Poisson and negative binomial regression in R

author: Maria Paniw date: 11.02.2016 width: 1620 height: 1080

What is the Poisson distribution?

Just like the binomial distribution, the Poisson distribution deals with counts. Unlike the binomial distribution however, there are no 0 (failure) events. We just observe λ successes in time or space, e.g., number of animals per $1km^2$ (abundance).

Thus, the Poisson distribution is a one parameter distribution defined as:

$$Y \sim \mathcal{P}(\lambda)$$

The two important properties of the Poisson distribution are:

$$E(Y) = \mu = \lambda$$

$$Var(Y) = \mu = \lambda$$

Likelihood function

Let's assume you have n observations Y_1, Y_2, Y_3,..., Y_i. Each of those can be treated as realizations of independent Poisson random variables with $\mu_i = f(\beta_0 + \beta_1 x_i)$.

Then, the likelihood for each Y_i can be expressed as:

$$P(Y = y_i|\beta_0, \beta_1) = \frac{e^{-\mu_i}\mu_i^{y_i}}{y_i!}$$

Taking the natural log of the right side of the equation and ignoring the denominator, we get the loglikelihood of the model as:

$$logL(\beta) = \sum_{i=1}^{n} y_i log(\mu_i) - \mu_i$$

The log-linear Poisson model

In the log-linear Poisson model, we are not interested in the probability as we were in logistic regression. Instead, we are interested in the μ or λ. This makes it easy for us because the Poisson regression is pretty close to the linear model:

$$\mu_i = f(\beta_0 + \beta_1 x_i)$$

However, if μ_i was a linear fucntion of the predictor x, we would run into a problem. The linear predictor on the right can assume any value, but the Poisson mean has to be non-negative.

Therefore, we assume for the Poisson model that:

$$\mu_i = exp^{(\beta_0 + beta_1 x_i)}$$

To make the model above linear, we need a link function - which is simply the log.

$$g(\mu_i) = log(\mu_i) = \beta_0 + \beta_1 x_i$$

The back-transformation is simply

$$exp(g(\mu_i))$$

Back to glm in R

The Poisson model can be fitted in R using the `glm` function. The formula is identical to the one in logistic regression with the exception of the "family" argument

```
#load the data
data=read.table("G:/Teaching/StatCourse/Presentations/Droso_abund.txt",
              header=T)

#Fit a Poisson model

modP=glm(abundance~d_forest+humanImpac,data=data,family=poisson)
```

Check the model output

```
summary(modP)
```

```
Call:
glm(formula = abundance ~ d_forest + humanImpac, family = poisson,
    data = data)

Deviance Residuals:
   Min       1Q   Median       3Q      Max
-46.23   -26.50   -15.70    16.72    69.83

Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept)  6.3218783  0.0252788  250.09   <2e-16 ***
d_forest     0.1925572  0.0026028   73.98   <2e-16 ***
humanImpac  -0.0446458  0.0007598  -58.76   <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: 52960  on 42  degrees of freedom
Residual deviance: 40053  on 40  degrees of freedom
AIC: 40370

Number of Fisher Scoring iterations: 6
```

Goodness of fit

```
#The residual deviance given by summary(modP) can also be obtained as

dev=2*sum(data$abundance*log(data$abundance/modP$fitted.values)
          -(data$abundance-modP$fitted.values))
dev
```

```
[1] 40053.1
```

We can calculate the relative deviance just as we did for logistic regression

Overdispersion

Remember that the two key assumptions of the Poisson distribution are:

$$E(Y) = \mu = \lambda$$

$$Var(Y) = \mu = \lambda$$

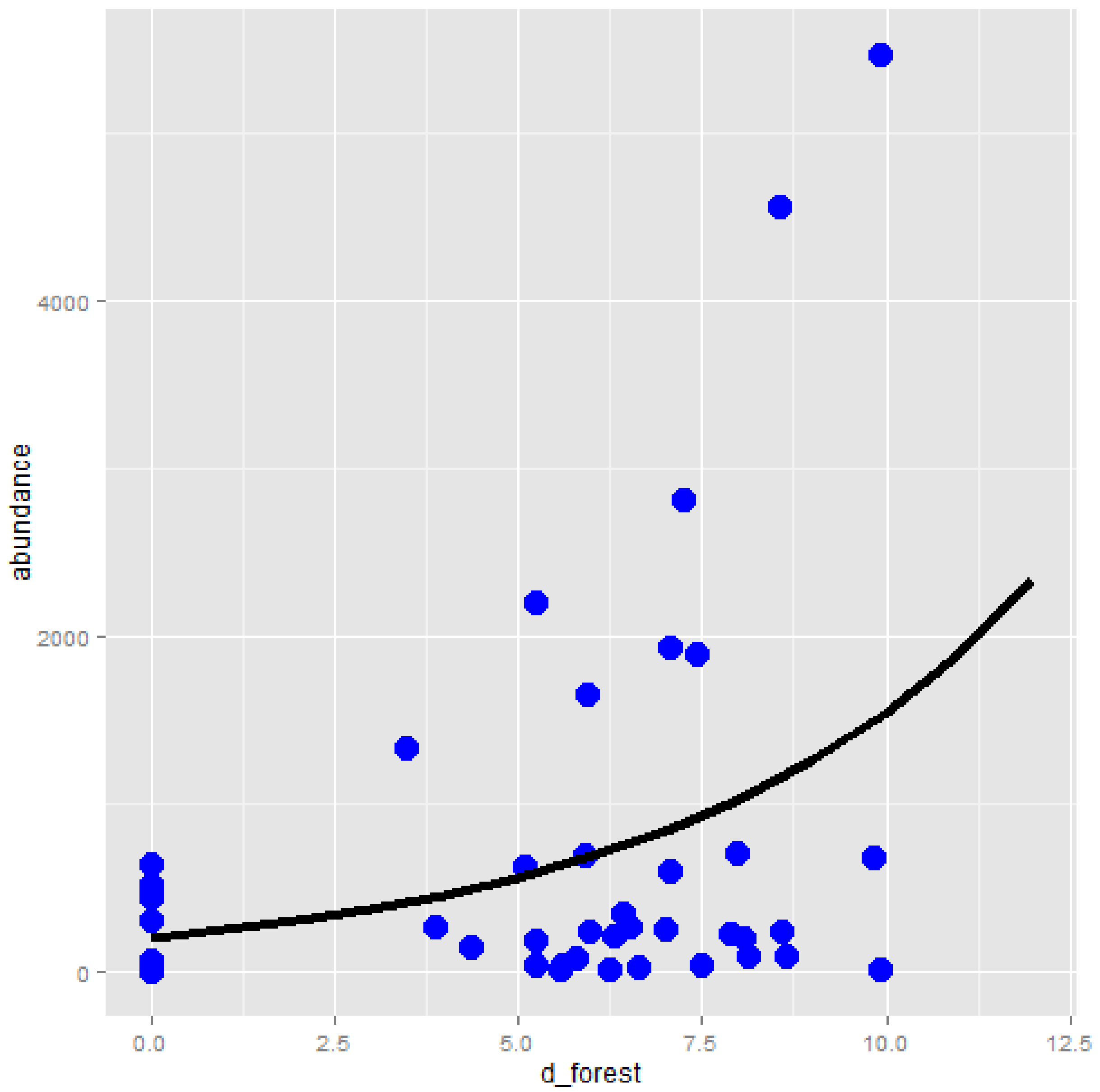
Now, let's plot the predictions of our model vs. the actual values

```
#To facilitate things, fit another model, where abundance is just a function
# of d_forest

modP2=glm(abundance~d_forest,data=data,family=poisson)

#use the predict function to get predictions for our model
# Attention! If you want the values backtransformed and not as logs -
# use the type = "response" argument
predictions=predict(modP2,newdata=data.frame(d_forest=seq(0,12)),
                    type="response")
#put the predictions into a data frame
data.p=data.frame(x=seq(0,12),y=predictions)
```

Plot

```
library(ggplot2)
```

```
ggplot(data,aes(x=d_forest,y=abundance))+xlim(0,12)+
  geom_point(col="blue",size=5)+
  geom_line(data=data.p,aes(x=x,y=y),size=1.5)
```



As we can see from the plot and as is the case for many empirical data sets, **the variance is larger than the mean, i.e., we have overdispersion**.

Detecting overdispersion

There is a simple test for overdispersion available in the package `AER`. This test simply test the following hypotheses:

$$H_0 : Var(Y) = \mu$$

$$H_a : Var(Y) = \mu + c * f(\mu)$$

So, we are pretty much testing $H_0 : c = 0$ vs. $H_a : c \neq 0$

```
#Install and load AES
library(AER)

dispersiontest(modP2,trafo=1)
```

```

    Overdispersion test

data:  modP2
z = 3.4773, p-value = 0.0002533
alternative hypothesis: true alpha is greater than 0
sample estimates:
   alpha
1164.535
```

```
#go to the help for the function to understand the arguments and output
```

Fixing overdispersion

The most common way to address overdispersion in Poisson models is to use the overdispersion parameter,ϕ, as a weight, w, on the mean, where $w = \frac{\mu}{\phi}$.

