Basic Statistics for Archaeologists

Logistic and Poisson Regression

Elic M. Weitzel

4/27/2019

Workshop Objectives

Now that you have an understanding of the basics of how R works, we can begin to conduct some simple statistical analyses of archaeological data using R. However, one cannot learn all of the statistical techniques needed for analysis of archaeological data in a brief workshop...

Therefore my goal here is not to provide you with a complete tutorial on the key statistical tools you will need as archaeologists. I aim instead to teach you one or two important models, but also some important *terminology* and *how to search for answers online*. When searching for information about statistics or R programming, Stack Exchange (https://stackexchange.com/) is your best friend. The programming website under the larger Stack Exchange umbrella is Stack Overflow (https://stackoverflow.com/) while the website that focuses on statistics is Cross Validated (https://stats.stackexchange.com/). Often times, internet searches will lead you to pages associated with these websites. Odds are, if you have a question about statistics or R, someone has already had that same question, asked it online, and someone else has answered it. In the rare cases where this is not true, you can post your question on Stack Exchange and someone will help you out.

Types of Data

Introductory statistics courses often teach that there are four types of data: *nominal/categorical*, *ordinal*, *interval*, and *ratio*. These terms are perfectly fine and useful, but I find that it is easier to understand data types a slightly different, but fundamentally identical, way.

First, there is a distinction between *discrete* and *continuous* data. Discrete data are integers (i.e. no decimals or fractions): there is nothing in between values. Such data could take values such as 0's or 1's, as in the presence or absence of acorn nutshell across excavation units at a site. These data could also look like counts, as in how many pieces of lithic debitage were recovered from a site. One cannot recover 1.76 lithic flakes; such a value must be an integer. In contrast to discrete data, continuous data can take the form of decimals: there are always more (theoretically possible) values in between two other values. Measurements of human femur length are continuous data (e.g. 37.6 cm) as are the ratio of artiodactyl to lagomorph bones (e.g. 0.582) or the percentage of positive test units (e.g. 36.94%).

Second, there are *bounded* and *unbounded* data. Bounded data have maximum and/or minimum values. Most commonly, data are bounded by 0, with possible values ranging from 0 to positive infinity, or bounded by both 0 and 1. Dimension measurements and counts are examples of data that are bounded by 0. For example, one cannot measure a handaxe to be -12.7 cm long or recover -73 flakes from a test unit. Conversely, unbounded data are those which can take values from negative to positive infinity. Derived calculations such as Net Primary Productivity or change in precipitation relative to a modern standard are examples of unbounded data, as are temperature data (on non-Kelvin scales).

Types of data distributions and types of models to use with each

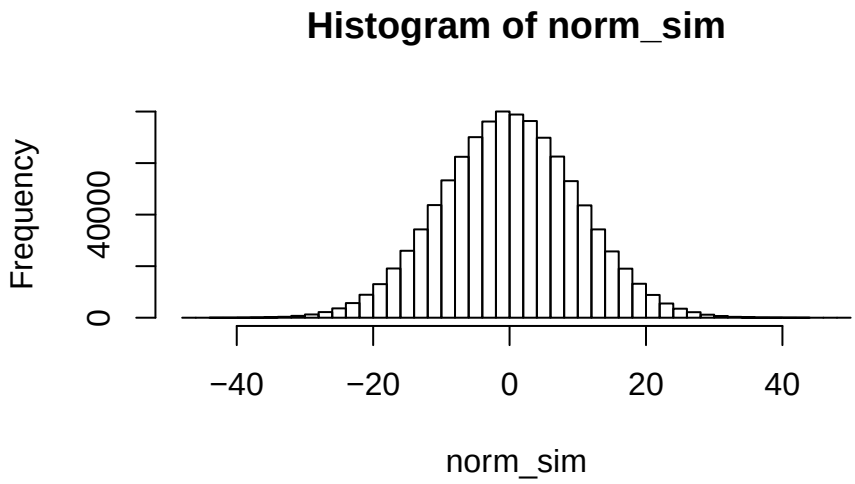
Now that you understand the difference between discrete/continuous and bounded/unbounded data, you can begin to understand something of common probability distributions. A probability distribution is a mathematical function that describes the probability of different possible values occurring. In statistical modeling and hypothesis testing, this matters because the variation in different types of data fit different probability distributions. If you use a statistical technique that is based on one probability distribution, but your data best fit another distribution, your results won't be correct.

Many of the basic statistical tests and models taught in introductory statistics courses assume that your data best fit a **Gaussian**, or normal, distribution. This is probably the best-known probability distribution. It is the well-known "bell-curve" where data are distributed evenly around a mean value based on a standard deviation. However, a lesser-known attribute of the Gaussian distribution is that it assumes that data are *unbounded* and *continuous*. That is to say, the possible values of a normal distribution range across all possible numbers, including decimals, from negative infinity to positive infinity. In reality, infinite values are not often possible, but Gaussian distributions fit data with both negative and positive continuous values well, such as the latitude of archaeological site locations (despite being bounded by -90 and +90).

Let's randomly generate a Gaussian distribution made up of one million observations:

```
norm_sim <- rnorm(n = 1000000, mean = 0, sd = 10)

hist(norm_sim, breaks = 50) #note that values range across all negative and positive numbers
```



When data are distributed normally, statistical analyses are simplified greatly for a variety of reasons that we won't get into. Most introductory statistics courses only deal with Gaussian data (or rather, treat all data as Gaussian...) for this reason. Some common statistical tests that assume normality (technically normality of residuals, not the data themselves) are t-tests, ANOVAs, and other forms of ordinary least squares (OLS) regression.

All of these tests are variations on a Gaussian linear model. Such linear models are familiar to graduates of high school statistics/mathematics courses as the familiar equation $y = mx + b$. y represents the modeled values, m represents the slope of the line (or the effect of the predictor/independent variable on the response/dependent variable), x represents values of the predictor/independent variable, and b represents the intercept of the model with the y-axis/response/dependent variable.

All linear models make the following assumptions, in rough order of importance:

1. Makes sense

- The model fits and predicts values that are realistic

2. Additivity

- Model parameters can be summed to predict y

3. Linearity

- Slope does not change

4. Independent errors

- Observations are independent of each other

5. Homoskedacity

- Residuals/errors have equal variance around model fit

6. Normality of errors

- Residuals/errors are normally distributed around model fit

Many, many data violate these assumptions!

