ENTMLGY 6707 Entomological Techniques and Data Analysis

R Activity 9: Generalized Linear (Mixed-Effects) Models

1 Introduction

As we have talked about in lecture, data and/or model residuals will often not be normally distributed. For when a response variable is continuous, we have covered some transformations that might help with violations of normality. When we have a response that is binary (e.g., presence/absence or alive/dead) or count data, our first guess would be that the data are non-normal. Presence/absence and count data are both very common in ecological research and are expected to follow a Binomial/Bernoulli distribution and Poisson distribution, respectively.

Logistic regression (used for binary data) and Poisson regression (used for count data) can be referred to as "generalized `linear` `models`," and, fittingly, we will use the `glm()` function in R to fit them. The syntax is exactly the same as for `lm()`, except we need to specify (i) the expected distribution of the response variable and (ii) the link function. We won't cover link functions in detail, but you can think of them somewhat as transformations. The canonical link functions (meaning the most "natural" or "mathematically suitable") are the logit link and log link for Binomial distributions and Poisson distributions, respectively. We will only work with canonical link functions.

Note: we are going to ignore some violations of independence in this week's tutorial analysis (e.g., there are some repeated measures I am sweeping under the rug). Just like there are linear mixed-effects models for when the response variable is continuous and residuals are assumed to be normal, there are generalized linear mixed-effects models for non-normal data, which I cover briefly at the end of this tutorial.

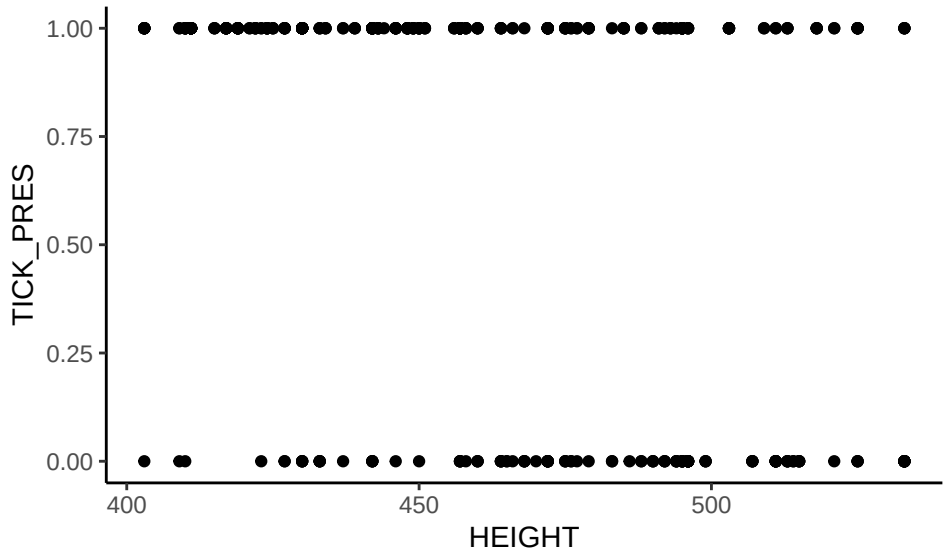
2 Logistic regression

We will analyze the `grouseticks` data from the `lme4` package. The data set report tick abundance, and let's pretend we are interested in modeling occupancy by ticks (presence/absence at a given location, rather than total counts) as a function of `HEIGHT` (= elevation). First, we need to create a new column indicating the presence/absence of ticks at each sampling location (each location sampled corresponds to a single observation or row in our data).

```
library(lme4)
data(grouseticks)
grouseticks$TICK_PRES <- ifelse(grouseticks$TICKS > 0, 1, 0)
```

It can be challenging to graphically assess the relationship when your response is binary, but if you squint, it looks like the points tend to be farther left for the "1" values than the "0" values... meaning that as elevation increased, it appears fewer and fewer sites had ticks.

```
library(tidyverse)
ggplot(data=grouseticks, mapping=aes(x=HEIGHT, y=TICK_PRES))+
        geom_point()+theme_classic()
```



Now let's see if our "guess" based on the graph was correct (statistics are often just needed to tell us if the picture we are looking at is meaningful). The code below indicates we are assuming a binomial error structure (`family=binomial`) and using the canonical link function, the logit link (`(link="logit")`)... hence the name `logistic` regression.