In this case, we can fit a `glm` with a `quasipoisson` link family:

```
modP2A=glm(abundance~d_forest,data=data,family=quasipoisson)
summary(modP2A)
```

```

Call:
glm(formula = abundance ~ d_forest, family = quasipoisson, data = data)

Deviance Residuals:
   Min       1Q   Median       3Q      Max
-54.29   -29.97   -17.75    15.79    77.94

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  5.31564    0.61862   8.593 1.03e-10 ***
d_forest     0.20305    0.08524   2.382   0.0219 *

Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for quasipoisson family taken to be 1222.436)
```

```
    Null deviance: 52960  on 42  degrees of freedom
Residual deviance: 44404  on 41  degrees of freedom
AIC: NA

Number of Fisher Scoring iterations: 6
```

If you need the likelihood, the quasi families may be problematice because they do not use maximum likelihood to estimate parameters. Instead, they use *iteratively weighted least square*.

Negative binomial regression

In addition to not having a likelihood, the quasi-Poisson distribution assumes that the overdispersed variance, given by the weights, is proportional to the mean. In reality, very small means should get little weight and larger means a larger weight.

A useful distribution that provides a more flexible treatment of the weights is the *negative binomial distribution*. The negative binomial distribution is pretty much the Poisson multiplied by a random effect,ϕ. This random effect is assumed to have a Gamma distribution with parameters α and β.

Negative binomial regression in R

In R, there is a function called `glm.nb` in the package `MASS` to do negative binomial regression

```r
#install and load MASS
library(MASS)
modNB=glm.nb(abundance~d_forest,data=data)
summary(modNB)
```

```
Call:
glm.nb(formula = abundance ~ d_forest, data = data, init.theta = 0.5940928444,
    link = log)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.2023  -1.3281  -0.6939   0.5087   1.4791

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  5.55148    0.39484  14.060  < 2e-16 ***
d_forest     0.16536    0.06379   2.592  0.00953 **

Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for Negative Binomial(0.5941) family taken to be 1)

    Null deviance: 59.854  on 42  degrees of freedom
Residual deviance: 52.998  on 41  degrees of freedom
AIC: 636.13
```

```
Number of Fisher Scoring iterations: 1

              Theta:  0.594
          Std. Err.:  0.108

 2 x log-likelihood:  -630.133
```

Compare Poisson vs. Negative Binomial

```r
AIC(modP2,modNB)
```

```
      df        AIC
modP2  2 44719.3015
modNB  3   636.1331
```

```r
anova(modP2,modNB,test="Chisq")
```

```
Analysis of Deviance Table

Model 1: abundance ~ d_forest
Model 2: abundance ~ d_forest
  Resid. Df Resid. Dev Df Deviance Pr(>Chi)
1        41      44404
2        41         53  0    44351
```

```r
#Here is the problem with the likelihood ration test.
# It doesn't recognize the models as nested models even though the NB model
# has one parameter more.

#A safe way to go if you like the LR test, is comapring the deviance to
# a random Chisq distribution with 1 df.
```