Now, linear models are often quite robust to violations of these assumptions, but that doesn't mean that they're a one-size fits all tool. For analyzing continuous, unbounded data, linear models work well. For analyzing continuous, but bounded data, linear models can still work decently well. However, there are alternative ways to model your data that do not require strict adherence to these assumptions. Non-parametric approaches and generalized linear models are two other types of analyses that can be used in such circumstances.

Here, we'll focus on two of the more common non-Gaussian data types encountered by archaeologists: count data and proportions.

Generalized Linear Models (GLMs)

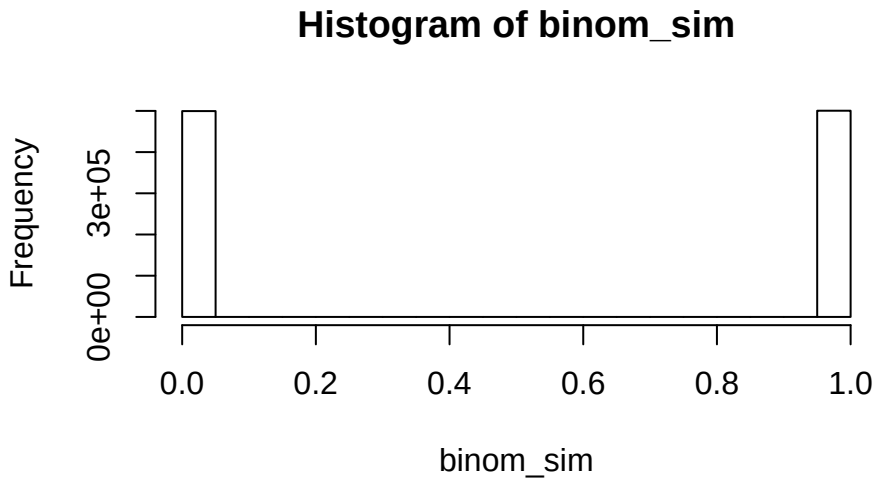
What many folks consider to be the one of the most useful way to approach modeling various types of data is through **generalized linear modeling**. A generalized linear model (GLM) is a type of parametric model, but a flexible one that can accomodate non-Gaussian data by specifying the relevant probability distribution. GLMs allow you to retain the power of parametric analyses while also permitting you to analyze data that violate assumptions 1, 5, and 6 (and with some tweaking, 3) mentioned above. Violations of the other assumptions, 2 and 4, require more sophisticated modeling approaches beyond our scope here (i.e. non-linear and hierarchical/mixed effects models).

To succesfully employ GLMs, however, one must have a bit of knowledge about alternative probability distributions and the types of data they best fit. There are many, many different types of probability distributions, but here we will only discuss two of the most common ones.

Binomial Logistic Regression

Not all of the data we collect as archaeologists can take all possible values from negative to positive infinity as the Gaussian distribution does. Sometimes, we are interested in presence/absence data: for example, whether grass phytoliths were found in a soil sample. Such data would only take values of 0, indicating an absence of grass phytoliths, or 1, indicating their presence. A binomial distribution best fits such data.

```
binom_sim <- rbinom(n = 1000000, size = 1, prob = 0.5)
hist(binom_sim) #note that all values are either 0 or 1
```



Proportion/Percentage Data

While the binomial distribution is intended for data that take values of only 0 or 1, this distribution can be used to approximate data which more generally fall between values of 0 and 1, such as percentages or proportions. This is one way to perform what's called *logistic regression*. As archaeologists very often try to model proportions and percentages, binomial-family GLMs are a nearly indispensible tool.

Let's use the `Nelson` dataset of ceramic types recovered from a pueblo in New Mexico, made available in the `archdata` package. To create a binomial-family GLM using these data, let's first load this dataset and inspect it:

```
#first, load the archdata package
library(archdata)

#then load the Nelson dataset into your environment
data("Nelson")

#inspect the first six rows in the dataset
head(Nelson)
```

```
##   Depth Corrugated Biscuit Type_I Type_II_Red Type_II_Yellow Type_II_Gray
## 1     1         57      10      2          24             23           34
## 2     2        116      17      2          64             90           76
## 3     3         27       2     10          68             18           48
```

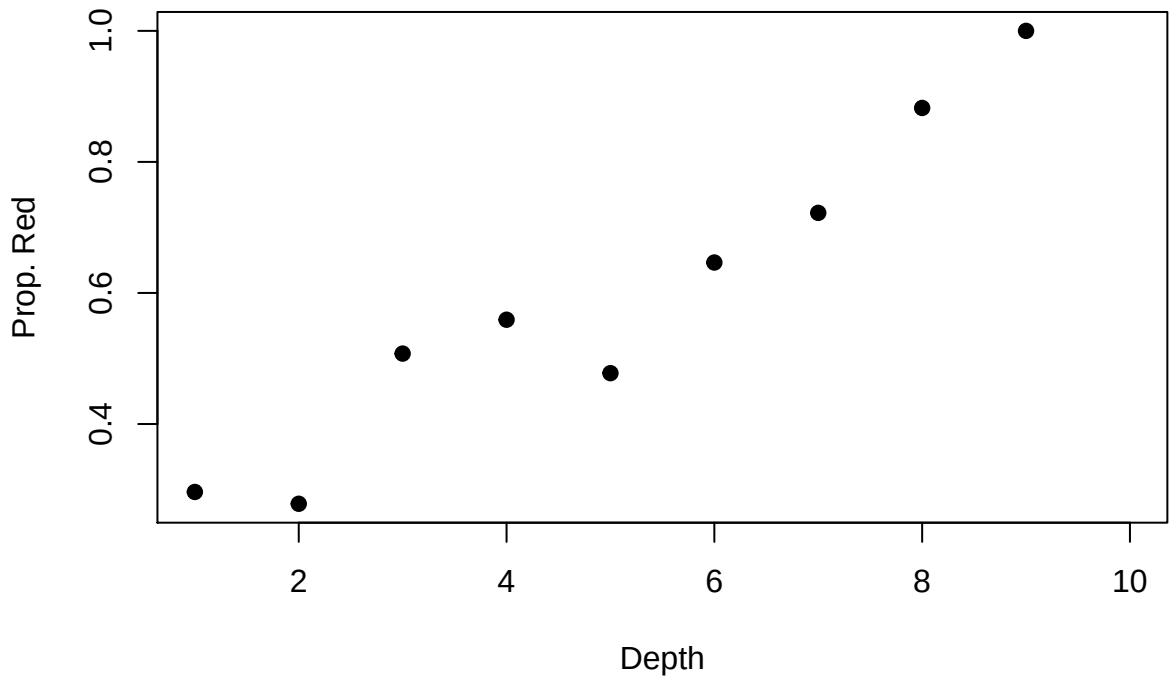
```
## 4     4          28        4       6           52              20             21
## 5     5          60       15       2          128              55             85
## 6     6          75       21       8          192              53             52
##   Type_III
## 1        5
## 2        6
## 3        3
## 4        0
## 5        0
## 6        1
```