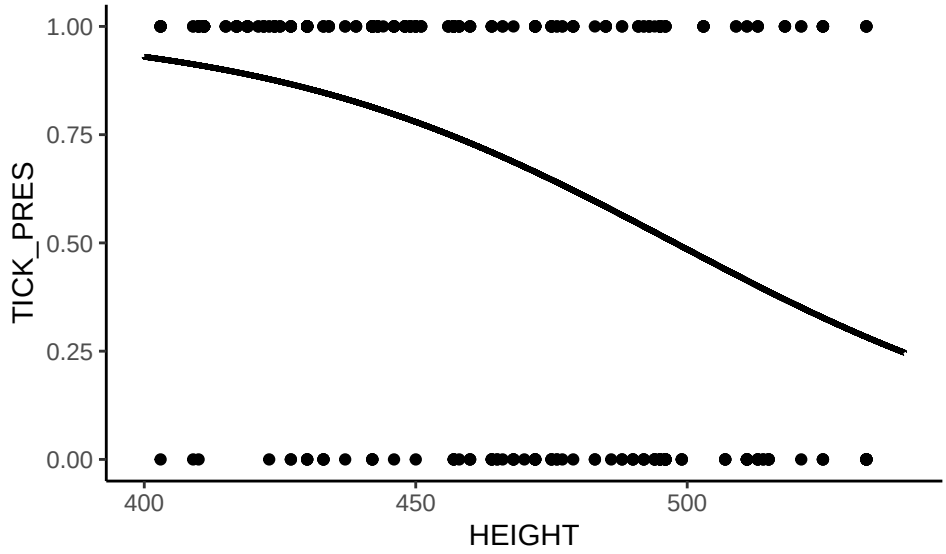
```
fit_tick_logistic_1 <- glm(TICK_PRES~HEIGHT, data=grouseticks,
                           family=binomial(link="logit"))
summary(fit_tick_logistic_1)
```

```
##
## Call:
## glm(formula = TICK_PRES ~ HEIGHT, family = binomial(link = "logit"),
##     data = grouseticks)
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) 13.160275   1.674645   7.859 3.89e-15 ***
## HEIGHT      -0.026439   0.003533  -7.483 7.25e-14 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 500.69  on 402  degrees of freedom
## Residual deviance: 433.84  on 401  degrees of freedom
## AIC: 437.84
##
## Number of Fisher Scoring iterations: 4
```

Now we will add the fit line from our model to the graph. Note the inclusion of the `response` argument below, which tells R to plot the data on the back-transformed (=raw) scale (i.e., R is doing some back-transforming for you under the hood). I mentioned the link function is a transformation of sorts, so the model is actually using the "link transformed" version of the response variable.

```
new_data <- data.frame(HEIGHT = seq(400, 540, 0.001))
new_data$Predicted_TICKS_logistic  <- predict(fit_tick_logistic_1, newdata=new_data,
```

```
                                                type="response")
ggplot(data=grouseticks, mapping=aes(x=HEIGHT, y=TICK_PRES))+
        geom_point()+theme_classic()+
        geom_line(data=new_data, aes(x=HEIGHT, y=Predicted_TICKS_logistic), linewidth=1)
```



2.1 Assumptions

1. Response is binary.
2. The logit-transformed response variable is a linear function of the predictor(s). logit(p) = log(p/(1-p)), were p is the probability of an outcome.
3. Observations are independent.
4. Predictors are not collinear.

2.2 Interpretation

The interpretation of coefficients in a logistic regression differs from simple linear regression. We interpret them as something called odds ratios. We can extract the odds ratio by exponentiating the slope coefficient ($exp(-0.0230647) = 0.977$).

For example, the odds of detecting a tick at a given location are 0.977 the odds (95% CI: 0.967-0.98) of detecting a tick at a location one unit higher in elevation. Said another way, with a one unit increase in elevation, the odds of detecting a tick by decrease by a factor of 0.977 (decreases in odds are generally more difficult to think about compared with increasing odds).

As I've done above, odds ratios are typically reported with a 95% confidence interval, which can be extracted using the `MASS` package:

```
library(MASS)
round(exp(cbind(coef(fit_tick_logistic_1), confint(fit_tick_logistic_1))),3)
```

```
##                            2.5 %      97.5 %
## (Intercept) 519319.747 21441.928 15424690.67
## HEIGHT           0.974     0.967        0.98
```

Additional bits of information often reported with logistic regressions are values of our predictors associated with different predicted probabilities. For example, it might be nice to calculate what level of concentration

of a pesticide is needed to kill 50% of a population of insects.

Our predictions are "transformed" using the logit link, so we would need to back transform those predictions (using the "inverse logit") to get probabilities. Thankfully, the `MASS` package in R has a helpful function for doing just that. For example, we can find the elevation at which we have a 5%, 50%, and 75% chance of finding a tick. The `Dose` is the $HEIGHT$ value associated with the probabilities we specified in the argument `p`.

```
dose.p(fit_tick_logistic_1, p = c(0.05,0.5,0.75))
```

```
##                Dose        SE
## p = 0.05: 609.1162 18.748524
## p = 0.50: 497.7510  5.552538
## p = 0.75: 456.1990  4.964864
```

So, in this study region, the elevation at which we would expect a 50% chance of finding a tick on a grouse was 497.75 $\pm$ 5.55 SE. In practice, I would probably round these values to integers.

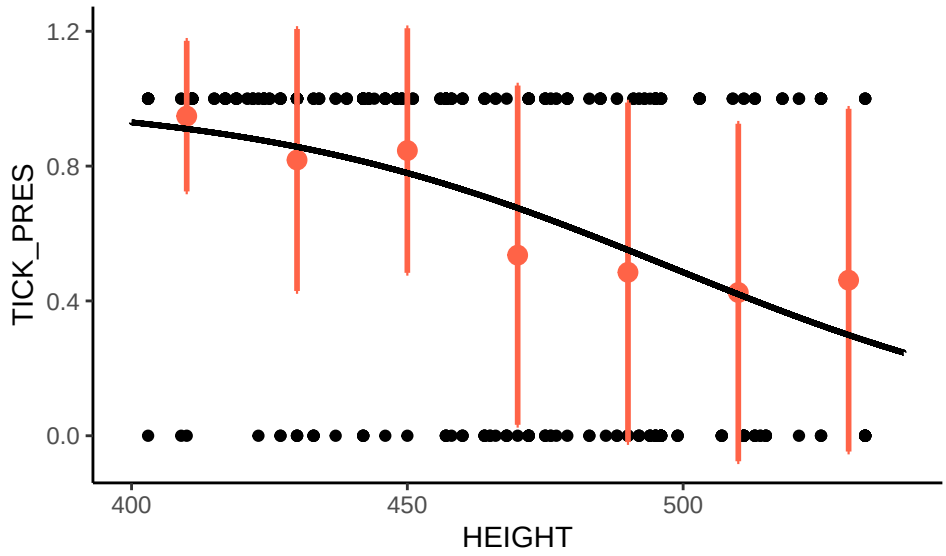
2.3 Checking assumptions and other diagnostics

Some assumptions of logistic regression can be checked using graphical inspection. For example, are there any outliers in the x-direction (e.g., large predictor values that could act as leverage points and "pull" the the fit line up or down).

