Overdispersion in Generalized Linear Models

ANTH 3720-001 Archaeological and Forensic Science Lab Methods: Data Analysis with R

Elic Weitzel

Jan. 29-31, 2021

If you would like the original R Markdown file, find it on my GitHub page at https://github.com/weitzele/Basic-R-Tutorial.

1 Overdispersion

With Poisson and Binomial generalized linear models, we have an extra consideration called overdispersion. Overdispersion is a phenomenon that can sometimes occur with Poisson and binomial family GLMs because both Poisson and binomial probability distributions assume a very specific relationship between the mean and the variance: in this case, that the variance increases with the mean. Other distributions don't necessarily make this assumption, so for example, we don't need to worry about overdispersion for log-normal GLMs - only Poisson and binomial. When this assumption of a one-to-one mean-variance relationship isn't quite met, we can get overdispersion or underdispersion, but the former is more of an issue.

Overdispersion and underdispersion have the unfortunate consequence of making our p-values inaccurate. They won't really impact our ability to predict values (i.e. our model fit), but they do impact our significance tests. Overdispersion is really bad because it makes us more likely to find a false positive: a significant result when we shouldn't find one. Underdispersion is a bit better simply because it leads to a more conservative significance test. It makes it more likely that you'll find a false negative, or a p-value that isn't significant when it should be.

But neither option is really a good thing, so we should control for these if we can.

1.1 Quasi-Family GLMs

If your model is overdispersed, you could find a different distribution that better captures what's going on with your data. Perhaps a negative binomial or beta distribution would work, but we're not covering those in this course as they're more complicated. Alternatively, you could add additional predictor variables. But if those approaches aren't feasible - and they often aren't - you can use a *quasi-family* model.

A quasi-family refers to a version of your specified family of residual distribution that tries to accommodate the fact that your mean-variance relationship isn't quite right. It's basically a correction applied to the residual deviance, but don't worry about the details. It has some unfortunate implications if you're doing higher-level modeling work, but it's still a legitimate course of action. It's simply something you might have to do if your residual deviance and residual degrees of freedom aren't 1:1.

1.2 Checking for Overdispersion

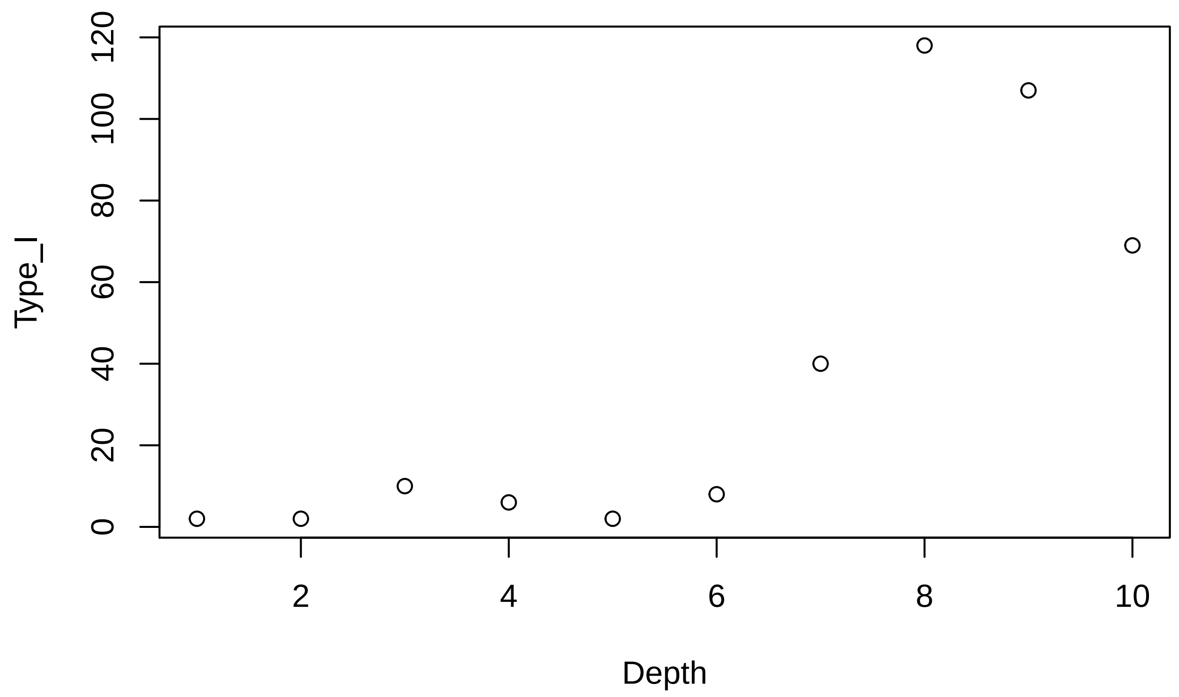
You could find code online that will check to see if your model is overdispersed. This is generally done by comparing the residual deviance of your model to the residual degrees of freedom in a hypothesis test and getting a p-value. However, I find that the easiest way is to simply apply the solution - a quasi-family distribution - and see if it changes things.

