Lecture 5 Logistic, Poisson and Negative Binomial Regressions

Ahmed Nadeem

2024-10-09

We're going to simulate some predictor variables for each logistic regression, poisson regression and negative bionomial regression, and then analyze each.

Logistic Regression

- We're going to have one response variable and two predictor variables (one which is yes or no, the other which is continuous).

QUESTION: What are you imagining your response variable to be? What are your two predictor variables?

- I am imagining a forced choice task (the most common psychophysics experiment methodology), where someone is trying to discern if the first stimuli or second stimuli is faster

```
N=50
predictor_cont<-rnorm(N,0,0.2)
predictor_binary<-rbinom(N,1,0.5)

y<-1.44*predictor_cont+2.5*predictor_binary+rnorm(N, 0, 0.01) ### Remember that to make this do-able in a linear framework, like this, we're assuming this is on a log scale!
exp(1.44)
```

```
## [1] 4.220696
```

```
### We're now going to draw a probability distribution from y, assuming that y is on a logit scale
y_prob<-plogis(y)
y_prob
```

```
##  [1] 0.4297075 0.9280403 0.4879707 0.9395580 0.9366252 0.3732710 0.9383962
##  [8] 0.3362835 0.8909783 0.4429920 0.8949733 0.9400513 0.9339490 0.3683425
## [15] 0.5694267 0.9270567 0.9301856 0.9300962 0.4893657 0.4459459 0.5395909
## [22] 0.5221986 0.3950844 0.9240946 0.9175359 0.9059089 0.9041947 0.4639260
## [29] 0.4332127 0.9215526 0.9285852 0.4812979 0.9389923 0.6097789 0.9396123
## [36] 0.9248008 0.9460888 0.3515584 0.6348080 0.5995490 0.4904547 0.4950127
## [43] 0.4218379 0.4662894 0.9425567 0.4976073 0.4108515 0.4503441 0.4820954
## [50] 0.4871322
```

```
### we can now draw from the probability distribution (so that all of our responses are 0,1)
```

```r
y_dummy = rbinom(n = N, size = 1, prob = y_prob)


data<-cbind.data.frame(1:N, y_prob, y, predictor_cont, predictor_binary)

log_reg_model<-glm(y_dummy~predictor_binary+predictor_cont, data=data,
family='binomial')

### if you have loaded and installed lmerTest, it will give you p-values in
your summary, if not, it won't
summary(log_reg_model)
```

```
##
## Call:
## glm(formula = y_dummy ~ predictor_binary + predictor_cont, family =
"binomial",
##     data = data)
##
## Coefficients:
##                   Estimate Std. Error z value Pr(>|z|)
## (Intercept)         0.1098     0.4120   0.267    0.790
## predictor_binary   19.4177  2281.7231   0.009    0.993
## predictor_cont      1.3304     1.8682   0.712    0.476
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 59.295  on 49  degrees of freedom
## Residual deviance: 38.296  on 47  degrees of freedom
## AIC: 44.296
##
## Number of Fisher Scoring iterations: 18
```

```r
### From here, you need to back transform to get the estimates on the y
scale, instead of the log(y) scale

exp(coef(log_reg_model)) ### these are on the 'regular' scale, not the log
scale
```

```
##      (Intercept) predictor_binary   predictor_cont
##     1.116103e+00     2.710272e+08     3.782536e+00
```

```r
exp(confint(log_reg_model))
```

```
## Waiting for profiling to be done...

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred##
Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

##                         2.5 %     97.5 %
## (Intercept)      4.987303e-01   2.585791
```

```
## predictor_binary 7.525318e-60         NA
## predictor_cont   1.026819e-01 196.800595
```

```
pR2(log_reg_model) ### r2CU (Nagelkerke R2 is the most comparable to our typical R2)
```

```
## fitting null model for pseudo-r2
```

```
##          llh     llhNull          G2    McFadden        r2ML        r2CU
## -19.1478947 -29.6476659  20.9995423   0.3541517   0.3429472   0.4937824
```

###Question, tell me about the distribution of y_prob?! - It would be a distribution between 0-1 and mostly focus on 1, because my predictor_cont has a low SD

Question: What did you expect the effect sizes of the predictors to be, based on what you simulated (hint, you will need to transform from line 25)?

