β_1 but $\hat{\beta}_0$ is unsurprisingly less accurate. Dispersion statistics were $\hat{\sigma}_p^2 = \hat{\sigma}_d^2 = 2.06$, demonstrating how omission of significant predictors leads to over-dispersion detectable by those statistics. The rootogram is Figure 2; note the characteristic wave pattern along the horizontal zero line identified by Kleiber and Zeilis. The overdispersed model is generally predicting values too close to the mean, and hence over-predicting central and under-predicting low and high counts.

The full Poisson GLM was then fit to data set *sim.2* containing excess zeros. The model had dispersion statistics $\hat{\sigma}_p^2 = 2.02$ and $\hat{\sigma}_d^2 = 2.86$. The dispersion statistics alone will not tell us the difference between the above case where a model is fit to Poisson-generated data with a missing

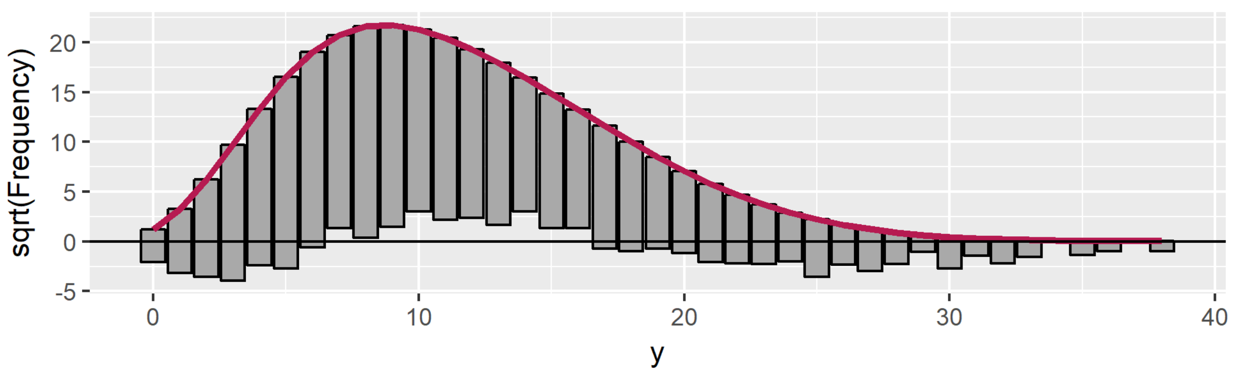

Figure 2: Hanging rootogram produced from Poisson model with significant predictors omitted.

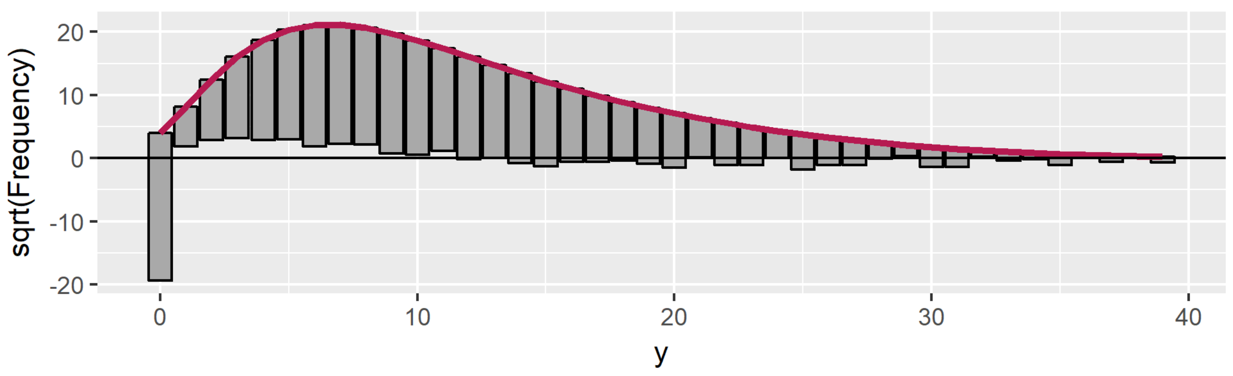

Figure 3: Hanging rootogram from Poisson model fitted to data containing excess zeros.

significant predictor, and this case where all relevant predictors are included but there are excess zeros in the data. The rootograms however immediately reveal the diffence; Figure 3 shows a clear underprediction spike at $y = 0$ which is absent from Figure 2.