Let's load the `archdata` package again and built a quick GLM using the Nelson dataset on pottery types from a pueblo site in New Mexico. We'll model the relationship between Type I ceramic counts and depth.

```r
library(archdata)

data("Nelson")

#plot the counts of Type I pottery by depth
plot(Type_I ~ Depth, data = Nelson)
```



Now let's build our model, using a Poisson family error distribution. Remember that because a log link function is the default option for a Poisson distribution in the `glm` function, we don't actually need to specify `family = poisson(link = "log")`. We can simply write `family = poisson` and the rest is implied.

```r
#build a poisson-family glm of type I pottery counts by depth
nels.mod <- glm(Type_I ~ Depth, data = Nelson, family = poisson)
```

If we inspect the summary output from this GLM, we'll see that in the middle of the output, there's a line in parentheses which says `(Dispersion parameter for poisson family taken to be 1)`. This dispersion parameter is what we care about when investigating overdispersion. This parameter is essentially the ratio of the residual deviance to the residual degrees of freedom. It's assumed to be 1 in Poisson and binomial family GLMs because the mean and variance are the same in these types of models. These distributions have only a single parameter value that defines them, unlike a normal distribution that has two: mean and standard deviation. So if we've got overdispersion, the dispersion parameter it will be greater than 1 and if we've got underdispersion, it will be less than 1.

```r
summary(nels.mod)
```

```
##
## Call:
## glm(formula = Type_I ~ Depth, family = poisson, data = Nelson)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -5.581  -3.175  -0.944   1.030   7.399
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)   0.6591     0.2203   2.992  0.00277 **
## Depth         0.4179     0.0260  16.074  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 497.58  on 9  degrees of freedom
## Residual deviance: 131.17  on 8  degrees of freedom
## AIC: 179.49
##
## Number of Fisher Scoring iterations: 5
```

Our summary output tells us that our model dispersion parameter is *taken* to be 1. This means that our model is simply assuming this paramter is 1, even if it's really not. So to check and see what this parameter really is, we can use a quasipoisson family instead and inspect that model output.

So, let's change our model from a poisson family GLM to a quasipoisson family GLM and inspect the dispersion parameter in the summary output.

```r
nels.mod <- glm(Type_I ~ Depth, data = Nelson, family = quasipoisson)

#pull out just the dispersion parameter from the summary output
summary(nels.mod)$dispersion
```

```
## [1] 16.70674
```

Our model summary tells us that our dispersion parameter for this quasipoisson family model is taken to be 16.70674. That is definitely overdispersed!

A reasonable range for this dispersion parameter is between 0.9 and 1.1. If you're a bit beyond that, it's probably okay - these are not hard cutoff points necessarily. But a value of more than 16 is huge - we are dealing with very substantial overdispersion here.

Because our dispersion paramter is so large, we ought to keep our model in this quasi-family distribution. If it were more reasonable, we could proceed with just a regular Poisson distribution, but a parameter of 16 is pretty large so we ought to go with a quasipoisson distribution.