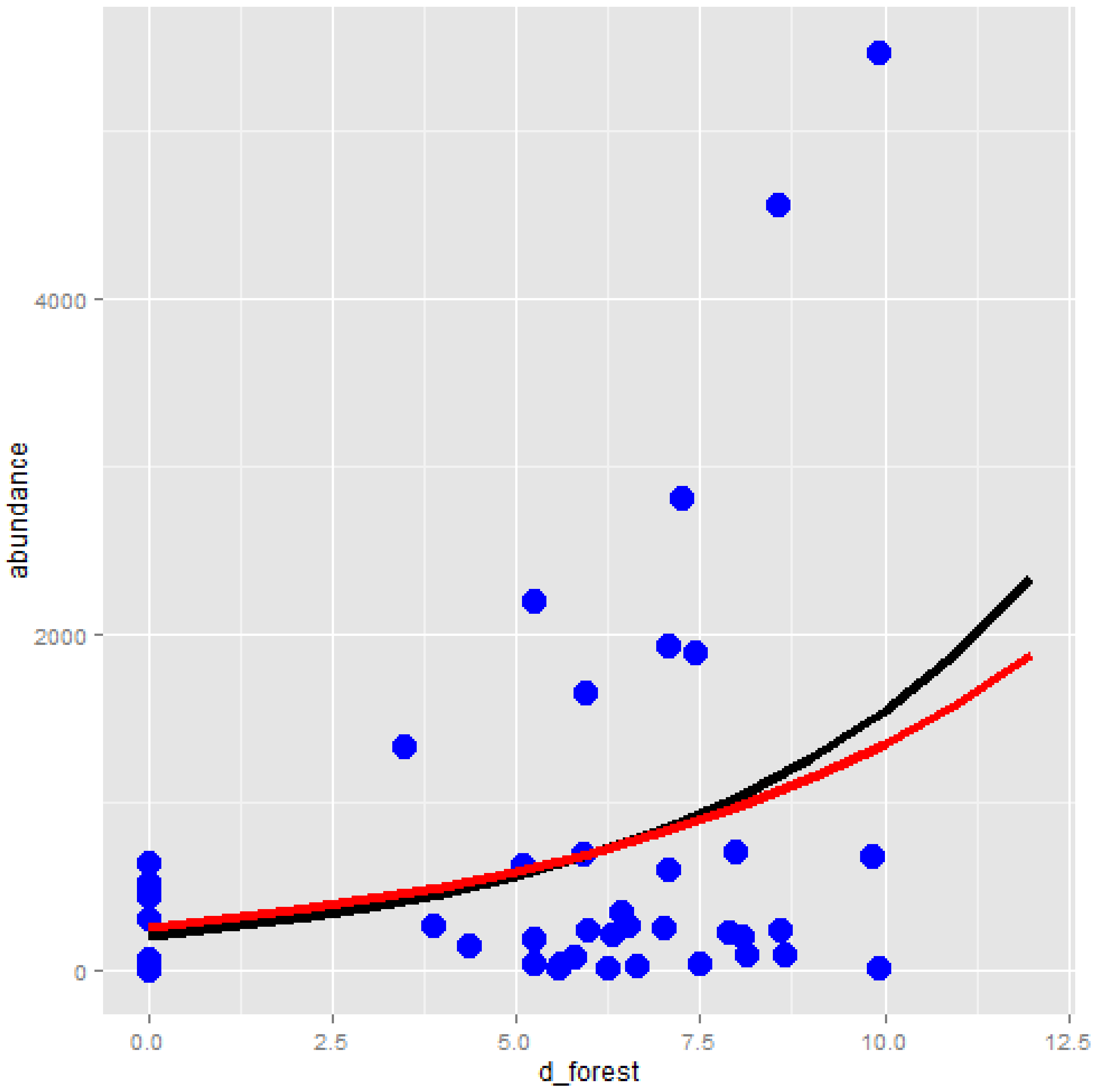
Plot predictions

Plotting our new predictions to the existing ones:

```r
predNB=predict(modNB,newdata=data.frame(d_forest=seq(0,12)),
                    type="response")
#put the predictions into a data frame
data.pNB=data.frame(x=seq(0,12),y=predNB)
```

Plot

```r
ggplot(data,aes(x=d_forest,y=abundance))+xlim(0,12)+
  geom_point(col="blue",size=5)+
  geom_line(data=data.p,aes(x=x,y=y),size=1.5)+
  geom_line(data=data.pNB,aes(x=x,y=y),size=1.5,col="red")
```



Confidence intervals

Because we are fitting generalized linear models, the parameters associated with the linear predictor are assumed to be normally distributed. That means that the 95% confidence interval of any value X_i can be obtained as:

$$upperbound : g(X_i) + 1.96 * SE(g(X_i))$$
$$lowerbound : g(X_i) - 1.96 * SE(g(X_i))$$

Manual calculation of the confidence intervals requires a bit of matrix algebra, so I won't go into detail, but if you are interested, check out this link.