Another check some folks use is to bin the x-values, calculate the means of those bins/groups, plot them over the raw data, and see if the fit line runs through those means. Below, the `tomato` colored dots and vertical lines are the means and standard deviations of binned `HEIGHT` data.

```
library(plotrix)
grouseticks_new <- grouseticks %>% mutate(new_bin = cut(HEIGHT, breaks=seq(400,540,20)))
bins_for_plot <- grouseticks_new %>% group_by(new_bin) %>%
  summarise(means = mean(TICK_PRES), SDs = sd(TICK_PRES))
bins_for_plot$HEIGHT <- seq(410,530,20) # to ensure we can plot on same graph below
```

```
new_data <- data.frame(HEIGHT = seq(400, 540, 0.001))
new_data$Predicted_TICKS_logistic  <- predict(fit_tick_logistic_1, newdata=new_data,
                                              type="response")
ggplot(data=grouseticks, mapping=aes(x=HEIGHT, y=TICK_PRES))+
        geom_point()+theme_classic()+
    geom_point(data=bins_for_plot, aes(x=HEIGHT, y=means), col="tomato1", size=3)+
  geom_errorbar(data = bins_for_plot,
                mapping = aes(x = HEIGHT, y = means,
                              ymin = means - SDs, ymax = means + SDs),
                linewidth=1, color="tomato1", width=.4) +
        geom_line(data=new_data, aes(x=HEIGHT, y=Predicted_TICKS_logistic), linewidth=1)
```



Another check could be to conduct what is called an analysis of deviance. Think of this as a comparison between models, in this case, a comparison between our fit model and a model with an intercept only (i.e., a flat line).

2.3.1 Analysis of Deviance

Running an `anova()` on models, rather than a traditional analysis of variance, can be used to compare lots of model types (e.g., `lm()`, `glm()`) *when you are analyzing the same exact data and the same exact response variable*. That is, we might use this approach when we want to know if adding an additional predictor is improving the fit. Note that the null hypothesis below is that our two models do an equivalent job of predicting changes in the response variable.

```
anova(fit_tick_logistic_1, test="Chisq")
```

```
## Analysis of Deviance Table
##
## Model: binomial, link: logit
##
## Response: TICK_PRES
##
## Terms added sequentially (first to last)
##
##
##        Df Deviance Resid. Df Resid. Dev Pr(>Chi)
## NULL                     402     500.69
## HEIGHT  1   66.857       401     433.84 2.92e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Below I will actually fit an "intercept-only" model, just to help convince you that that is truly what the `anova()` command above is comparing your model against. Also note that below I input both models into the `anova()` command.

```
intercept_only <- glm(TICK_PRES~1, data=grouseticks,
                      family=binomial(link="logit"))
anova(intercept_only,fit_tick_logistic_1,test="Chisq")
```

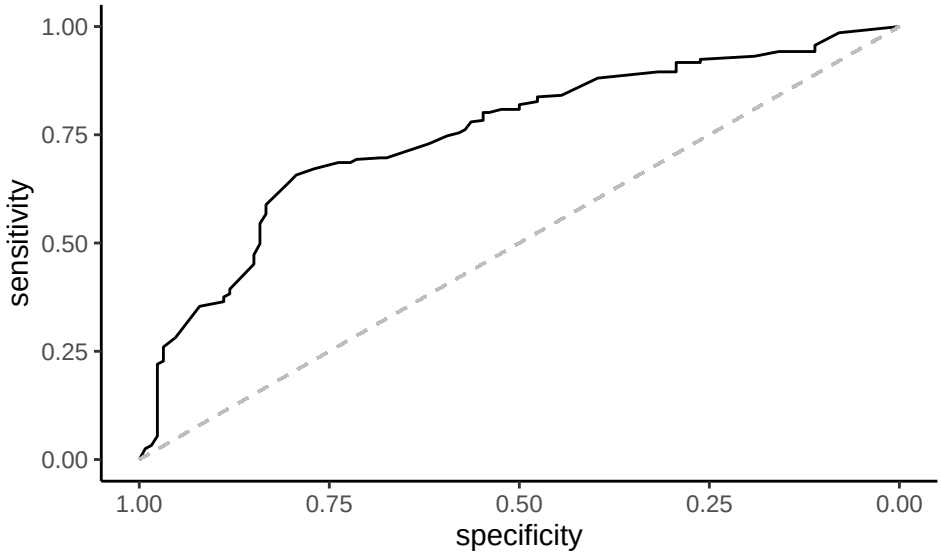
```
## Analysis of Deviance Table
##
## Model 1: TICK_PRES ~ 1
## Model 2: TICK_PRES ~ HEIGHT
##   Resid. Df Resid. Dev Df Deviance Pr(>Chi)
## 1       402     500.69
## 2       401     433.84  1   66.857 2.92e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

2.3.2 ROCs and AUC

A receiver operating characteristic curve (ROC) - which you will hear called a "rock plot" - and the area under such curves (area under the curve, or AUC), are common tools to determine how well a logistic regression predicts outcomes. In short, they estimate how well your model classifies true positives ("sensitivity") and true negatives ("specificity"). Larger AUC values are better and an AUC = 0.50 would mean that your models makes predictions as well as flipping a coin (meaning the model would not be very useful for making predictions).