Let's pretend we're interested in only the Type II ceramics recovered from this site. We want to know what proportion of Type II ceramics are red and how this proportion changes with depth at the site. Since the ceramic data are reported as counts, we need to create the proportion ourselves:

```
#create a new column in the Nelson data frame called RedProp and assign to it the proportions
Nelson$RedProp <- Nelson$Type_II_Red / c(Nelson$Type_II_Red +
                                         Nelson$Type_II_Yellow +
                                         Nelson$Type_II_Gray)
```

Now that we've calculated the proportion of Type II ceramics that are red for each of the ten depths, let's plot these data:

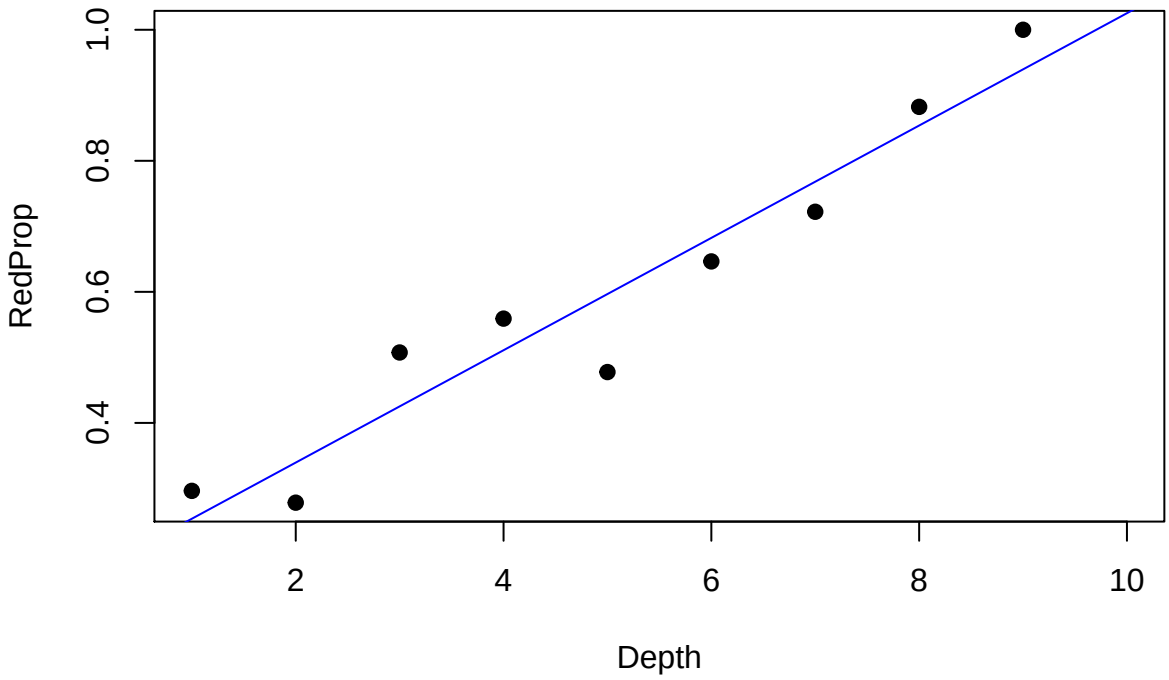
```
plot(RedProp ~ Depth, data = Nelson, pch = 19, xlab = "Depth", ylab = "Prop. Red")
```



At first glance, the proportion of red ceramics at the site seems to increase linearly as depth increases. However, adding a regular Gaussian linear model to the previous plot shows us that this isn't really the case...

```
plot(RedProp ~ Depth, data = Nelson, pch = 19)

abline(lm(RedProp ~ Depth, data = Nelson), col = "blue")
```

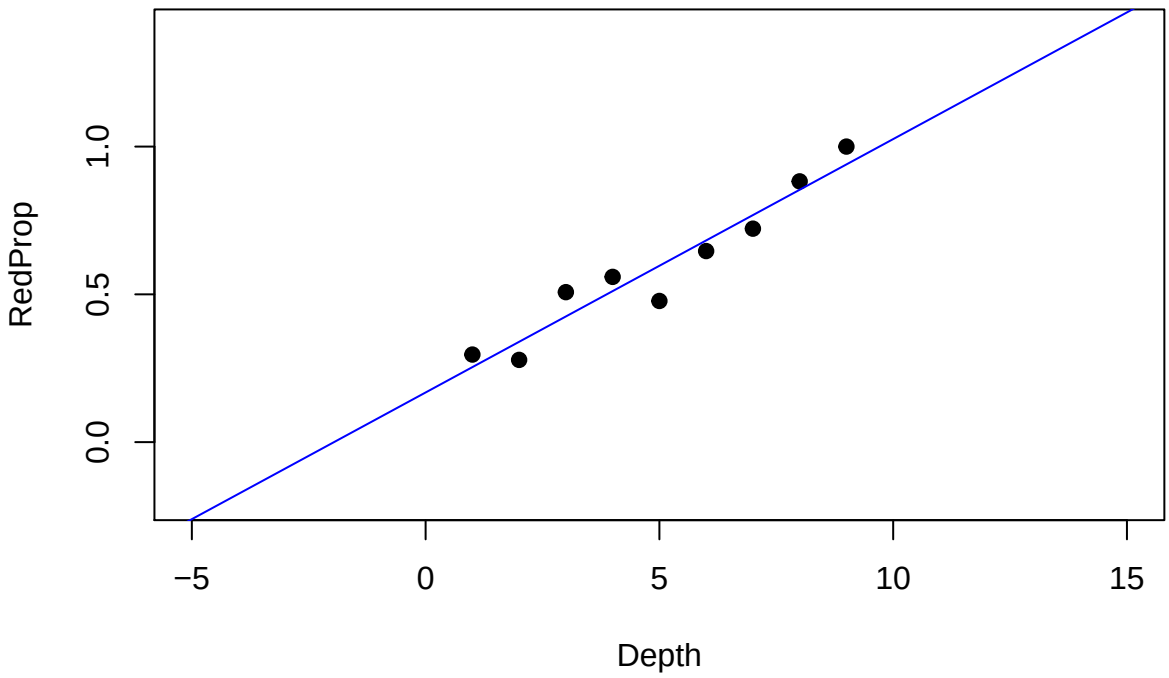


Whenever some of the smallest and largest data points are both on the same side of the regression line, this is often a clue that the data are not strictly linear... Perhaps a curved regression line would fit these data better than a straight one.