- The effect sizes would be 1.44 and 2.5, but they wont be those exact values as a tiny bit of noise was added

Question: The coefficient for predictor_cont is 1.44 This means that the expected log OR for a one-unit increase in predictor_cont is 1.44. This means that the OR (biological scale) for a one unit increase in predictor_cont is 4.220696. The R2 for this model is 0.2516362. This means that 25.1% of the variance can be explained by the model.

Let's simulate some Poisson data!

Poisson Regression

This is mostly borrowed from Simulate Simulate Simulate 3!

Question: What is your imagined response variable here? What is your imagined predictor?

- Count of correct responses in the stimuli presented to a participant ### Question: What happens if you add more than one predictor (i.e. x2)?
- Then I am modelling 2 different effects on one outcome, as opposed to one

```
x1 = rnorm(1000,1,0.2) ### here there is one predictor, this a good place to imagine some biology

### Question,

lambda = exp(0.5 + 2.5*x1) ### this is where I change the effect sizes for each predictor
exp(2.5)
```

```
## [1] 12.18249
```

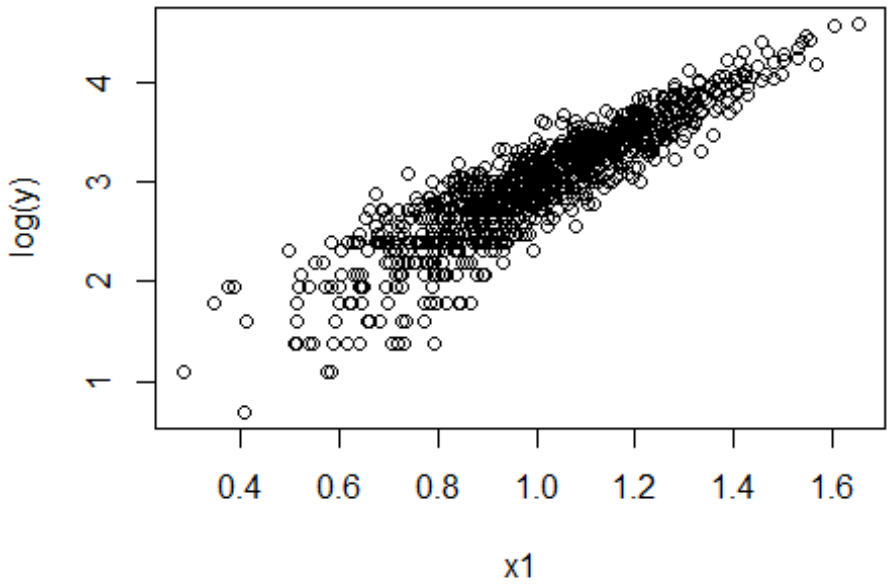
```r
###remember that this means that log(lambda) is a linear model!!
### here I'm simulating three betas against my three predictor variables to make my lambda
### The step above simulates the mean of each value of the response variable. lambda values are continuous, not discrete counts.
## λ is the unobserved true mean of the Poisson distribution for the t-th observation

head(lambda)
```

```
## [1] 33.61513 30.51258 30.88292 13.24118 18.61517 14.56936
```