Making a ROC plot:

```
library(pROC)
predicted <- predict(fit_tick_logistic_1, type="response")
roccurve <- roc(grouseticks$TICK_PRES ~ predicted)
ggroc(roccurve)+theme_classic()+
    geom_segment(aes(x = 1, xend = 0, y = 0, yend = 1), color="grey", linetype="dashed")
```



And then calculating AUC for the above ROC plot:

```
auc(roccurve)
```

```
## Area under the curve: 0.7496
```

3 Poisson Regression

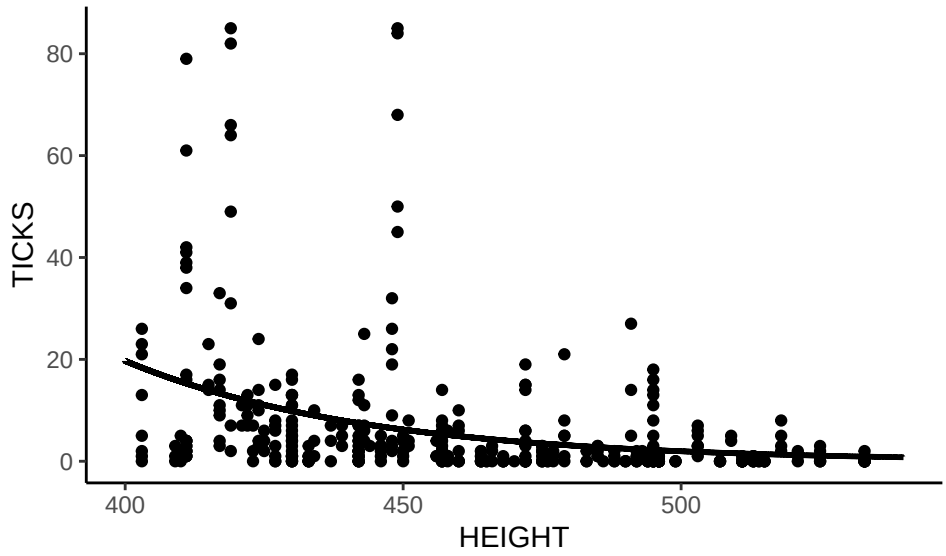
Modeling number of ticks - count data - as a function of height above sea level is a great candidate for Poisson regression. Another way to describe heteroscedasticity is a "mean-variance relationship": as the mean (our predicted values) goes up (or down), does the variance go up (or down)? In simple linear regression, that would be a bad thing. You might remember from class that, for a Poisson distribution, the mean and variance are exactly equal ($= \lambda$). As a result, in Poisson regressions, we expect a mean-variance relationship: the variance *should* increase with increases in our predicted values.

```
fit_tick_poisson_1 <- glm(TICKS~HEIGHT, data=grouseticks, family=poisson(link="log"))
summary(fit_tick_poisson_1)
```

```
##
## Call:
## glm(formula = TICKS ~ HEIGHT, family = poisson(link = "log"),
##     data = grouseticks)
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) 12.2061106  0.3069365   39.77   <2e-16 ***
## HEIGHT      -0.0230647  0.0006999  -32.95   <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: 5847.5  on 402  degrees of freedom
## Residual deviance: 4506.4  on 401  degrees of freedom
## AIC: 5441.4
##
## Number of Fisher Scoring iterations: 6
```

Now we will add the fit line from our model to the graph. Note the inclusion of the `response` argument below, which tells R to plot the data on the back-transformed (=raw) scale (i.e., R is doing some back-transforming for you under the hood). I mentioned the link function is a transformation of sorts, so the model is actually using the "link transformed" version of the response variable.

```
new_data <- data.frame(HEIGHT = seq(400, 540, 0.001))
new_data$Predicted_TICKS_poisson  <- predict(fit_tick_poisson_1,
                                             newdata=new_data, type="response")
ggplot(data=grouseticks, mapping=aes(x=HEIGHT, y=TICKS))+
        geom_point()+theme_classic()+
        geom_line(data=new_data, aes(x=HEIGHT, y=Predicted_TICKS_poisson), linewidth=1)
```



3.1 Assumptions

1. Response variable is count data (Poisson distributed)
2. The log-transformed expected values, $\log(\lambda)$, is a linear function of the predictor(s)
3. Observations are independent
4. There is a mean-variance relationship (λ = mean = variance in a Poisson distribution)

3.2 Interpretation

The interpretation of coefficients in a Poisson regression also differs from simple linear regression. We interpret them as a "multiplicative" factor. That multiplicative factor is found by exponentiating the slope coefficient. For example, the expected number of ticks changes by a multiplicative factor of 0.98 ($= exp(-0.0230647)$), or a 2.3% decrease ($= (1 - 0.977) \times 100$), with an increase in 1 unit of elevation.