Additionally, think about the type of data we're modeling here. The proportion of red ceramics, as it is a proportion, cannot be greater than 1 or less than 0. Let's extend the dimensions of our plot here to see how our current Gaussian model is handling this:

```
plot(RedProp ~ Depth, data = Nelson, pch = 19, ylim = c(-0.2, 1.4), xlim = c(-5, 15))

abline(lm(RedProp ~ Depth, data = Nelson), col = "blue")
```



As you can see, our current Gaussian linear model is predicting that at a depth of 150 cm (depth measurements are provided as 10 cm levels), the ceramics recovered should be 140% red. That obviously makes no sense. If we were excavating the Nelson site and we were interested in predicting what ceramics we were likely to find in deeper, unexcavated levels, this model would be useless.

To build a more realistic model that better fits these data, let's employ a GLM:

```
#you will get a warning message, but don't worry about this
redmod <- glm(RedProp ~ Depth, data = Nelson, family = binomial(link = "logit"))
```

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

You should get a warning message from running this code. This is because binomial family GLMs are not technically intended for data that are not entirely 1s and 0s. We're using this distribution more loosely to model data bounded by 0 and 1, but which can take any continuous value between those bounds. This is why R is giving these warnings, but it's not really a problem: the model still works well for this kind of data.

As you can see, there is a nice `glm()` function in base R for us to use. We can use the same syntax as the `lm()` function for the first part, but there is an extra argument in the `glm()` function that we haven't yet seen: `family`. This argument is referring to the probability distribution that best fits our data. As we're dealing with proportions, which are continuous and bounded by 0 and 1, our data fit a binomial distribution well. So we specify this in the `family` argument in the `glm()` function.

If you read the help page for `glm()`, you'll see that there is another argument associated with `family` called `link`. This is referring to the link function, which is a sort of transformation conducted internally that allows our model to work. For binomial family models, the link function typically used (and the default option in

`glm()`) is **logit**. Logit is a mathematical operation that basically transforms our data so that values from 0 to 1 extend from negative infinity to positive infinity. You can see how by taking the logit of 0 and of 1 using the `qlogis()` function, which is how R calculates logits:

```
qlogis(0)
```

```
## [1] -Inf
```

```
qlogis(1)
```

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

So this logit link function is internally transforming our data so that we can better model them. The binomial argument handles how our data are distributed around the model fit (the error distribution in our model) while the logit argument describes how our model handles the bounds of our data (the deterministic part of our model).

Now let's look at the model output:

```
summary(redmod)
```

```
##
## Call:
## glm(formula = RedProp ~ Depth, family = binomial(link = "logit"),
##     data = Nelson)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -0.29168  -0.14370   0.07712   0.13048   0.46973
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -1.5700     1.6727  -0.939    0.348
## Depth         0.4132     0.3294   1.254    0.210
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2.39232  on 8  degrees of freedom
## Residual deviance: 0.42493  on 7  degrees of freedom
##   (1 observation deleted due to missingness)
## AIC: 12.012
##
## Number of Fisher Scoring iterations: 4
```

You'll note that the `summary()` function applied to a `glm` object doesn't do quite the same thing as when applied to an `lm` object... We still see the estimated parameter values, where we note that for every one unit increase in depth, the proportion of red ceramics increases by 0.4132. However, because we used a logit link function in our model, these coefficient estimates are now on the logit scale. So our interpretation is actually for every one unit increase in depth, the proportion of red ceramics increases by the logit of -1.57 (the intercept value) + 0.4132 (the slope for RedProp), which is 0.239. R calulates inverse logits using `plogis()`. However, we don't need to worry about this for such introductory purposes - it should be enough

to know that there is a positive relationship between depth and the proportion of red ceramics at this site, which we can also see from plotting the model (which we'll do soon).

While the coefficients appear in the model summary as they do with an `lm` object (with the above-mentioned twist...), nowhere in this output do we see a p-value or R^2 value! This is because the mathematics of building a GLM are not the same as those of building a linear model. I mentioned earlier that once we move away from Gaussian data, things get more complicated. One reason for this is that when you assume your data fit a normal distribution, you can mathematically calculate all of the pieces necessary to build a linear model. But such mathematical calculation is no longer possible if your data are non-Gaussian... For this reason, GLMs are fitted using something called maximum likelihood estimation. This is well beyond the scope of this tutorial, but is the reason that we don't see p and R^2 values reported in our model summary.

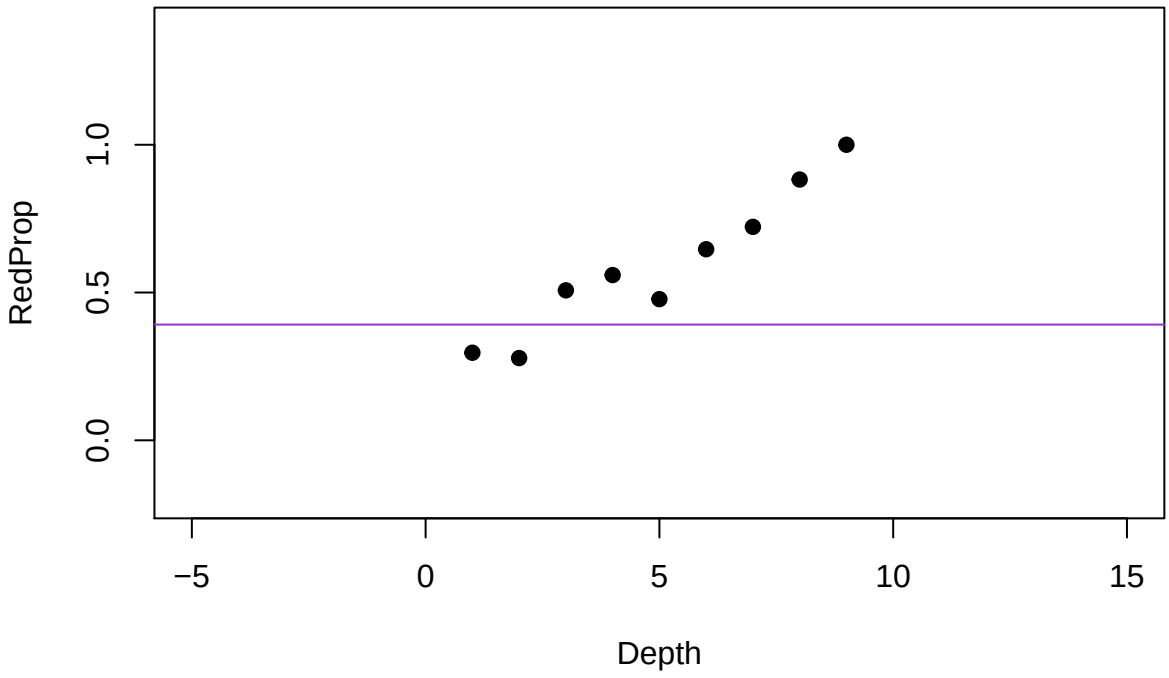
But a little additional coding, we can calculate our own versions of p and R^2 values! Granted, they're not exactly the same as true p or R^2 values, but they get the point across.