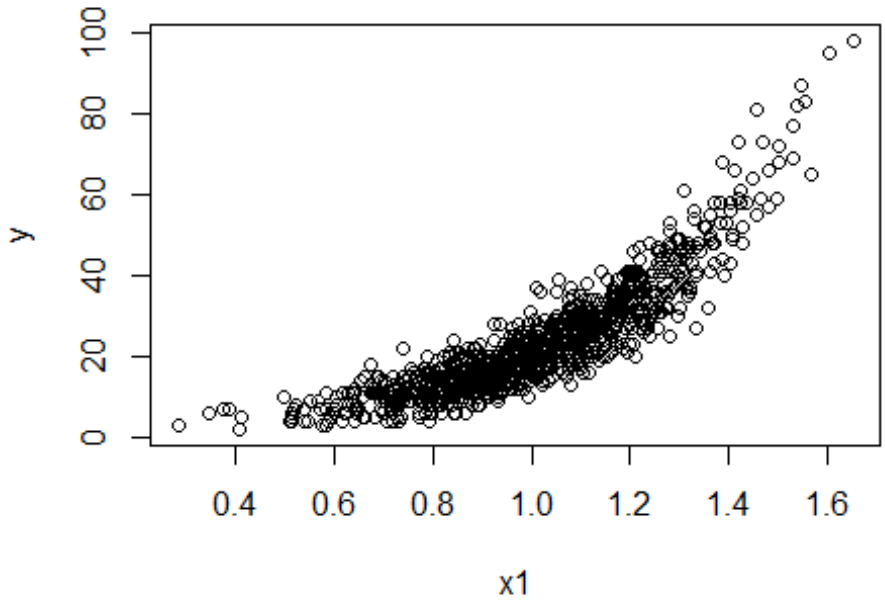
```r
y = rpois(1000, lambda = lambda) ### error is added in this step, remembering that the mean and the variance are the same!
### remember that y is count data!
head(y)
```

```
## [1] 41 35 39 14 17 13
```

```r
plot(x1, log(y)) ### note the residuals - are they normally distributed? Is that okay?
```




```r
plot(x1,y)
```




```
data_poisson<-cbind.data.frame(y, x1)

poisson_model<-glm(y ~ x1, family="poisson", data=data_poisson)
summary(poisson_model)

##
## Call:
## glm(formula = y ~ x1, family = "poisson", data = data_poisson)
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.55256    0.03582   15.43   <2e-16 ***
## x1           2.45012    0.03176   77.13   <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: 6979.7  on 999  degrees of freedom
## Residual deviance: 1005.2  on 998  degrees of freedom
## AIC: 5836.7
##
## Number of Fisher Scoring iterations: 4

###back transform the coefficients
```

```r
exp(coef(poisson_model)) ### these are on the 'regular' scale, not the log scale
```

```
## (Intercept)          x1
##     1.73770    11.58969
```

```r
exp(confint(poisson_model))
```

```
## Waiting for profiling to be done...
```

```
##                 2.5 %    97.5 %
## (Intercept)  1.619762  1.863905
## x1          10.890202 12.334215
```

```r
pR2(poisson_model) ### r2CU (Nagelkerke R2 is the most comparable to our typical R2)
```

```
## fitting null model for pseudo-r2
```

```
##           llh       llhNull            G2      McFadden          r2ML
## -2916.3667909 -5903.6236704  5974.5137590     0.5060039     0.9974573
##          r2CU
##     0.9974647
```

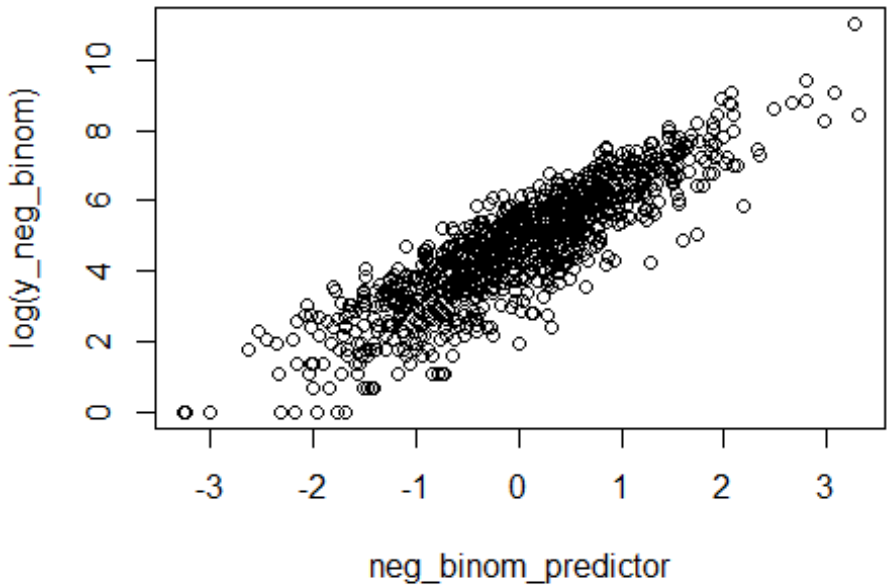
Question: The coefficient for x1 is 2.5 (FILL ME IN!) This means that the expected log count for a one-unit increase in x1 is 2.5. This means that the count difference (biological scale) for a one unit increase in x1 is 12.18249. The R2 for this model is 0.9956404. This means that the model explains 99.56% of the variation.

```r
library(MASS)

neg_binom_predictor <- rnorm(1000, mean = 0, sd = 1) ### what is your imagined predictor?
# - my imagined predictor would be arm length

# Log link function: log(mu) = beta0 + beta1 * X
mu <- exp(5+1.5 * neg_binom_predictor) ### notice that this is ALSO on a log scale
### mu is the expected mean for each observation
exp(1.5)
```