3.3 Checking assumptions and other diagnostics

Similar to the logistic regression, we can use a so-called goodness-of-fit test:

```
anova(fit_tick_poisson_1, test="Chisq")
```

```
## Analysis of Deviance Table
##
## Model: poisson, link: log
##
## Response: TICKS
##
## Terms added sequentially (first to last)
##
##
##        Df Deviance Resid. Df Resid. Dev  Pr(>Chi)
## NULL                     402     5847.5
## HEIGHT  1     1341       401     4506.4 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

3.3.1 Overdispersion

You can also check for overdispersion, aka more variation than expected if the data were truly following a Poisson distribution.

```
library(AER)
deviance(fit_tick_poisson_1)/fit_tick_poisson_1$df.residual
```

```
## [1] 11.23798
```

```
dispersiontest(fit_tick_poisson_1)
```

```
##
##  Overdispersion test
##
## data:  fit_tick_poisson_1
## z = 4.2556, p-value = 1.042e-05
## alternative hypothesis: true dispersion is greater than 1
## sample estimates:
## dispersion
##    18.5806
```

When dispersion is a problem (as it appears to be here), you can try using a negative binomial distribution to model the errors instead of a Poisson:

```
library(MASS)
fit_tick_negbin_1 <- glm.nb(TICKS~HEIGHT, data=grouseticks)
summary(fit_tick_negbin_1)
```

```
##
## Call:
## glm.nb(formula = TICKS ~ HEIGHT, data = grouseticks, init.theta = 0.4989859431,
##     link = log)
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) 12.802679   0.997135   12.84   <2e-16 ***
## HEIGHT      -0.024394   0.002168  -11.25   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for Negative Binomial(0.499) family taken to be 1)
##
##     Null deviance: 545.40  on 402  degrees of freedom
## Residual deviance: 428.18  on 401  degrees of freedom
## AIC: 2080.4
##
## Number of Fisher Scoring iterations: 1
##
##
##               Theta:  0.4990
##           Std. Err.:  0.0422
##
##  2 x log-likelihood:  -2074.4070
```

3.3.2 Zero-inflated Poisson

Sometimes having too many 0s in count data can be a problem (called "zero inflation"), so you can fit zero-inflated models. Note I use something called Akaike (Ah-kah-ee-key) information criterion (AIC) values to compare the models below. AIC values are VERY VERY commonly used in model comparisons - including for simple/multiple linear regression, logistic regression, and Poisson regression, among many other linear modeling frameworks. A LOWER AIC indicates the "better" model.

```
library(pscl)
ZI_pois <- zeroinfl(TICKS~HEIGHT, data=grouseticks, dist = "poisson")
summary(ZI_pois)
```

```
##
## Call:
## zeroinfl(formula = TICKS ~ HEIGHT, data = grouseticks, dist = "poisson")
##
## Pearson residuals:
##     Min      1Q  Median      3Q     Max
## -2.6277 -1.1579 -0.6952  0.0865 17.8572
##
## Count model coefficients (poisson with log link):
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)  9.3935360  0.3177692   29.56   <2e-16 ***
## HEIGHT      -0.0161307  0.0007141  -22.59   <2e-16 ***
##
## Zero-inflation model coefficients (binomial with logit link):
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -12.747941   1.864472  -6.837 8.07e-12 ***
## HEIGHT        0.025507   0.003966   6.432 1.26e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Number of iterations in BFGS optimization: 8
## Log-likelihood: -2357 on 4 Df
```

```
AIC(fit_tick_poisson_1,ZI_pois)
```

```
##                     df      AIC
## fit_tick_poisson_1  2 5441.429
## ZI_pois             4 4722.254
```

3.3.3 Zero-inflated Negative Binomial

...and sometimes, if you're really lucky (= sarcasm), your count data might have too many 0s AND overdispersion, so you can fit zero-inflated negative binomial models (note we get some errors...so proceed with caution):

```
ZI_negbin <- zeroinfl(TICKS~HEIGHT, data=grouseticks, dist = "negbin")
summary(ZI_negbin)
```

```
## Warning in sqrt(diag(object$vcov)): NaNs produced
```

```
##
## Call:
## zeroinfl(formula = TICKS ~ HEIGHT, data = grouseticks, dist = "negbin")
##
## Pearson residuals:
```

```
##      Min       1Q   Median       3Q      Max
## -0.71326 -0.61935 -0.42816  0.04451  8.41197
##
## Count model coefficients (negbin with log link):
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) 12.162885   0.942710   12.90   <2e-16 ***
## HEIGHT      -0.022912   0.001981  -11.57   <2e-16 ***
## Log(theta)  -0.645603   0.049286  -13.10   <2e-16 ***
##
## Zero-inflation model coefficients (binomial with logit link):
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -22.72545        NaN     NaN      NaN
## HEIGHT        0.04058        NaN     NaN      NaN
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Theta = 0.5243
## Number of iterations in BFGS optimization: 34
## Log-likelihood: -1037 on 5 Df
```

There also mixed-effects versions of most of these models... but, aside from what I provide below, I'll leave it to you to seek those out when you need 'em.

3.3.4 Fitting an offset variable

Sometimes we have counts that we assume are influenced by our sampling area (e.g., the size of a patch or transect) or effort (e.g., the time we spent looking). In such cases, we need to standardize our counts to adjust for the different sampling intensities. However, depending on how we do that, it could result in non-integer counts. If that happens, the `glm()` command - which expects integers when doing Poisson regressions with a log link - will return warnings.

To illustrate how fitting an offset variable works, we will use the beetle data from class