Finally a Poisson GLM including all variables was fit to data set *sim.3*, which was generated from the Negative Binomial distribution. The coefficient estimates were tolerably accurate ($\hat{\beta}_0 = 1.91, \hat{\beta}_1 = 1.08, \hat{\beta}_2 = -0.96, \hat{\beta}_3 = 0.56, \hat{\beta}_4 = -0.04$) but $x4$ was considered significant with p-value 0.0136, despite being pure noise. The dispersion statistics were $\hat{\sigma}_p^2 = 6.28$, $\hat{\sigma}_d^2 = 5.85$. The overdispersion has led to underestimated standard errors and hence to incorrect significance estimates. The rootogram is in Figure 4; from this we can see that the Poisson model underpredicts severely at low counts, overpredicts towards the mean and then underpredicts for high counts, with predictions falling to zero long before positive counts stop appearing in the data.

Fitting a full Poisson GLM to the Facebook data set leads to significant parameter estimates for all available predictors. The AIC was 10556.26. The dispersion statistics are $\hat{\sigma}_p^2 = 23.9, \hat{\sigma}_d^2 = 17.1$; the data is severely overdispersed. The rootogram for this model is Figure 5. The qualitative similarity between Figures 4 and 5 suggests that the Poisson model fails to fit the Facebook data similarly to how it fails to fit data from a negative binomial distribution, which suggests a negative binomial model as a candidate for modelling the Facebook data.

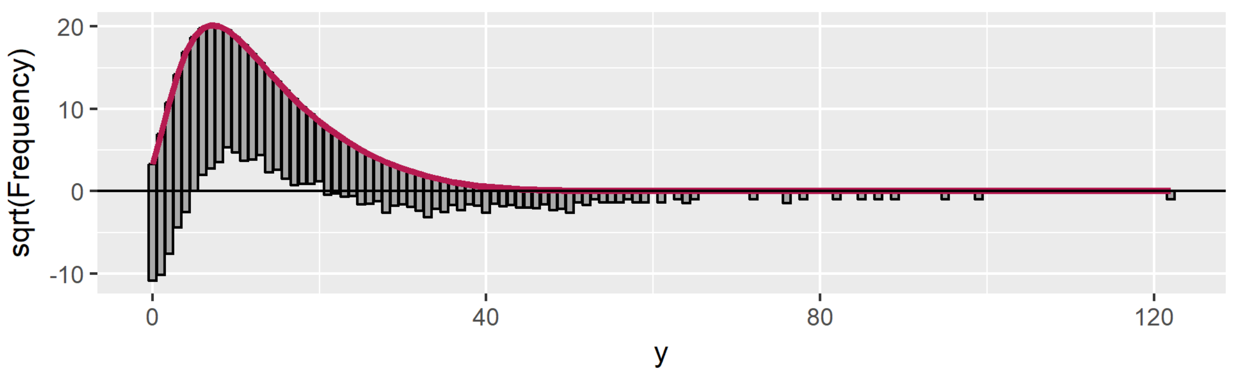

Figure 4: Rootogram for Poisson GLM fit to data generated from Negative Binomial distribution.