Let's start with the p-value. Essentially, what we do with GLMs is compare our GLM to a simplified version called an intercept-only model. An intercept only model basically tries to predict the response variable based only on its intercept value. If we plot the intercept-only model, we can see how poor a model it is. In R, plotting the response variable as a function of 1 indicates an intercept-only model:

```
plot(RedProp ~ Depth, data = Nelson, pch = 19, ylim = c(-0.2, 1.4), xlim = c(-5, 15))

abline(glm(RedProp ~ 1, data = Nelson, family = binomial), col="darkorchid")
```

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



But this poor-fitting model provides a useful null model against which we can compare ours. To do this, we have need to load in a new package:

```
library(lmtest)
```

```
## Loading required package: zoo
```

```
##
## Attaching package: 'zoo'
```

```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
lrtest(redmod, #our full model
       glm(RedProp ~ 1, data = Nelson, family = binomial)) #the intercept-only model
```

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

```
## Likelihood ratio test
##
## Model 1: RedProp ~ Depth
## Model 2: RedProp ~ 1
##   #Df  LogLik Df  Chisq Pr(>Chisq)
## 1   2 -4.0060
## 2   1 -5.8224 -1 3.6328    0.05665 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The `lrtest()` function in the `lmtest` package computes what is called a likelihood ratio test. This is exactly what I just described: a comparison of the GLM to an intercept-only form. What we see in the output under the `Pr(>Chisq)` column is a p-value! This p-value tells us that adding Depth as a covariate to our intercept-only model (thus creating the model we fit) *nearly* significantly improves our model (p=0.057). Thus, our model of the proportion of red ceramics as a function of depth does not actually provide a significantly better estimate of the proportion than the simple intercept-only model.

To report the results of this likelihood ratio test, we would say that including depth as a covariate in our model has a nearly significant effect on predicting the proportion of red ceramics (χ^2=3.633, df=1, p=0.057).

So now that we have calculated a p-value, we need an R^2 value. This is a bit trickier and hotly debated among statisticians... Basically, there is no agreed-upon way to calculate an R^2 equivalent for GLMs. One option is to calculate what is called a pseudo-R^2 value. There are many different versions of this, but some of the more common ones are McFadden's, Cox-Snell, and Nagelkerke.

However, pseudo-R^2 values cannot be interpreted in the same way as a true R^2 value. A McFadden's R^2 of 0.65 does not mean that 65% of the variance in the response variable is explained by the predictor variable. In fact, it doesn't really have a clear meaning other than "this model seems to fit pretty well... maybe?" So many statisticians caution against using pseudo-R^2 values.

To check out some pseudo-R^2 values, let's load the `modEvA` package and use the `RsqGLM()` function

```r
library(modEvA)

RsqGLM(redmod)
```

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

```
## $CoxSnell
## [1] 0.3321201
##
## $Nagelkerke
## [1] 0.4575966
##
## $McFadden
## [1] 0.3119689
##
## $Tjur
## [1] NaN
##
## $sqPearson
## [1] 0.8829535
```

From this output, we can see various common pseudo-R^2 values given for this model. You can see the variability in these values, and sometimes models have even more variable pseudo-R^2 values than these! You can read a lot about these different calculations on Stack Exchange if you'd like by Googling pseudo-R^2 values.

An alternative goodness of fit metric that I personally like is to calculate the proportion of deviance explained by the model. This measure was proposed by Guisan and Zimmerman (2000, *Ecological Modeling*). This is sometimes written as D^2 as a parallel to R^2, or sometimes simply written out in words. To calculate this measure of goodness of fit, we'll make use of the `Dsquared()` function in the `modEvA` package:

```r
Dsquared(redmod)
```

```
## [1] 0.8223783
```

We don't need to get into the weeds on the issue of what deviance is here, but you can kind of think of it as essentially the GLM version of variance for the time being. At least that understanding of it helps us grasp this D^2 value. So what this value tells us is that including Depth as a covariate in our model explains 82.2% of the deviance left over from an intercept-only model. If our model was perfect, it would explain 100% of the deviance remaining after fitting an intercept-only model, so our value of 82.2% is pretty good.

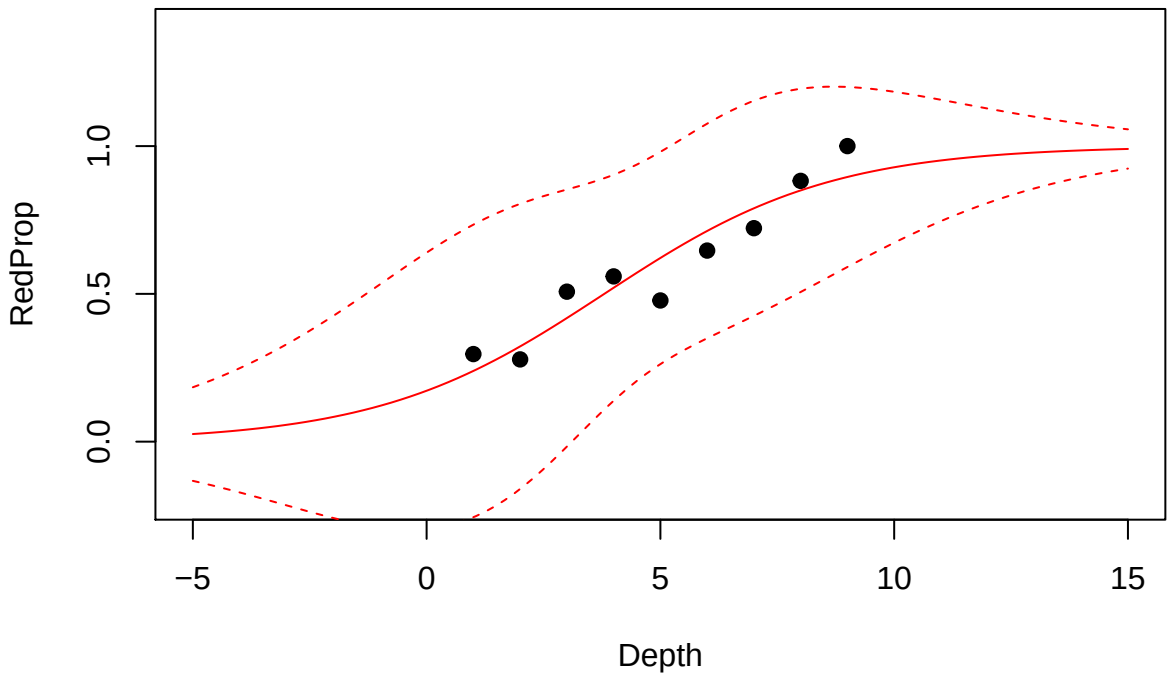
Now, let's plot this GLM. We do this using the `predict()` function. This function is able to compute confidence intervals automatically for `lm` objects, but not for `glm` objects. So what we do is tell it to compute the standard error (the `se` argument) and we can use this to calculate confidence intervals.