```
beetles <- read.csv("examples/completely_fabricated_data/beetles.csv")
summary(beetles)
```

```
##       ELB         temperature
##  Min.   :12.00   Min.   : 5.334
##  1st Qu.:14.00   1st Qu.:19.079
##  Median :18.50   Median :24.263
##  Mean   :21.04   Mean   :24.913
##  3rd Qu.:24.25   3rd Qu.:29.701
##  Max.   :46.00   Max.   :42.869
```

We are working with eastern larch beetle traps in this hypothetical example (again, these data are completely made up). When first working with these data, we were modeling beetles per trap. Let's now pretend that we deployed traps in woodlots of varying sizes (measured in hectares) and that each woodlot had one trap. We want to estimate how temperature affects the "number of beetles trapped per woodlot" but our woodlots differ in size AND woodlot size could certainly affect the number of beetles flying around (e.g., larger patch could provide more habitat and host trees for larch beetles).

To illustrate, let's first add a variable for woodlot size:

```
set.seed(123)
beetles$woodlot_size <- runif(nrow(beetles), 1,2)
summary(beetles)
```

```
##       ELB         temperature      woodlot_size
##  Min.   :12.00   Min.   : 5.334   Min.   :1.042
##  1st Qu.:14.00   1st Qu.:19.079   1st Qu.:1.389
##  Median :18.50   Median :24.263   Median :1.607
##  Mean   :21.04   Mean   :24.913   Mean   :1.593
##  3rd Qu.:24.25   3rd Qu.:29.701   3rd Qu.:1.890
##  Max.   :46.00   Max.   :42.869   Max.   :1.994
```

To adjust our counts for the effect of woodlot, we could add woodlot size on the left hand side of the regression equation. That is, we could divide beetles trapped by woodlot size... but as you'll see, we get warnings because we are no longer fitting integers here, we are fitting densities (beetles per trap per woodlot).

```
fit_beetles_a <- glm(ELB/woodlot_size~temperature,data=beetles,family=poisson(link=log))
```

```
## Warning in dpois(y, mu, log = TRUE): non-integer x = 10.096479
## Warning in dpois(y, mu, log = TRUE): non-integer x = 9.506208
## Warning in dpois(y, mu, log = TRUE): non-integer x = 21.292045
## Warning in dpois(y, mu, log = TRUE): non-integer x = 7.965938
## Warning in dpois(y, mu, log = TRUE): non-integer x = 9.791456
## Warning in dpois(y, mu, log = TRUE): non-integer x = 36.344282
## Warning in dpois(y, mu, log = TRUE): non-integer x = 18.323342
## Warning in dpois(y, mu, log = TRUE): non-integer x = 7.926363
## Warning in dpois(y, mu, log = TRUE): non-integer x = 11.602162
## Warning in dpois(y, mu, log = TRUE): non-integer x = 13.043943
## Warning in dpois(y, mu, log = TRUE): non-integer x = 17.886040
## Warning in dpois(y, mu, log = TRUE): non-integer x = 15.137606
## Warning in dpois(y, mu, log = TRUE): non-integer x = 13.114202
## Warning in dpois(y, mu, log = TRUE): non-integer x = 12.081646
## Warning in dpois(y, mu, log = TRUE): non-integer x = 11.786843
## Warning in dpois(y, mu, log = TRUE): non-integer x = 24.212757
## Warning in dpois(y, mu, log = TRUE): non-integer x = 18.457769
## Warning in dpois(y, mu, log = TRUE): non-integer x = 12.475295
## Warning in dpois(y, mu, log = TRUE): non-integer x = 24.097824
## Warning in dpois(y, mu, log = TRUE): non-integer x = 6.139666
## Warning in dpois(y, mu, log = TRUE): non-integer x = 7.409213
## Warning in dpois(y, mu, log = TRUE): non-integer x = 8.270305
## Warning in dpois(y, mu, log = TRUE): non-integer x = 7.924380
## Warning in dpois(y, mu, log = TRUE): non-integer x = 7.521550
```

```
summary(fit_beetles_a)
```

```
##
## Call:
## glm(formula = ELB/woodlot_size ~ temperature, family = poisson(link = log),
##     data = beetles)
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) 1.536396   0.172831   8.890  < 2e-16 ***
## temperature 0.040818   0.005728   7.126 1.03e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 74.241  on 23  degrees of freedom
## Residual deviance: 23.695  on 22  degrees of freedom
## AIC: Inf
##
## Number of Fisher Scoring iterations: 4
```

It is better to use an `offset` variable, which still standardizes our response variable (beetles trapped) by woodlot size while also keeping the predictor on the right hand side of the equation (see below). This ensures our response variable remains as an integer. Note that the below model is treating woodlot as a nuisance variable; that is, it does not appear in the list of predictors. This makes sense, since we aren't trying to determine how woodlot size affects trap catch, we are interested in how temperature affects trap catch.