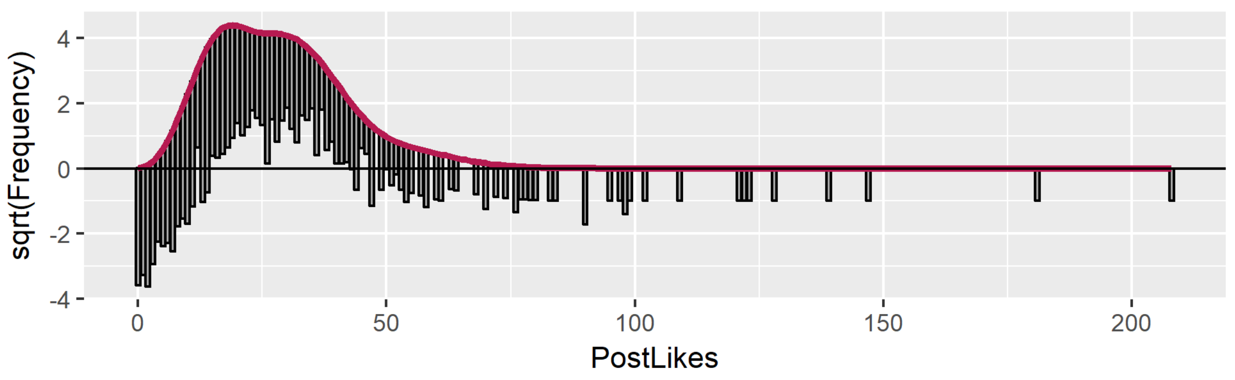

Figure 5: Rootogram for Poisson GLM fit to Facebook data.

3.2.2 Quasi-Poisson

One method of correcting for the effects of overdispersion on estimates is the use of a quasipoisson (QP) model, which introduces an overdispersion parameter $\theta > 1$ such that we have $Var(Y) = \theta\mu$, i.e. assuming that the variance is proportional to the mean.[4]. The effect in practice is to increase the standard errors in the model by the square root of the Pearson dispersion statistic of the standard Poisson model. QP models do not necessarily have a distributional form or likelihood and hence we cannot derive statistics such as AIC for model selection; while quasi-AIC measures have been developed they should not be used to compare quasi- and true likelihood models[4]. The countreg package does not offer support for producing rootograms for QP models, although the predictions are the same as for a Poisson model.

A QP model including all variables was fit to data set *sim.3*. The increased standard errors led to noise variable $x4$ being correctly found not significant, unlike the standard Poisson model above.

A QP model including all variables was then fit to the Facebook data set; the PageLikes, Month and Paid variables were not significant under this model; this is perhaps analogous to the elimination of the noise variable $x4$ in the simulated case. Predictions from the QP model are identical to those from the Poisson model.

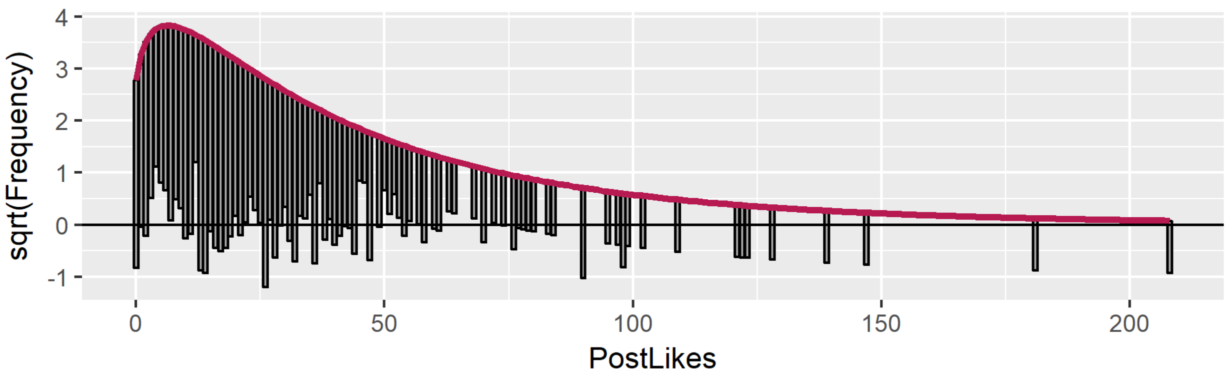

Figure 6: Rootogram for Negative Binomial GLM fit to Facebook data

3.2.3 Negative Binomial

The Negative Binomial distribution allows for variance different to the mean. The distribution can be parametrised in various ways; in one such, we have for $E(Y) = \mu, Var(Y) = \mu + \frac{\mu}{\theta}$, so as $\theta \to \infty$, we return to the equidispersion relationship of the Poisson. In practice, clearly, we would work with a large finite value of θ to approximate a Poisson by the negative binomial in this parametrisation.

