**Section 5.3 – Overdispersion (continued)**

Negative binomial regression models

One way to write the negative binomial distribution is

 for y = 0, 1, … and k > 0

This probability distribution occurs when one is interested in the probability of y failures before the kth success.

Examine a mathematical statistics book if you are interested in more detail about this particular distribution. Also, a more in-depth examination on how the negative binomial distribution comes about for count data is available on p. 553 of Agresti (2013).

For us, there are two important aspects to this distribution. First, the values of Y are non-negative integers just like a Poisson random variable. Second, the distribution can be rewritten as

 for y = 0, 1, …, and k>0

where E(Y) = μ and Var(Y) = μ + μ2/k. Compare the relationship between Var(Y) and E(Y) here with what was used for the quasi-Poisson regression models.

As k0, we have more overdispersion. As k∞, we have E(Y) = Var(Y) = μ; thus, the same mean and variance as in the Poisson case.

We will use this probability distribution in a way different from how it would be used in standard applications (probabiity of number of failures prior to the kth success). For our regression model, k is allowed to be a non-integer value.

Maximum likelihood estimation is used to estimate the regression parameters and k. Therefore, Wald and likelihood-based methods can be used for inferences. Information criteria can be used to choose among models.

Example: Horseshoe crabs and satellites (Horseshoe\_ch5.R, horseshoe.csv)

The glm() function cannot estimate a negative binomial regression model, so we will use the glm.nb() function from the MASS package:

> library(package = MASS)

> mod.fit.nb <- glm.nb(formula = satellite ~ width, data = crab, link = log)

> summary(mod.fit.nb)

Call:

glm.nb(formula = satellite ~ width, data = crab, link = log, init.theta = 0.90456808)

Deviance Residuals:

 Min 1Q Median 3Q Max

-1.7798 -1.4110 -0.2502 0.4770 2.0177

Coefficients:

 Estimate Std. Error z value Pr(>|z|)

(Intercept) -4.05251 1.17143 -3.459 0.000541 \*\*\*

width 0.19207 0.04406 4.360 1.3e-05 \*\*\*

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for Negative Binomial(0.9046) family taken to be 1)

 Null deviance: 213.05 on 172 degrees of freedom

Residual deviance: 195.81 on 171 degrees of freedom

AIC: 757.29

Number of Fisher Scoring iterations: 1

 Theta: 0.905

 Std. Err.: 0.161

 2 x log-likelihood: -751.291

> class(mod.fit.nb)

[1] "negbin" "glm" "lm"

The estimated negative binomial regression model is



R refers to the “dispersion parameter” estimate as  = 0.905, which is 1/. Thus,  = 1/0.905 = 1.1, and the estimated variance of Y is now  (remember that  depends on the value of x). This extra variability then shows up in the estimated variances for  and :

> vcov(mod.fit.nb)

 (Intercept) width

(Intercept) 1.37225388 -0.051449169

width -0.05144917 0.001941126

> vcov(mod.fit.quasi)

 (Intercept) width

(Intercept) 0.93565064 -0.034334475

width -0.03433447 0.001268475

> vcov(mod.fit)

 (Intercept) width

(Intercept) 0.29402590 -0.0107895239

width -0.01078952 0.0003986151

> sum.fit.nb <- summary(mod.fit.nb)

> sum.fit <- summary(mod.fit)

> std.err <- data.frame(Poisson = sum.fit$coefficients[,2],

 quasi = sum.fit.quasi$coefficients[,2], nb = sum.fit.nb$coefficients[,2])

> std.err

 Poisson quasi nb

(Intercept) 0.54224155 0.96729036 1.17143241

width 0.01996535 0.03561565 0.04405821

What affect will this have on inferences using the model?

Remember that as k∞, we have LESS overdispersion. Thus, as θ∞, we have more overdispersion, where θ = 0 corresponds to no overdispersion. Using this information and the standard error for  given in the output, we have an informal way to assess if overdispersion exists. Note that 0.905 – 2×0.161 > 0 so it appears that overdispersion exists (3 standard errors results in same conclusion).

Notes:

* The usual functions can be used for LRTs and for confidence intervals with respect to making inferences about regression parameters.
* Confidence intervals for μ can be found through predict(). For example, using the ci.mu() function from Chapter 4 (see Horseshoe.R), below are the intervals for width = 23 and 33 from the three regression models:

> ci.mu <- function(newdata, mod.fit.obj, alpha) {

 lin.pred.hat <- predict(object = mod.fit.obj, newdata

 = newdata, type = "link", se = TRUE)

 lower <- exp(lin.pred.hat$fit - qnorm(1-

 alpha/2)\*lin.pred.hat$se)

 upper <- exp(lin.pred.hat$fit + qnorm(1-

 alpha/2)\*lin.pred.hat$se)

 list(lower = lower, upper = upper)

 }

> nb23 <- ci.mu(newdata = data.frame(width = 23), mod.fit.obj = mod.fit.nb, alpha = 0.05)

> quasi23 <- ci.mu(newdata = data.frame(width = 23), mod.fit.obj = mod.fit.quasi, alpha = 0.05)

> Pois23 <- ci.mu(newdata = data.frame(width = 23), mod.fit.obj = mod.fit, alpha = 0.05)