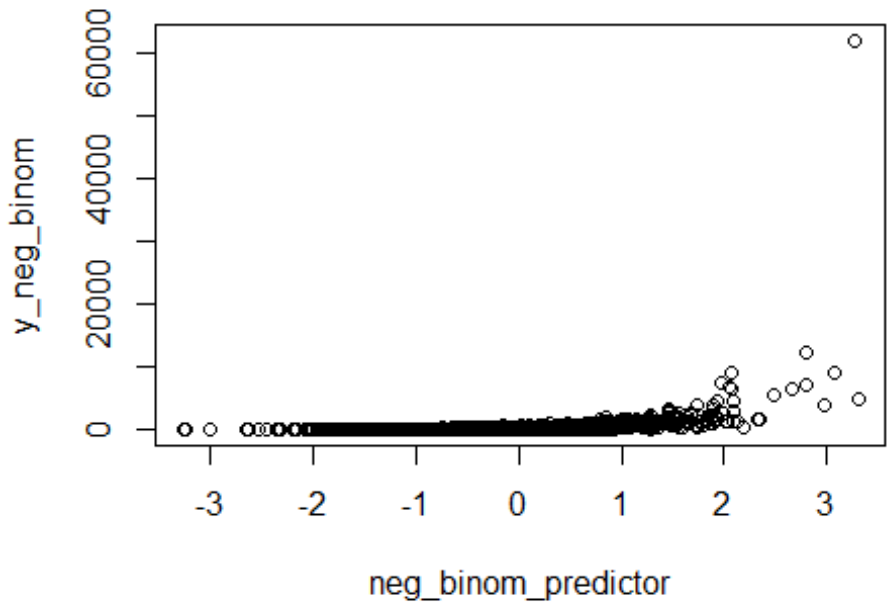
```
## [1] 4.481689
```

```r
# Dispersion parameter for the negative binomial distribution
theta <- 2 ### this makes the difference between negative binomial and poisson. A very large theta would lead the negative binomial to approximate a poisson
##Play around with theta and see what happens!
```

```r
y_neg_binom <- rnegbin(1000, mu = mu, theta = theta)

# Create a data frame to store the simulated data
data_negbiom <- cbind.data.frame(y_neg_binom, neg_binom_predictor)

plot(neg_binom_predictor, log(y_neg_binom))
```




```r
plot(neg_binom_predictor, y_neg_binom)
```




```
negative_binomial_model<-glm.nb(y_neg_binom ~ neg_binom_predictor,
data=data_negbiom)
summary(negative_binomial_model)

##
## Call:
## glm.nb(formula = y_neg_binom ~ neg_binom_predictor, data = data_negbiom,
##     init.theta = 2.062995895, link = log)
##
## Coefficients:
##                      Estimate Std. Error z value Pr(>|z|)
## (Intercept)          4.99848    0.02241   223.0   <2e-16 ***
## neg_binom_predictor  1.48062    0.02369    62.5   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for Negative Binomial(2.063) family taken to be 1)
##
##     Null deviance: 5556.1  on 999  degrees of freedom
## Residual deviance: 1082.6  on 998  degrees of freedom
## AIC: 11794
##
## Number of Fisher Scoring iterations: 1
##
##
##               Theta:  2.0630
##           Std. Err.:  0.0902
```

```
##
##  2 x log-likelihood:  -11787.5310
```

```
exp(coef(negative_binomial_model)) ### these are on the 'regular' scale, not the log scale
```

```
##         (Intercept) neg_binom_predictor
##           148.188170            4.395671
```

```
exp(confint(negative_binomial_model))
```

```
## Waiting for profiling to be done...
```

```
##                           2.5 %     97.5 %
## (Intercept)          141.860664 154.890923
## neg_binom_predictor    4.196119   4.605886
```

```
pR2(negative_binomial_model)
```

```
## fitting null model for pseudo-r2
```

```
##           llh       llhNull            G2      McFadden          r2ML
## -5893.7652823 -6833.8022956  1880.0740265     0.1375570     0.8474212
##          r2CU
##     0.8474222
```

```
### r2CU (Nagelkerke R2 is the most comparable to our typical R2)
```

Question: The coefficient for neg_binom_predictor is 1.5 (FILL ME IN!) This means that the expected log count for a one-unit increase in neg_binom_predictor is 1.5. This means that the count difference (biological scale) for a one unit increase in neg_binom_predictor is 4.481689. The R2 for this model is 0.8342348 This means that 83.42% of the model explains the variance seen in the data.