McCullagh and Nelder discuss the way in which a mixture of Poisson and Gamma distributions leads to the negative binomial distribution. In particular, if each observation in a sample is the realisation of a Poisson random variable whose mean is itself the realisation of a Gamma distributed random variable, this mixture leads to the negative binomial[3]. In the context of our Facebook data, we could plausibly consider each post to have a hypothetical hidden *likeability* score distributed as a Gamma random variable with parameters determined by post characteristics, which then acts as the mean for the Poisson distribution of the number of likes received.

A negative binomial model including all variables was fitted to the Facebook data set. *PageLikes*, *Month* and *Paid* were not found to be significant under this model; the AIC was 4152. Refitting the model excluding those variables led to an improved AIC of 4147. The rootogram for this model is given in Figure 6. The fit is clearly far superior to that of the Poisson model; only a few values of *PostLikes* are under- or over-predicted by more than 1, although the issue of occasional posts with far more likes than predicted remains. One of the values being under-predicted is zero, which brings us to the next model.

3.2.4 Hurdle Negative Binomial

As discussed in the introduction, when there are excess zeros in our count we imagine a two-stage generation process; firstly, a process which determines whether the count is zero, and then a subsequent process which determines the value of the count *given that it is greater than zero.* If we consider this as a random variable Y which is a function of a Bernoulli random variable Z and

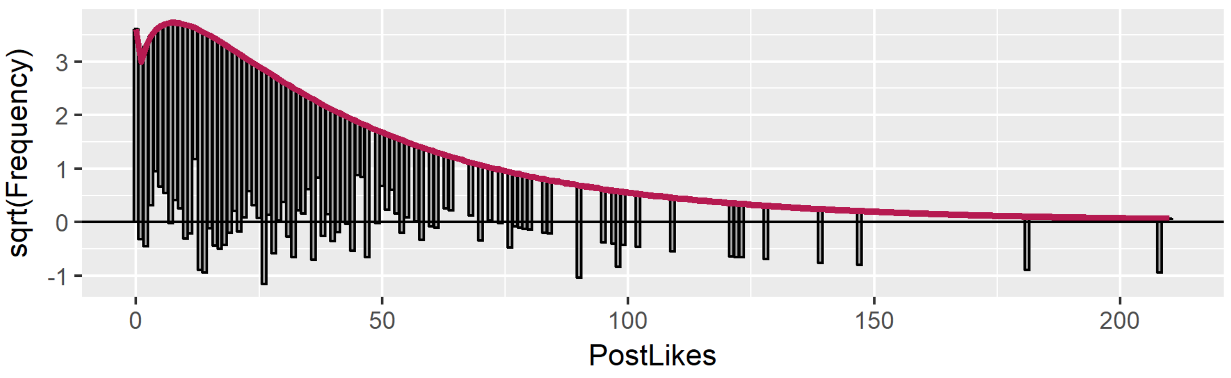

Figure 7: Rootogram for Hurdle Negative Binomial GLM fit to Facebook data

another random variable X taking values in the non-negative integers, we have

$$\mathbb{P}(Y = y) = \begin{cases} \mathbb{P}(Z = 0), & \text{if } y = 0, \\ \mathbb{P}(Z = 1, X = y), & \text{if } y > 0. \end{cases}$$

This situation can be modelled by a hurdle negative binomial model, which is a mixture model consisting of two components; the first component is a binary model (e.g. binary logistic), and the second a truncated negative binomial model (i.e. a model based on the negative binomial distribution rebalanced to give 0 counts a probability of 0). We can consider the effect of each predictor on each model separately.