1.3 P-values from Quasi-Family Models

As stated above, the reason we care about overdispersion and underdispersion is because they can mess up our p-values. But using a quasi-family model also impacts our calculation of p-values. To see this for yourself, try to run the `lrtest()` function from the `lmtest` package, as we did to run likelihood ratio tests for other GLMs.

```r
library(lmtest)

lrtest(nels.mod)
```

```
## Likelihood ratio test
##
## Model 1: Type_I ~ Depth
## Model 2: Type_I ~ 1
##   #Df LogLik Df Chisq Pr(>Chisq)
## 1   2
## 2   1        -1
```

It won't calculate a p-value! This is because the `lrtest()` function is not meant to work for quasi-family GLMs.

To run a likelihood ratio test on this model, we actually need to use the `anova()` function. This function is used to conduct an analysis of variance on Gaussian linear models, but it can also be used to conduct a likelihood ratio test on GLMs. When this is done, it's called an analysis of deviance because deviance, not variance, is what we care about with GLMs.

To run this likelihood ratio test, we simply feed our model object into the `anova()` function, but add the `test` argument set to `"F"`.

```r
anova(nels.mod, test = "F")
```

```
## Analysis of Deviance Table
##
## Model: quasipoisson, link: log
##
## Response: Type_I
##
## Terms added sequentially (first to last)
##
##
##       Df Deviance Resid. Df Resid. Dev      F   Pr(>F)
## NULL                      9     497.58
## Depth  1   366.41         8     131.17 21.932 0.001575 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

And there we have our p-value! In this case the LRT tells us that our model is significantly better than the null, intercept-only model (F = 21.93, df = 1, $p < 0.01$).

The reason that we added the `test = "F"` argument is that we had to tell the `anova()` function to use an F test to conduct the LRT. The usual test statistic for an LRT, if you'll recall from previous ones we've run, is a X^2 test. We won't really get into the different reasons for this here, but I will say that a X^2 distribution can be used when variance is known and an F distribution is for when variance is estimated. Since quasi-family models estimate the deviance in a weird way, we can no longer use a X^2 distribution. LRTs can only be based on a X^2 distribution for binomial and poisson models for which deviance is better understood, not quasi-binomial or quasi-poisson models for which deviance is estimated. For quasi- distributions, we need an F test.

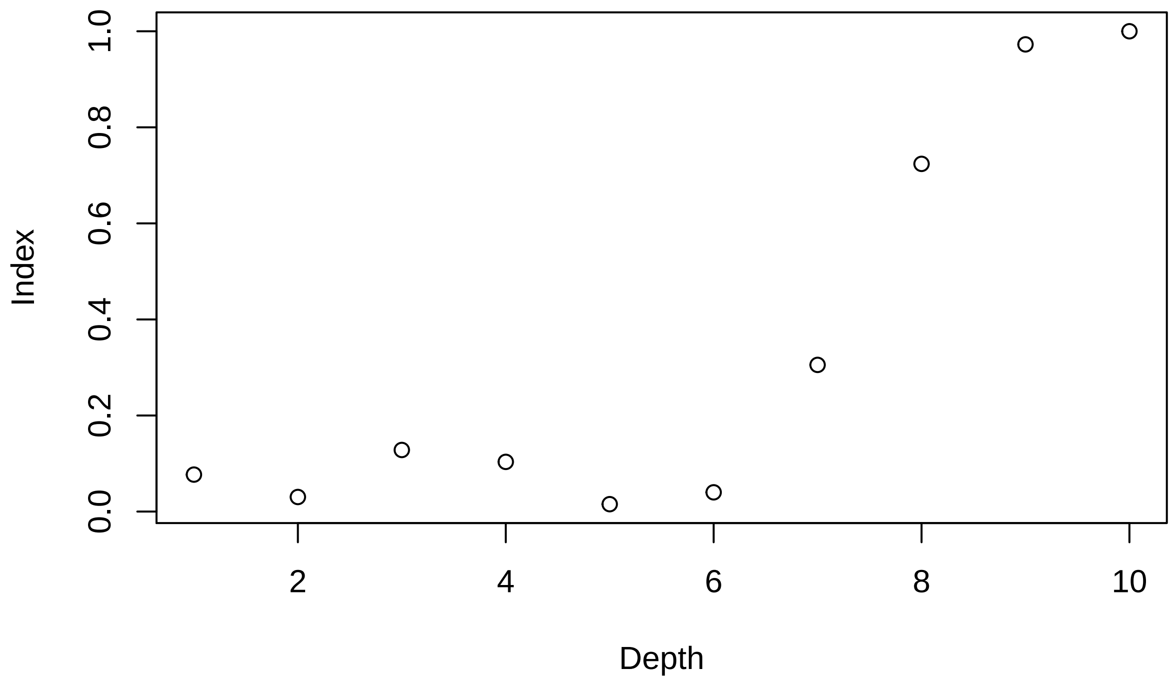
2 Underdispersion

Let's explore a case where we have underdispersion. This time, we'll use a binomial GLM instead of a Poisson.

First, we'll create an index of the relative abundance of Type I to Type II Red ceramics by depth. This index will range between 0 and 1 because it's essentially describing the proportion of Type I and Type II Red ceramics that are just Type I.