> data.frame(type = c("Negative binomial", "Quasi-Poisson", "Poisson"),

 lower = round(c(nb23$lower, quasi23$lower, Pois23$lower),2),

 upper = round(c(nb23$upper, quasi23$upper,

 Pois23$upper),2))

 type lower upper

1 Negative binomial 1.01 2.05

2 Quasi-Poisson 1.16 2.21

3 Poisson 1.33 1.92

> nb33 <- ci.mu(newdata = data.frame(width = 33),

 mod.fit.obj = mod.fit.nb, alpha = 0.05)

> quasi33 <- ci.mu(newdata = data.frame(width = 33),

 mod.fit.obj = mod.fit.quasi, alpha = 0.05)

> Pois33 <- ci.mu(newdata = data.frame(width = 33),

 mod.fit.obj = mod.fit, alpha = 0.05)

> data.frame(type = c("Negative binomial", "Quasi-

 Poisson", "Poisson"),

 lower = round(c(nb33$lower, quasi33$lower,

 Pois33$lower),2),

 upper = round(c(nb33$upper, quasi33$upper,

 Pois33$upper),2))

 type lower upper

1 Negative binomial 5.45 17.73

2 Quasi-Poisson 5.29 12.82

3 Poisson 6.43 10.56

The intervals for the negative binomial model are shorter for the smaller width and longer for the larger width values when compared to the quasi-Poisson model. This occurs due to how the two models take into account the overdispersion (see equations for Var(Y)).

* There are much fewer extreme residuals with this model than with the Poisson model:



Choosing between the options to handle overdispersion

If it is clear why there is extra variability, take it into account! For example, include an additional explanatory variable. Also, if the data is observed in clusters, such as at time = 1, 2, and 3 an observation is observed for the same individual, then use a Poisson generalized linear mixed model.

If it is not clear why there is extra variability, the quasi-Poisson and negative binomial models provide viable options. To choose between the two, you should focus on the differences in how the models account for overdispersion:

Quasi-Poisson: Var(Y) = γμ

Negative binomial: Var(Y) = μ + μ2/k

Ver Hoef and Boveng (2007) suggest to examine the relationship between the mean and variance by plotting the squared residuals vs. the estimated mean from the Poisson regression model fit. If there is an increasing quadratic trend in the points, the negative binomial model is preferred. If there is mostly a linear trend, the quasi-Poisson is preferred.

Example: Horseshoe crabs and satellites (Horseshoe\_ch5.R, horseshoe.csv)

> set1 <- data.frame(res.sq = residuals(object = mod.fit,

 type = "response")^2, mu.hat = mod.fit$fitted.values)

> fit.lin <- lm(formula = res.sq ~ mu.hat, data = set1)

> fit.quad <- lm(formula = res.sq ~ mu.hat + I(mu.hat^2),

 data = set1)

> summary(fit.quad)

Call:

lm(formula = res.sq ~ mu.hat + I(mu.hat^2), data = set1)

Residuals:

 Min 1Q Median 3Q Max

-12.409 -6.134 -2.673 -1.072 120.892

Coefficients:

 Estimate Std. Error t value Pr(>|t|)

(Intercept) -3.1635 7.1585 -0.442 0.659

mu.hat 6.1627 3.9398 1.564 0.120

I(mu.hat^2) -0.6097 0.4913 -1.241 0.216

Residual standard error: 16.4 on 170 degrees of freedom

Multiple R-squared: 0.0188, Adjusted R-squared: 0.007258

F-statistic: 1.629 on 2 and 170 DF, p-value: 0.1992

> plot(x = set1$mu.hat, y = set1$res.sq, xlab = expression(hat(pi)), ylab = "Squared Residual")

> curve(expr = predict(object = fit.lin, newdata =

data.frame(mu.hat = x), type = "response"), col =

"blue", add = TRUE, lty = "solid")

> curve(expr = predict(object = fit.quad, newdata =

 data.frame(mu.hat = x), type = "response"), col = "red", add = TRUE, lty = "dashed")

> legend(x = 6, y = 100, legend = c("Quadratic", "Linear"), col = c("red", "blue"), lty = c("dashed", "solid"), bty = "n")



At least with respect to the regression model, the quadratic term is non-significant. There may be one point at about  = 9 that is distorting a quadratic trend. If I had to choose here, I would likely go with the quasi-Poisson regression model (as long as there were no other explanatory variables available to explain the overdispersion).

Ver Hoef and Boveng (2007) also suggest to group the data first to help see trend better in the plot. My program shows how to do this. The overall conclusions are not different from here.

Overdispersion for other types of responses

A similar problem can occur with binomial or multinomial responses as well. For example, consider a setting when binomial observations are observed each with the same number of trials n. If the observed variance of the binomial observations is greater than the variance as given by the binomial distribution (nπ(1 – π) for each set of trials), then this a violation of the binomial assumptions. Simply, if n > 1 and ALL binomial observations were 0 and n, overdispersion would occur.

When overdispersion occurs, the options outlined previously hold for a binomial response too, except a quasi-binomial model replaces the quasi-Poisson model and the beta-binomial regression model replaces the negative binomial regression model. The glm() function with family = quasibinomial(link = "logit") estimates the quasi-binomial model. The vglm() function of the VGAM package can estimate these models using a betabinomial() argument value for family.

Overdispersion cannot be measured when there are Bernoulli responses because these responses are only 0 or 1.

The MGLM package includes methods to handle overdispersion for multinomial data.