A hurdle negative binomial model including the predictors *Type*, *Category* and *Weekend* was fitted to the Facebook data set using *hurdle* method provided by the R package *countreg*[5]. The AIC was 4140, slightly superior to the standard negative binomial model. In the binary model, the only significant predictor was *Weekend*, with a coefficient of -1.3 suggesting likes were more likely to be zero at weekends (the odds of positive likes decrease by approximately 20% at weekends). All predictors were significant in the truncated negative binomial model. The rootogram is Figure 7; comparison with Figure 6 shows it to be almost identical, but with an improved fit at 0, as expected.

4 Discussion / Conclusion

For this Facebook dataset, the best-fitting model as assessed by AIC and rootogram is the hurdle negative binomial (HNB) model. Both measures of fit show only a small improvement over the NB model, however; whether the additional complexity of the two-stage model is justified by this small improvement is open to debate. It is easier to interpret the coefficients of the NB model than those of the two-stage HNB model. In general, the NB model offers an effective way of dealing with overdispersion of the kind found in this data.

There is agreement between QP and NB as to which predictors are significant for the Facebook data.

The QP model more accurately assesses which predictors are significant than the Poisson model when fit to data simulated from a NB distribution, which is overdispersed with respect to the Poisson.

In the NB model, the variance is a quadratic function of the mean, while in the QP model the variance is a linear function of the mean. Ver Hoef and Boveng[4] showed that the effect in practice is that small counts are given more weight in NB regression and large counts more in QP. Which model is more appropriate in this regard depends on the context; in this case it appears that the NB model is producing better predictions. The NB model requires the estimation of an additional parameter, however, so fitting it is less desirable if the number of observations is small.

The HNB offers a better fit than the NB when there are excess zeros in the data, although at the cost of additional complexity. Whether this cost is worth paying depends on the number of excess zeros in the data and the amount of observations; if the number of excess zeros or amount of observations is small, better results may be achieved by fitting the simpler NB model.

The Pearson and deviance dispersion statistics provide a quick and easy way to identify the presence of overdispersion in a model, but cannot on their own be used to diagnose whether it is true or merely apparent overdispersion. This can be assessed with a rootogram, which is in general an excellent tool for inspecting the fit of count models, and can also, in at least some cases, give a hint which model family might be most appropriate. It is simple to produce rootograms using the *countreg* package.

5 Bibliography

References

[1] J. M. Hilbe, *Modelling Count Data*, 1st Ed. Cambridge University Press, 2014, p. 283, ISBN: 9781107611252. [Online]. Available: `www.cambridge.org`.

[2] C. Kleiber and A. Zeileis, "Visualizing Count Data Regressions Using Rootograms," *American Statistician*, 2016, ISSN: 15372731. DOI: `10.1080/00031305.2016.1173590`. arXiv: `arXiv:1605.01311v1`.

[3] P. McCullagh and J. A. Nelder, *Generalized Linear Models*. 1989, ISBN: 978-0-412-31760-6. DOI: `10.1007/978-1-4899-3242-6`. arXiv: `arXiv:1011.1669v3`. [Online]. Available: `http://link.springer.com/10.1007/978-1-4899-3242-6`.

[4] J. M. Ver Hoef and P. L. Boveng, "QUASI-POISSON VS. NEGATIVE BINOMIAL REGRESSION: HOW SHOULD WE MODEL OVERDISPERSED COUNT DATA?" *Ecology*, vol. 88, no. 11, pp. 2766–2772, 2007. [Online]. Available: `http://digitalcommons.unl.edu/usdeptcommercepub%20http://digitalcommons.unl.edu/usdeptcommercepub/142`.

[5] A. Zeileis and C. Kleiber, *Count data regression*, 2017. [Online]. Available: `https://r-forge.r-project.org/projects/countreg/`.