http://www.ats.ucla.edu/stat/r/faq/deltamethod.htm

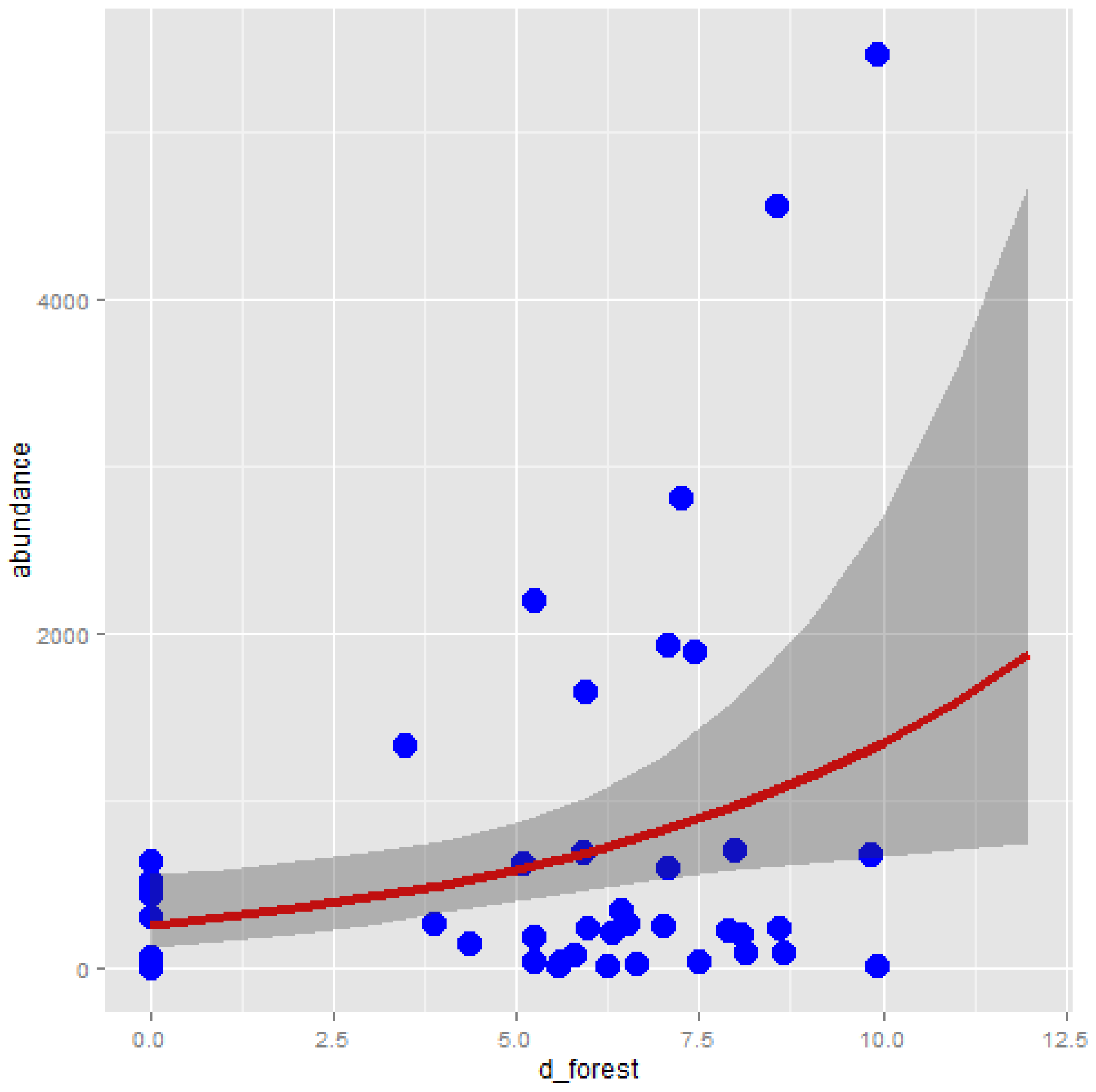
Confidence intervals in R

R coputes the standard errors of the estimates easily for all values of X with the `predict` function:

```r
#You have to specify se.fit
predNB=predict(modNB,newdata=data.frame(d_forest=seq(0,12)),
                    se.fit=T)
#put the predictions into a data frame with BACKTRANSFORMED CI

data.pNB=data.frame(d_forest=seq(0,12),abundance=exp(predNB$fit),
                    LL=exp(predNB$fit-1.96*predNB$se.fit),
                    UL=exp(predNB$fit+1.96*predNB$se.fit))
```

Plot with CI

```r
ggplot(data,aes(x=d_forest,y=abundance))+xlim(0,12)+
  geom_point(col="blue",size=5)+
  geom_line(data=data.pNB,aes(x=d_forest,y=abundance),size=1.5,col="red")+
  geom_ribbon(data=data.pNB,aes(ymin = LL, ymax = UL), alpha = 0.3)
```



There are other types of CI you can calculate: http://www.ats.ucla.edu/stat/r/dae/logit.htm

Overdispersion binomial data

```
# We will work with an example in the package faraway
#(download it if you don't have it).
# Here, we want to know how pesticide use affects the beetles on corn plants.

library(faraway)

data(beetle)
#The data contain the total number of beetles exposed to pesticides (exposed)
```

```r
# of different concentration (conc)
# and how many of the exposed got affected (affected)

str(beetle)
```