```r
#create index
Nelson$Index <- with(Nelson, Type_I/(Type_I + Type_II_Red))

#plot the index values
plot(Index ~ Depth, data = Nelson)
```



We can see that our data are proportions, so they range between 0 and 1 and generally increase with values of depth. There's not much variation and the relationship is pretty tight.

Let's model this relationship using a binomial family model.

```r
nels.mod2 <- glm(Index ~ Depth, data = Nelson, family = binomial)
```

```
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
```

Note that we got an warning message saying that there were "non-integer successes in a binomial glm!" This is because we're using a binomial distribution to model proportions even though a binomial distribution is technically a discrete distribution. However, binomial GLMs are commonly used for proportion data, so this warning message won't actually stop us from continuing or impact our model in any way.

Now if we were to inspect the model summary, we'd see that the *assumption* is once again a dispersion paramter of 1.

```r
summary(nels.mod2)$dispersion
```

```
## [1] 1
```

But this is only an assumption. We need to check this by seeing if this value changes substantially if we specify a quasibinomial family instead of binomial

```r
nels.mod2 <- glm(Index ~ Depth, data = Nelson, family = quasibinomial)

summary(nels.mod2)$dispersion
```

```
## [1] 0.2764664
```

Our dispersion parameter is actually 0.276. That's much lower than the 0.9 cutoff point we mentioned above, so we've definitely got underdispersion.

Now the safest bet would be to just switch to a quasi-family GLM and proceed as normal. But if you read up on underdispersion, you might sometimes opt to keep your GLM with a regular binomial family error distribution. Underdispersion is often less common than overdispersion, and it also leads to more conservative statistical results - we'd be less likely to find a false positive, unlike with overdispersion which makes us more likely to find a false positive. The latter is a worse possibility.

But our model's dispersion paramter is pretty low, so it's probably a safer bet to just switch to a quasibinomial family GLM.

Now to calculate a p-value for this model, remember once again that we can't use the `lrtest()` function because it always assumes a X^2 test. With quasi-family GLMs, we need our likelihood ratio test to use an F test. So we can use the `anova()` function and specify `test = "F"`.

```r
anova(nels.mod2, test = "F")
```

```
## Analysis of Deviance Table
##
## Model: quasibinomial, link: logit
##
## Response: Index
##
## Terms added sequentially (first to last)
##
##
```

```
##       Df Deviance Resid. Df Resid. Dev      F   Pr(>F)
## NULL                      9     7.4165
## Depth  1   5.8261         8     1.5903 21.074 0.001777 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

And there we have it - a significant p-value. Our model is significant ($F = 21.07$, $df = 1$, $p < 0.01$).