```
fit3b <- glm(ELB~temperature + offset(woodlot_size),data=beetles,family=poisson(link=log))
summary(fit3b)
```

```
##
## Call:
## glm(formula = ELB ~ temperature + offset(woodlot_size), family = poisson(link = log),
##     data = beetles)
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) 0.290442   0.140818   2.063   0.0392 *
## temperature 0.042103   0.004696   8.967   <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: 136.725  on 23  degrees of freedom
## Residual deviance:  57.845  on 22  degrees of freedom
## AIC: 177.42
##
## Number of Fisher Scoring iterations: 4
```

However, if we wanted to estimate the effect of both woodlot size and temperature on trap catch AND get

statistical output for the predictive value of both variables, we would need to fit both as predictors:

```
fit3c <- glm(ELB~temperature + woodlot_size,data=beetles,family=poisson(link=log))
summary(fit3c)
```

```
##
## Call:
## glm(formula = ELB ~ temperature + woodlot_size, family = poisson(link = log),
##     data = beetles)
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)   2.017528   0.285992   7.054 1.73e-12 ***
## temperature   0.039041   0.004677   8.347  < 2e-16 ***
## woodlot_size -0.007732   0.147868  -0.052    0.958
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 83.276  on 23  degrees of freedom
## Residual deviance: 12.952  on 21  degrees of freedom
## AIC: 134.52
##
## Number of Fisher Scoring iterations: 4
```

If you want a bit more on the math of this, here is a nice explanation

*https://stats.stackexchange.com/questions/66791/where-does-the-offset-go-in-poisson-negative-binomial-regression

4 Generalized linear mixed-effects models

As mentioned above, there are also mixed-effects (ME) models for binary and count data. We can fit them using the `lme4` package, but instead of `lmer()`, we will use `glmer()`. I provide some examples below again using the `grouseticks` data and I will fit the models using the centered value for `HEIGHT`, `cHEIGHT`. Centering or standardizing predictors, [x-mean]/sd, is common in regression and can help the model algorithm fit the data if you have multiple predictors on different scales (think elevation in meters vs. grouse weight in grams).

4.1 Interpretation

You can interpret the estimates of the intercept and slope from generealized linear mixed-effects models equivalently to logistic and Poisson regression models that contain only fixed-effects.

4.2 Binomial response variables

We can again split the tick data into a binary variable as we have done previously. Generalized linear mixed-effects models that assume a Binimoial error structure can be used to model presence/absence data. Here, `BROOD` is being fit as a random intercept.

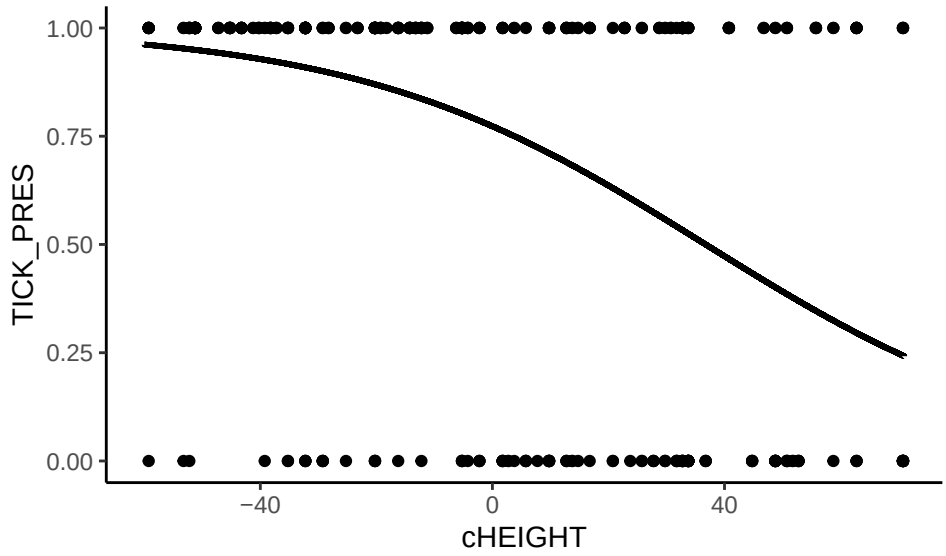
```
grouseticks$TICK_PRES <- ifelse(grouseticks$TICKS > 0, 1, 0)
library(lme4)
glmer_binomial <- glmer(TICK_PRES ~ cHEIGHT + (1|BROOD), data=grouseticks,
                        family=binomial(link="logit"))
summary(glmer_binomial)
```

```
## Generalized linear mixed model fit by maximum likelihood (Laplace
##   Approximation) [glmerMod]
##  Family: binomial  ( logit )
## Formula: TICK_PRES ~ cHEIGHT + (1 | BROOD)
##    Data: grouseticks
##
##      AIC      BIC   logLik deviance df.resid
##    414.5    426.4   -204.2    408.5      400
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -3.0293 -0.5128  0.2454  0.4344  1.8573
##
## Random effects:
##  Groups Name        Variance Std.Dev.
##  BROOD  (Intercept) 2.405    1.551
## Number of obs: 403, groups:  BROOD, 118
##
## Fixed effects:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  1.225349   0.237531   5.159 2.49e-07 ***
## cHEIGHT     -0.033344   0.006643  -5.020 5.18e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##         (Intr)
## cHEIGHT -0.301
```

4.3 Plotting mixed effects logistic regression

Note the inclusion of `re.form=NA`, which essentially tells R to avoid trying to plot the line for each grouping variable and to just plot the "global" fit.

```
new_data_me <- data.frame(cHEIGHT = seq(-60, 71, 0.001))
new_data_me$Predicted_TICKS_logistic  <- predict(glmer_binomial, newdata=new_data_me,
                                                 type="response", re.form=NA)
ggplot(data=grouseticks, mapping=aes(x=cHEIGHT, y=TICK_PRES))+
        geom_point()+theme_classic()+
        geom_line(data=new_data_me, aes(x=cHEIGHT, y=Predicted_TICKS_logistic), linewidth=1)
```



4.4 Poisson response variables

As we have covered, the variable `TICKS` is a count variable. Generalized linear mixed-effects models that assume a Poisson error structure can be used to model count data.