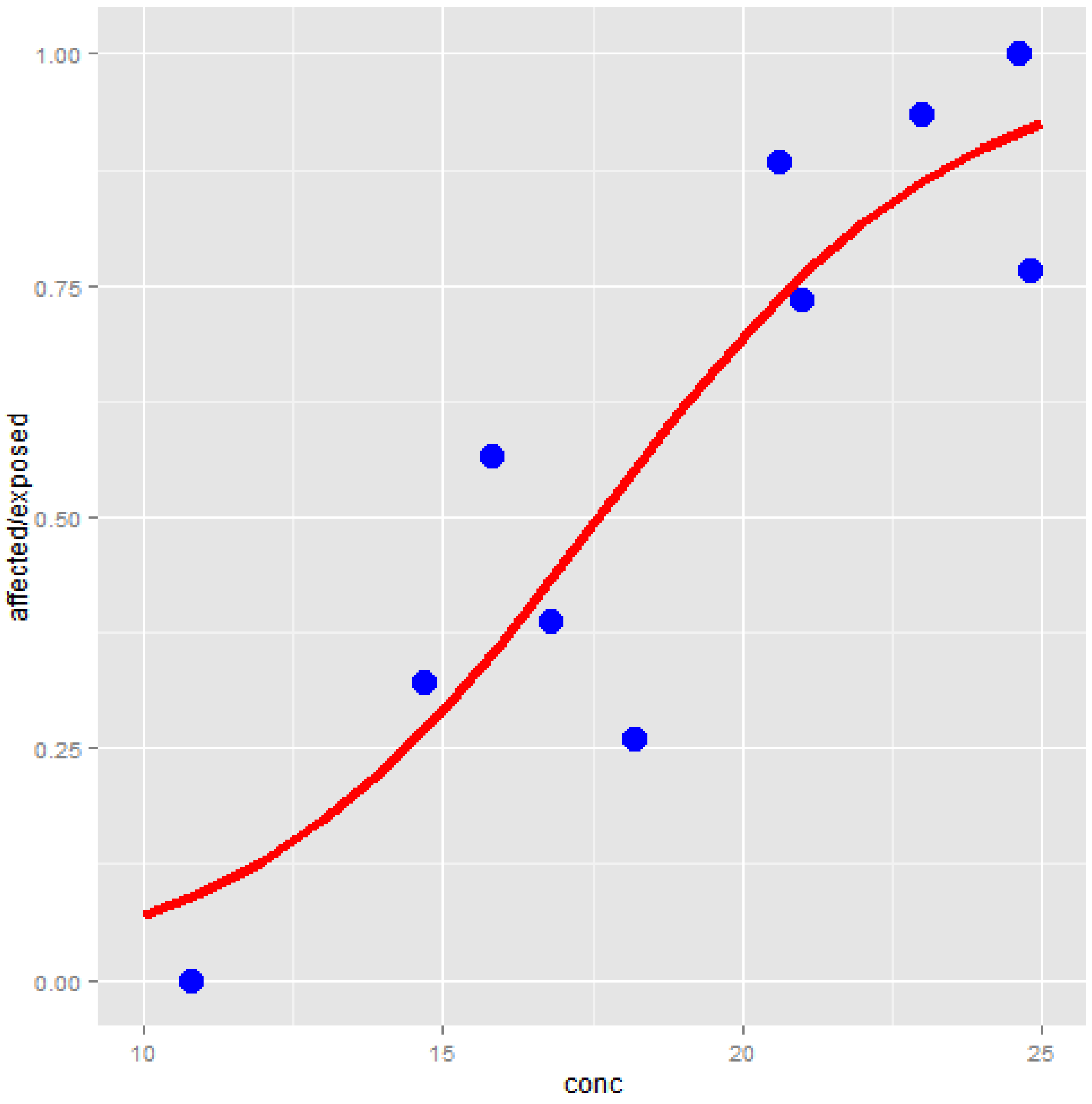
```
'data.frame':	10 obs. of  3 variables:
 $ conc    : num  24.8 24.6 23 21 20.6 18.2 16.8 15.8 14.7 10.8
 $ affected: num  23 30 29 22 23 7 12 17 10 0
 $ exposed : num  30 30 31 30 26 27 31 30 31 24
```

```r
affected=beetle$affected #positive response 1

not.affected=beetle$exposed-beetle$affected # 0

response=cbind(affected,not.affected) # bind the two into a matrix

# Fit the model
model_binom <- glm(response ~ conc, family = binomial,data=beetle)

#Your deviance should be approximately equal to you df
```

Plot predictions

```r
#predict the probability of being affected
pred=data.frame(conc=seq(10,25),
                 affected=predict(model_binom,
                           newdata=data.frame(conc=seq(10,25)),
                           type="response"))
```

```r
ggplot(beetle,aes(x=conc,y=affected/exposed))+xlim(10,25)+
  geom_point(col="blue",size=5)+
  geom_line(data=pred,aes(x=conc,y=affected),size=1.5,col="red")
```



What does dispersion mean in binomial regression?

In a binomial regression that deal with proportions, like the case of the beetle data, where each observation i (row) is considered an independent random binomial variable with n trials and p successes, the mean and variance are expressed as:

$$\mu_i = n_i p_i$$

and

$$\sigma_i^2 = \frac{\mu_i(n_i - \mu_i)}{n_i}$$

Overdispersion means that the variance of y_i is larger than the expected variance σ_i^2.

If overdispersion is present in a dataset, the estimated standard errors and test statistics the overall goodness-of-fit will be distorted and adjustments must be made.

Note, there is **no overdispersion for ungrouped data** (0-1 counts). Overdispersion is not possible if $n_i = 1$.