6 Appendix

```
# appendix code


# function which returns data simulated from poisson or neg binomial
sim <- function(coefficients, num.observations=50000,
excess.zeros=0, type="poisson")
{
# generate random uniform predictors
num.predictors = length(coefficients)
# create predictor matrix (intercept term)
pred.mat <- matrix(rep(1,num.observations),nrow=num.observations)
# add other predictors to matrix
for(i in 2:num.predictors){
vals <- runif(num.observations)
pred.mat <- cbind(pred.mat, vals)
if(i==2){
# create a data frame to fit model to later
df <- data.frame(x1=vals)
}
else{
# add next predictor to data frame
df[, paste("x",i-1,sep="")] <- vals
}
}

if(type=="poisson"){
# generate observations from poisson
df$y <- rpois(num.observations, exp(pred.mat %*% coefficients))
}
else if(type=="negbin"){
# theta fairly arbitrarily set at 2
theta = 2;
# generate observations from negbin
df$y <- rnegbin(num.observations,
exp(pred.mat %*% coefficients), theta)
}
# if we have an excess zero probability defined, convert some
# values to zeros
# this is not totally equivalent to the two stage described,
# but close enough for our purposes
if(excess.zeros>0 & excess.zeros < 1){
df$y = ifelse(runif(num.observations) < excess.zeros, 0, df$y);
}
return(df);
}

# function which returns both dispersion stats
dispersionStats <- function(model){
prdisp <- sum(residuals(model, type="pearson")^2)/model$df.residual
devdisp <- sum(residuals(model, type="deviance")^2)/model$df.residual
return(list(pearson.dispersion=prdisp,deviance.dispersion=devdisp))
}

#===== DATA SIMULATION ================================
set.seed(12)
data.sim.1 <- sim(c(2,1,-1,0.5,0), 5000, excess.zeros=0, type="poisson")
data.sim.2 <- sim(c(2,1,-1,0.5,0), 5000, excess.zeros=0.1, type="poisson")
data.sim.3 <- sim(c(2,1,-1,0.5,0), 5000, excess.zeros=0, type="negbin")
data.sim.4 <- sim(c(2,1,-1,0.5,0), 5000, excess.zeros=0.1, type="negbin")

#===== POISSON ========================================
library(countreg)

# poisson full fit to poisson data
model.poisson <- glm(y ~ ., family="poisson", data=data.sim.1)
summary(model.poisson)
dispersionStats(model.poisson)
rootogram.true.poisson <- rootogram(model.poisson, plot=FALSE,
max=max(data.sim.1$y))
autoplot(rootogram.true.poisson);
ggsave(file="images/rootogram_true_poisson.png", height=2, width=6.5)

# poisson fit with missing predictors to poisson data
model.poisson.missing.1 <- glm(y ~ x1, family="poisson", data=data.sim.1)
summary(model.poisson.missing.1)
```

```
AIC(model.poisson.missing.1)
dispersionStats(model.poisson.missing.1)
rootogram.poisson.missing.1 <- rootogram(model.poisson.missing.1,
plot=FALSE, max=max(data.sim.1$y))
autoplot(rootogram.poisson.missing.1);
ggsave(file="images/rootogram_poisson_missing_1.png", height=2, width=6.5)

# poisson fit with only noise term to poisson data
model.poisson.noise <- glm(y ~ x4, family="poisson", data=data.sim.1)
summary(model.poisson.noise)
dispersionStats(model.poisson.noise)
rootogram.poisson.noise <- rootogram(model.poisson.noise,
max=max(data.sim.1$y), plot=FALSE)
autoplot(rootogram.poisson.noise);
ggsave(file="images/rootogram_poisson_noise.png", height=2, width=6.5)