```
glmer_poisson <- glmer(TICKS ~ cHEIGHT + (1|BROOD), data=grouseticks,
                       family=poisson(link="log"))
summary(glmer_poisson)
```

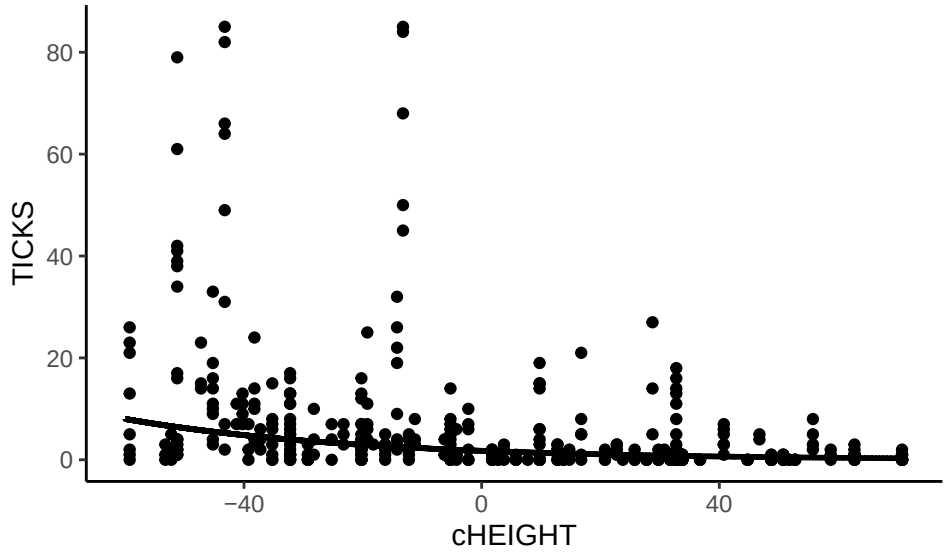
```
## Generalized linear mixed model fit by maximum likelihood (Laplace
##   Approximation) [glmerMod]
##  Family: poisson  ( log )
## Formula: TICKS ~ cHEIGHT + (1 | BROOD)
##    Data: grouseticks
##
##      AIC      BIC   logLik deviance df.resid
##   2044.5   2056.5  -1019.2   2038.5      400
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -3.0752 -0.7515 -0.4184  0.6209  5.6897
##
## Random effects:
##  Groups Name        Variance Std.Dev.
##  BROOD  (Intercept) 1.697    1.303
```

```
## Number of obs: 403, groups:  BROOD, 118
##
## Fixed effects:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.568353   0.137535   4.132 3.59e-05 ***
## cHEIGHT     -0.025214   0.003887  -6.486 8.81e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##         (Intr)
## cHEIGHT 0.085
```

4.5 Plotting mixed effects Poisson regression

Note the inclusion of `re.form=NA`, which essentially tells R to avoid trying to plot the line for each grouping variable and to just plot the "global" fit.

```
new_data_pois_me <- data.frame(cHEIGHT = seq(-60, 71, 0.001))
new_data_pois_me$Predicted_TICKS_poisson  <- predict(glmer_poisson,
                                            newdata=new_data_pois_me, type="response",
                                            re.form=NA)
ggplot(data=grouseticks, mapping=aes(x=cHEIGHT, y=TICKS))+
        geom_point()+theme_classic()+
        geom_line(data=new_data_pois_me,
                  aes(x=cHEIGHT, y=Predicted_TICKS_poisson), linewidth=1)
```



5 R Activity

5.1 Part A

Let's pretend that commercial Chilean bass cannot have a Mercury concentration higher than 0.354 (PPM). Fish above this level will be considered contaminated (and thus not marketable), whereas fish below the level will not be contaminated (= marketable). Let's determine the probability of a fish being marketable as a function of the `pH` of the lake they were sourced from. This is not my area of expertise, but we are hypothesizing here that pH levels might moderate levels of mercury contamination.

1. Load in the `bass.txt` dataset.
2. Using R, create a new column called `marketable` in which each observation of `marketable` is a 1 when `AvgMercury` is less than 0.354 and 0 otherwise (make sure `marketable` is numeric!).
3. Plot `marketable` as function of `pH`.
4. Fit a logistic regression modeling the effect of `pH` on `marketable`.
5. Reproduce the graph of `marketable` as a function of `pH` and overlay the fit/predicted line from your logistic regression.
6. Using your model, find the pH values at which there is a 50% chance of fish being marketable with mercury.
7. Write 2-3 sentences interpreting the model and provide summary statistics (e.g., odds ratios, z-values) to support any claims you make about statistical significance.

5.2 Part B

For this second part, you will analyze cricket chirps per unit of time as a function of temperature in degrees Fahrenheit.

1. Load in the `chirps.txt` dataset.
2. Plot `Chirps` as a function of `Temperature`.
3. Fit a poisson regression modeling `Chirps` as a function of `Temperature`.
4. Reproduce the graph of `Chirps` as a function of `Temperature` and overlay the fit/predicted line from your Poisson regression.
5. Write 1-2 sentences interpreting the results.