Adjusting for overdispersion

Use the `quasibinomial` link in logistic regression

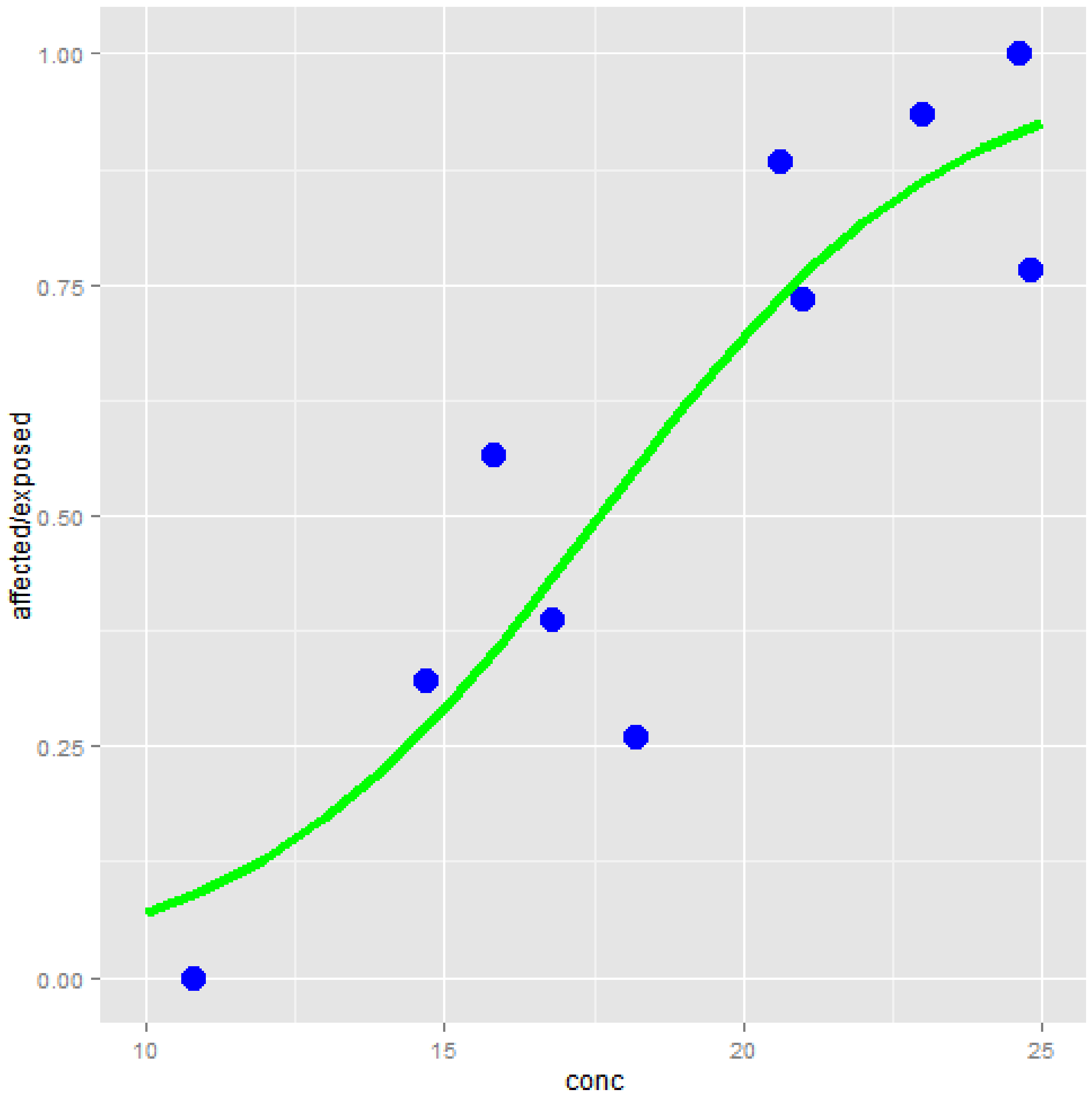
```
model_overdispersed <- glm(response ~ conc,
                           family=quasibinomial(), data=beetle)

# The Chisq test simply tests whether your quasibinomial model perform better
pchisq(summary(model_overdispersed)$dispersion * model_binom$df.residual,
       model_binom$df.residual, lower = F)
```

```
[1] 2.207846e-05
```

Plot the results of the quasibinomial model

```
#first, make the predictions

pred.qb=data.frame(conc=seq(10,25),
                 affected=predict(model_overdispersed,
                           newdata=data.frame(conc=seq(10,25)),
                           type="response"))
```

```
ggplot(beetle,aes(x=conc,y=affected/exposed))+xlim(10,25)+
  geom_point(col="blue",size=5)+
  geom_line(data=pred,aes(x=conc,y=affected),size=1.5,col="red")+
  geom_line(data=pred.qb,aes(x=conc,y=affected),size=1.5,col="green")
```