```r
#create a named data frame with the range of possible x values
nd <- data.frame("Depth" = seq(-5, 15, length.out = 100))

#use our glm to predict across this range of values
redmod_pred <- predict(redmod, newdata = nd, type = "response", se = T)
```

```r
#plot the model fit
plot(RedProp ~ Depth, data = Nelson, pch = 19, ylim = c(-0.2, 1.4), xlim = c(-5, 15))
lines(redmod_pred$fit ~ nd$Depth, col = "red")

#plot the 95% confidence interval
lines(redmod_pred$fit + (redmod_pred$se*1.96) ~ nd$Depth, col = "red", lty = 2)
lines(redmod_pred$fit - (redmod_pred$se*1.96) ~ nd$Depth, col = "red", lty = 2)
```



There are a few things to point out here. First, in the `predict()` function, we set the argument `type` equal to `"response"`. Doing so lets the function know that we want it to predict values on the response scale, not the link scale. In the case of binomial family logistic regression, as mentioned above, the link scale is the logit scale. So we can avoid having to do any inverse logit calculations by including this argument.

When we plot the confidence interval, we take the fitted values from the model (`$fit`) and either add or subtract the standard error values (`$se`) multiplied by 1.96. The number 1.96 is the number of standard deviations on either side of a mean value equivalent to 95%. So multiplying `$se` by 1.96 and adding it to `$fit` gives us the upper 95% confidence interval while subtracting it gives us the lower 95% confidence interval.

Plotting the model fit (`$fit`), we can see that the model only predicts values between 0 and 1. This makes our model much more realistic than a regular linear model, thus satisfying assumption 1 from above (the model must make sense). Furthermore, specifying a binomial distribution for our model better explains the patterning in the residuals (the data points relative to the model fit) than a Gaussian distribution would. This is why assigning the correct distribution and link function matter so much in creating models.

Success/Failure Data

The above model is a bit of an extension of logistic regression. Canonically, this method is meant for data that are just 0s and 1s, not the values in between. This is why those `warning` messages kept appearing before. If we want to create a binomial family GLM for success/failure (i.e. presence/absence) data, not proportions/percentages, we can plug in that variable much as we did above. There are some other ways to code this type of model that you can find by Googling things if you're interested.

Let's use the Nelson dataset again, but this time let's pretend we're only interested in whether Type III ceramics were found in a layer. We currently have counts of Type III ceramics by layer, but let's create a new column in this dataset which contains a 1 if the type is present and a 0 if the type is absent. Don't worry about the code below for our purposes here, but if you want to look this up on your own, this is an *if... else* statement within a *for loop*.

```
for(i in 1:nrow(Nelson)){ #for each row (i.e. depth) in the Nelson object...
  if (Nelson$Type_III[i] > 0) { #if there are more Type III ceramics than 0...
    Nelson$Type3[i] <- 1 #assign that row a 1 in this new vector called Type3
  } else { #otherwise (i.e. if there are 0 Type III ceramics in the row)...
    Nelson$Type3[i] <- 0 #give that row a 0 in the new Type3 vector
  }
}
```

Now that we have a new vector with our presence/absence data, we can create and inspect the output from our model.

```
type3mod <- glm(Type3 ~ Depth, data=Nelson, family=binomial)

summary(type3mod)
```

```
##
## Call:
## glm(formula = Type3 ~ Depth, family = binomial, data = Nelson)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.3551  -0.4203  -0.1627   0.4305   1.7676
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)   3.8795     2.5650   1.512   0.1304
## Depth        -0.8677     0.5187  -1.673   0.0944 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 13.460  on 9  degrees of freedom
## Residual deviance:  7.108  on 8  degrees of freedom
## AIC: 11.108
##
## Number of Fisher Scoring iterations: 5
```

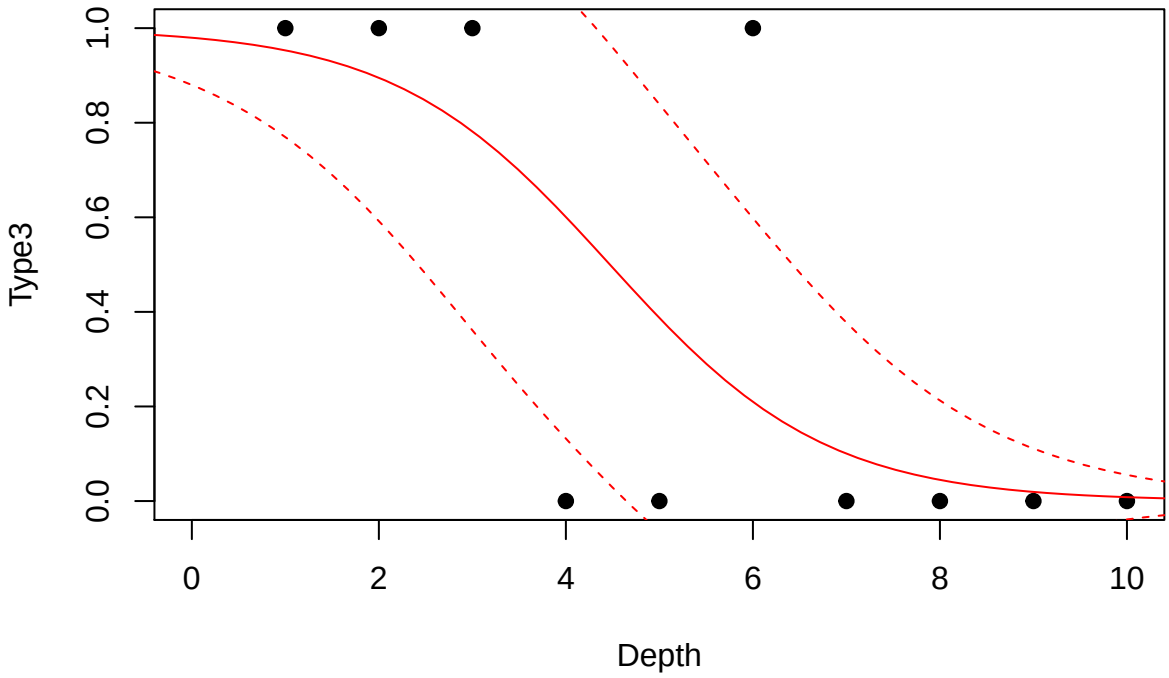
Now let's plot this model:

```r
#we can use the same nd object as above, or define it again
nd <- data.frame("Depth" = seq(-5, 15, length.out = 100))

#use our glm to predict across this range of values
type3mod_pred <- predict(type3mod, newdata = nd, type = "response", se = T)

#plot the model fit
plot(Type3 ~ Depth, data = Nelson, pch = 19, ylim = c(0.0, 1.0), xlim = c(0, 10))
lines(type3mod_pred$fit ~ nd$Depth, col = "red")

#plot the 95% confidence interval
lines(type3mod_pred$fit + (type3mod_pred$se*1.96) ~ nd$Depth, col = "red", lty = 2)
lines(type3mod_pred$fit - (type3mod_pred$se*1.96) ~ nd$Depth, col = "red", lty = 2)
```



Now that we've plotted the model (and inspected the `summary()` output) can see that there is a negative relationship between Type III ceramic presence and depth. Let's see if this relationship is strong and/or significant:

```r
#I'm assuming here that the lmtest and ModEvA packages are already loaded
lrtest(type3mod,
       glm(Type3 ~ 1, data=Nelson, family=binomial))
```

```
## Likelihood ratio test
##
## Model 1: Type3 ~ Depth
```

```
## Model 2: Type3 ~ 1
##   #Df  LogLik Df  Chisq Pr(>Chisq)
## 1   2 -3.5540
## 2   1 -6.7301 -1 6.3522    0.01172 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
Dsquared(type3mod)
```

```
## [1] 0.4719223
```

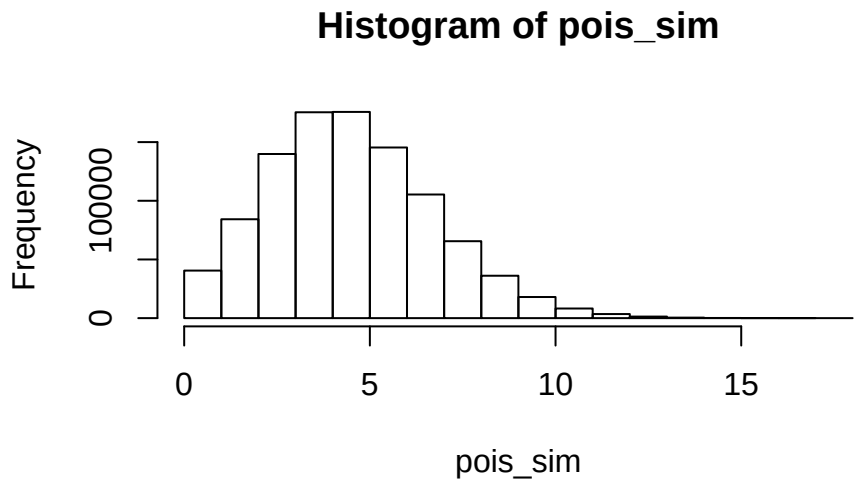
So our likelihood ratio test tells us that this model performs significantly better than an intercept-only model (p = 0.012). Our D^2 value is 0.472 which is pretty decent for archaeological data. Based on this, we can state that the presence of Type III ceramics at the Nelson site significantly declines with depth (χ^2=6.352, df=1, p<0.05) and that this relationship is moderately strong (D^2=0.472).

For logistic regression that deals with success/failure data, another common way to evaluate your model is by calculating the area under the receiver operating curve. This is beyond what we're dealing with here, but if you Google around, you can find tutorials on creating a receiver operating curve (ROC) for your logistic regression model and calculating the area under the curve (AUC) to assess how good your model is.

Poisson

Another very common type of data is *count* data. Archaeologists count many things, from sites to flakes to bones. Count data are often best modeled using a Poisson distribution, which takes positive integer values from 0 to infinity. These types of data are **discrete** not continuous, as decimals are not possible, and bounded only by 0.

```
pois_sim <- rpois(n = 1000000, lambda = 5)
hist(pois_sim, breaks = 20) #note that all values are positive integers
```



To try out a Poisson family GLM, let's use the `Snodgrass` dataset from the `archdata` package. This dataset contains information on artifacts recovered from Mississippian period house pits in Missouri.

```
data("Snodgrass")

head(Snodgrass)
```

```
##     East  South Length Width Segment  Inside  Area Points Abraders Discs
## 1 901.39  75.07   12.0  12.0       2 Outside 144.0      0        1     0
## 2 973.01  81.33   16.0  16.0       2 Outside 256.0      0        0     0
## 3 889.71 163.21   17.0  18.0       1  Inside 306.0      1        0     1
## 4 924.16 193.10   21.0  21.5       1  Inside 451.5      2        1     1
## 5 911.90 216.55   20.5  20.0       1  Inside 410.0      3        2     2
## 6 939.73 250.75   16.5  16.0       1  Inside 264.0      0        0     0
##   Earplugs Effigies Ceramics Total Types
## 1        0        0        0     1     1
## 2        0        1        0     1     1
## 3        0        1        1     4     4
## 4        1        0        5    10     8
## 5        1        0        4    12     9
## 6        0        0        0     0     0
```

Let's make a model of how many distinct artifact types are recovered from house pits of various sizes. One could reasonably expect larger houses to contain a greater variety of artifact types, so let's test this out:

```
snodmod <- glm(Types ~ Area, data = Snodgrass, family = poisson)
```

Now, the main thing that is different in this model as opposed to our previous ones is that the `family` argument is now set to `poisson`. This denotes that our data are counts, i.e. they are positive integers (no decimals and no negative values). Our model will therefore assume that the variation in the data around the model fit is distributed according to a Poisson distribution.

We didn't explicitly define it in the model (though we could), but the default link function for a Poisson model is a logarithm. Specifying `family = poisson(link="log")` would work too, it's just not necessary since the log link is the default. This link function means that the GLM is internally taking the log of our values when building the model. Thinking about positive integer data like ours, values below 0 are not possible. If we take the log of 0...