# poisson full fit to poisson data with excess zeros
model.poisson.excess.zeros <- glm(y ~ ., family="poisson", data=data.sim.2)
summary(model.poisson.excess.zeros)
dispersionStats(model.poisson.excess.zeros)
rootogram.poisson.excess.zeros <- rootogram(model.poisson.excess.zeros,
plot=FALSE, max=max(data.sim.2$y))
autoplot(rootogram.poisson.excess.zeros)
ggsave(file="images/rootograms_excess_zeros.png", height=2, width=6.5)

# poisson full fit to negbin data
model.poisson.negbin <- glm(y ~ ., family="poisson", data=data.sim.3)
summary(model.poisson.negbin)
dispersionStats(model.poisson.negbin)
rootogram.poisson.negbin <- rootogram(model.poisson.negbin,
plot=FALSE, max=max(data.sim.3$y))
autoplot(rootogram.poisson.negbin);
ggsave(file="images/rootogram_poisson_negbin.png", height=2, width=6.5)

# poisson fit to facebook data
model.poisson.fb <- glm(PostLikes ~ ., family="poisson", data=facebook.proc)
summary(model.poisson.fb)
AIC(model.poisson.fb)
dispersionStats(model.poisson.fb)
```

```r
rootogram.poisson.fb <- rootogram(model.poisson.fb, plot=FALSE, max=208)
autoplot(rootogram.poisson.fb);
ggsave(file="images/rootogram_poisson_facebook.png", height=2, width=6.5)

#===== QUASIPOISSON ==========================================

# quasipoisson fit to poisson data
model.quasipoisson <- glm(y ~ .,
family="quasipoisson", data=data.sim.1)
summary(model.quasipoisson)

# quasipoisson fit to negbin data
model.quasipoisson.negbin <- glm(y ~ .,
family="quasipoisson", data=data.sim.3)
summary(model.quasipoisson.negbin)
# note quasipoisson doesn't think x4 is significant, as opposed to poisson

# quasipoisson fit to facebook data
model.quasipoisson.fb <- glm(PostLikes ~ .,
family="quasipoisson", data=facebook.proc)
summary(model.quasipoisson.fb)
# doesn't think PageLikes, Month, Paid are significant

#===== NEG BIN  =====================
library(MASS)
model.nb.sim.1 <- glm.nb(y ~ ., data = data.sim.1)
summary(model.nb.sim.1)
rootogram.nb.sim.1 <- rootogram(model.nb.sim.1,
plot=FALSE, max=max(data.sim.1$y))
autoplot(rootogram.nb.sim.1)
ggsave(file="images/rootogram_nb_sim.png", height=2, width=6.5)

model.nb.sim.2 <- glm.nb(y ~ x1, data = data.sim.1)
summary(model.nb.sim.2)
AIC(model.nb.sim.2)
rootogram.nb.sim.2 <- rootogram(model.nb.sim.2,
plot=FALSE, max=max(data.sim.1$y))
autoplot(rootogram.nb.sim.2)
```

```
model.nb.full <- glm.nb(PostLikes ~ ., data = facebook.proc)
summary(model.nb.full)
AIC(model.nb.full)
rootogram.nb.full <- rootogram(model.nb.full,
plot=FALSE, max=208)
autoplot(rootogram.nb.full)

model.nb.2 <- glm.nb(PostLikes ~ Category + Type + Weekend,
data = facebook.proc)
summary(model.nb.2)
AIC(model.nb.2)
rootogram.nb.2 <- rootogram(model.nb.2, plot=FALSE, max=208)
autoplot(rootogram.nb.2)
ggsave(file="images/rootogram_nb_2.png", height=2, width=6.5)

#===== HURDLE NEG BIN ==============================
model.hurdle.negbin.fb <- hurdle(PostLikes ~ Type + Category + Weekend,
dist="negbin",
data=facebook.proc,
zero.dist="binomial",
link="logit")
summary(model.hurdle.negbin.fb);
AIC(model.hurdle.negbin.fb)

rootogram.hurdle.negbin.fb <- rootogram(model.hurdle.negbin.fb,
plot=FALSE, max=210)
autoplot(rootogram.hurdle.negbin.fb)
ggsave(file="images/rootogram_hurdle_nb_fb.png", height=2, width=6.5)
```