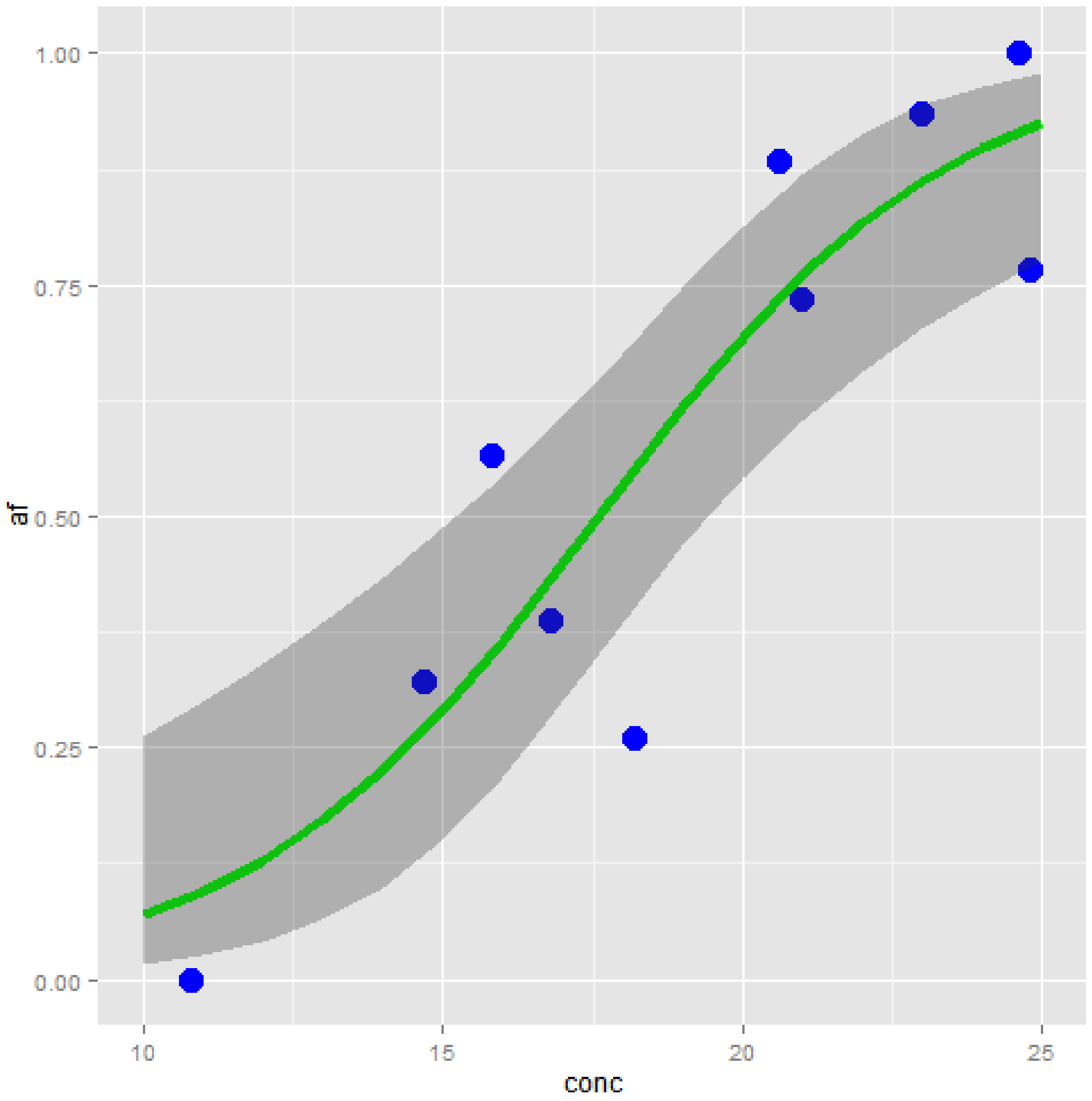


Plot the confidence intervals

```
#You have to specify se.fit
pred.pq=predict(model_overdispersed,newdata=data.frame(conc=seq(10,25)),
                   se.fit=T)
#put the predictions into a data frame with BACKTRANSFORMED CI
LL=exp(pred.pq$fit-1.96*pred.pq$se.fit)/(1+exp(pred.pq$fit-1.96*pred.pq$se.fit))
UL=exp(pred.pq$fit+1.96*pred.pq$se.fit)/(1+exp(pred.pq$fit+1.96*pred.pq$se.fit))
data.pq=data.frame(conc=seq(10,25),
                  af=exp(pred.pq$fit)/(1+exp(pred.pq$fit)),
                   LL=LL,
```

```r
                    UL=UL)
# All y should have the same name for geom_ribbon to work.
beetle$af=beetle$affected/beetle$exposed
```

```r
ggplot(beetle,aes(x=conc,y=af))+xlim(10,25)+
  geom_point(col="blue",size=5)+
  geom_line(data=data.pq,aes(x=conc,y=af),size=1.5,col="green")+
  geom_ribbon(data=data.pq,aes(ymin = LL, ymax = UL), alpha = 0.3)
```


af
conc