```
log(0)
```

```
## [1] -Inf
```

...we get negative infinity. Technically, the logarithm of 0 is undefined, but the log of super small numbers approaches negative infinity. Just as taking the logit of 0 and 1 transformed our binomially-distributed data above, this log link function is therefore putting our positive integer data on a scale from negative to positive infinity so a linear model can be fit.

Now that we understand what's going on with this model, let's inspect it.

```
summary(snodmod)
```

```
##
## Call:
## glm(formula = Types ~ Area, family = poisson, data = Snodgrass)
##
```

```
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.5272  -1.5914  -0.2492   0.8987   2.7011
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -0.5772267  0.1659427  -3.478 0.000504 ***
## Area         0.0074230  0.0005113  14.517  < 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: 405.38  on 90  degrees of freedom
## Residual deviance: 195.19  on 89  degrees of freedom
## AIC: 431.6
##
## Number of Fisher Scoring iterations: 5
```

As with all GLMs, model R^2 and p-values are not computed. Given the link function we're using, our model coefficients are reported on the log scale; remember that if you try to interpret what's going on with those. For example, for every 1 unit increase in house pit area, the number of artifact types increases by `10^(-0.5772267 + 0.0074230)`. This is because you must exponentiate a logarithm to get it back on our regular response scale.

But again, for most archaeological purposes, visualizing and evaluating the model is more than enough. So let's plot it!

```
#we need to define a new newdata object to cover the range of area values here
range(Snodgrass$Area) #looks like were dealing with values between 20 and 500
```

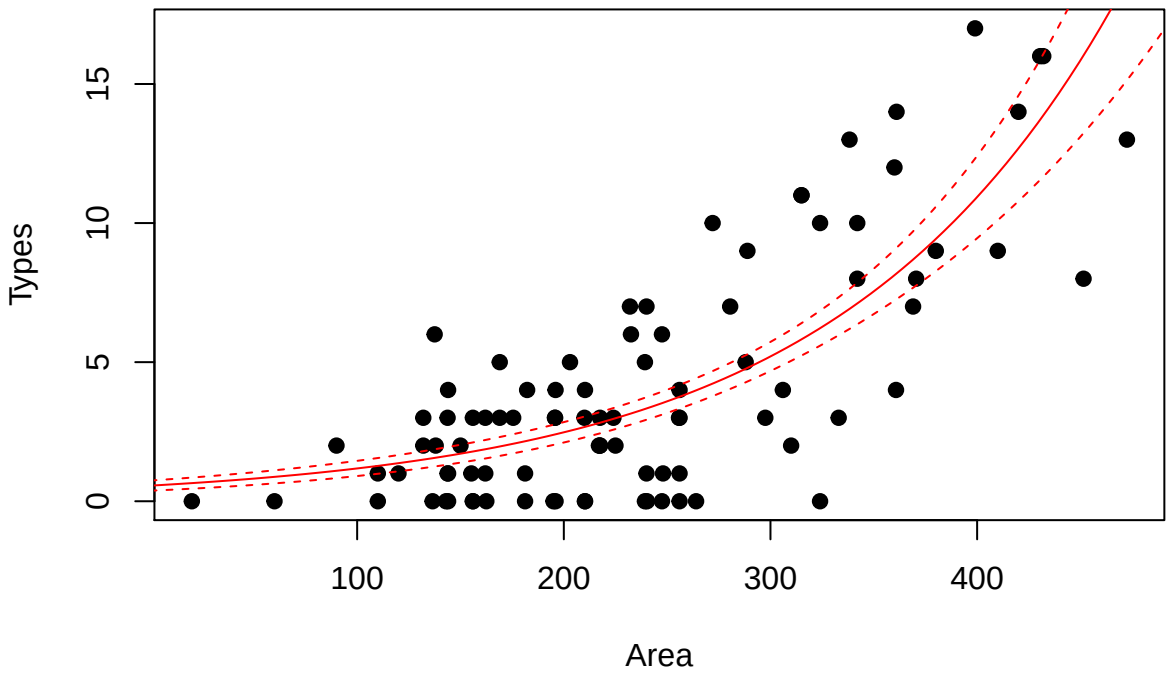
```
## [1]  20.0 472.5
```

```
nd_snod <- data.frame("Area" = seq(0, 500, length.out = 100))

#use our glm to predict across this range of values
snodmod_pred <- predict(snodmod, newdata = nd_snod, type = "response", se = T)

#plot the model fit
plot(Types ~ Area, data = Snodgrass, pch = 19)
lines(snodmod_pred$fit ~ nd_snod$Area, col = "red")

#plot the 95% confidence interval
lines(snodmod_pred$fit + (snodmod_pred$se*1.96) ~ nd_snod$Area, col = "red", lty = 2)
lines(snodmod_pred$fit - (snodmod_pred$se*1.96) ~ nd_snod$Area, col = "red", lty = 2)
```



Looks good! There's clearly an exponential and positive relationship between artifact types and house pit area at this site. But let's see if this is significant and how strong this relationship is:

```
#I'm assuming here that the lmtest and ModEvA packages are already loaded
lrtest(snodmod,
       glm(Types ~ 1, data = Snodgrass, family = poisson))
```

```
## Likelihood ratio test
##
## Model 1: Types ~ Area
## Model 2: Types ~ 1
##   #Df  LogLik Df  Chisq Pr(>Chisq)
## 1   2 -213.80
## 2   1 -318.89 -1 210.19  < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
Dsquared(snodmod)
```

```
## [1] 0.5184912
```

Our model is significantly better than an intercept only model (p=2.2e-16) and our D^2 value of 0.518 is pretty good, suggesting 52% of the deviance left over in an intercept-only model of artifact type can be explained by area.

We now know that there is a positive, exponential relationship between the number of artifact types and house pit area, and that this relationship is significant (χ^2=210.19, df=1, p<0.0001) and moderately strong (D^2=0.518).

Practice

1. Take a look at the `OxfordPots` dataset in the `archdata` package. Model the relationship between the percentage of Oxford pottery and distance to Oxford in miles.
2. Now use the `Mesolithic` dataset in the `archdata` package to investigate whether the number of microliths present at a site predicts the number of scrapers recovered.