Lecture 9. Odds and ends: data types that don't fit into normal/Poisson/binomial

What is a generalized linear model?

So far we've discussed Poisson, negative binomial, and binomial GLMs, as well as quasi-Poisson and quasi-binomial extensions. We've also discussed zero-inflated models, which are a mixture of binomial and Poisson or negative binomial models. Let's take a step back and think about how generalized linear models are defined, um, in general.

The idea of the GLM was created in the 1970s, as an overarching framework uniting several important statistical techniques. The framework encompasses these conditions:

1) The data comes from a probability distribution that is in the *exponential family*

2) Predictors in the model are specified using a linear model

3) The linear predictors are related to the response variable with a link function

The exponential family of probability distributions is large and includes most of the distributions you will encounter, including the normal (aka gaussian), poisson, binomial, negative binomial, beta, gamma, exponential, and others. I won't get into the details of why these distributions all belong in one family together, but the punchline is that a common framework can be used for fitting and analyzing models using these distributions.

As we've already seen, GLMs are an extension of linear models in the sense that a set of predictors is specified like a linear model, and then the value predicted by this model is related to the data with a link function. So although these models are in reality nonlinear (usually), they can be constructed and analyzed using the same kind of conceptual framework used in classic multiple regression and analysis of variance.

Let's take a quick look at the help file for the 'family' object used to specify a probability distribution with glm():

family {stats} R Documentation

Family Objects for Models

Description

Family objects provide a convenient way to specify the details of the models used by functions such as `glm`. See the documentation for `glm` for the details on how such model fitting takes place.

Usage

```
family(object, ...)

binomial(link = "logit")
gaussian(link = "identity")
Gamma(link = "inverse")
inverse.gaussian(link = "1/mu^2")
poisson(link = "log")
quasi(link = "identity", variance = "constant")
quasibinomial(link = "logit")
quasipoisson(link = "log")
```

This is a function in the 'stats' package, which is part of the base R install. The basic family options are listed, and most of them we've already encountered: the binomial, poisson, quasibinomial, and quasipoisson. The gaussian family is another name for the normal distribution. The gaussian distribution is part of the exponential family, and so classic linear modeling with normally distributed residuals is a special case of GLMs. The are a few other options listed here: Gamma, inverse gaussian, and quasi. The quasi option lets you construct your own quasi-likelihood model, and we won't go into that further. The inverse gaussian distribution can be used for modeling things like lifespans, but we won't go into that either. The gamma distribution is similar to the lognormal distribution, and can be used for modeling continuous data that is always greater than zero. I'll talk more about it shortly.

These family options are the defaults in R because they are commonly used. It is possible to use other GLM (or GLM-ish) distributions as well by installing other packages, such as the negative binomial that we implemented with the MASS package. Another option I'll mention later is the beta distribution. It's worth noting that there are some distributions that are used with a linear model framework, but that are not technically GLMs because the distribution is not in the exponential family. The Weibull distribution used in survival analysis is an example of this.

Let's sum up what we've learned so far, and what kinds of questions we can ask with these models.

1) Fit models with linear-type predictors. This lets us ask questions like: as some predictor X increases, does Y tend to increase, or decrease? Does the slope of response for X_1 change as the value of another predictor X_2 changes

(i.e. interactions)? Do different groups have different mean values of Y? Do different groups have different slopes of response to a predictor X (also interactions)?

2) Ask these kinds of questions with data that is not normally distributed.

3) Analyze models fit with maximum likelihood, using likelihood ratio tests and confidence intervals from likelihood profiles.

These methods are useful for many questions in biology, and are widely used. But they don't cover all the kinds of questions or kinds of data you will encounter. Much of the rest of this course will extend GLMs to a wider range of applications, and we will cover the following:

Extensions of GLMs:

- fitting nonlinear relationships with nonparametric smoothers (GAMs)
- accounting for structure in the data (non-independence), using random effects (mixed models) and using correlated residual models (generalized least squares, generalized estimating equations)

Other methods to test hypotheses on fitted models: information criteria, bayesian methods

Multivariate response data: ordination and related approaches that arise for analyzing community composition and other multivariate responses

Nonlinear models: GLMs have nonlinear link functions but are still essentially linear models. Some questions involve truly nonlinear models, especially when we have an *a priori* mechanism we would like to test or quantify, like Michaelis-Menten curves for nutrient uptake or functional responses for predation rates.

There are many more specialized models like mark-recapture models, phylogenetic estimation, and structural equation modeling that fall under 'models that are fit with likelihood methods but don't really fit into the GLM framework'. We won't dig into such things in this course but the methods we cover should prepare you to understand how they work.

Other kinds of data that need to be modeled in biology

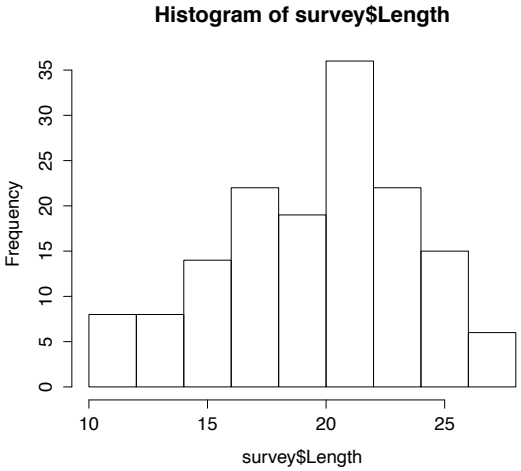
A large proportion of data in biology can be modeled with the normal, Poisson, or binomial distributions (or their overdispersed extensions). However, there are a number of important cases that arise which do not fit easily with any of these distributions. Some of these cases are difficult and have been handled in various

ways, which means that recommending a protocol is more difficult, but I will cover some relatively straightforward options.

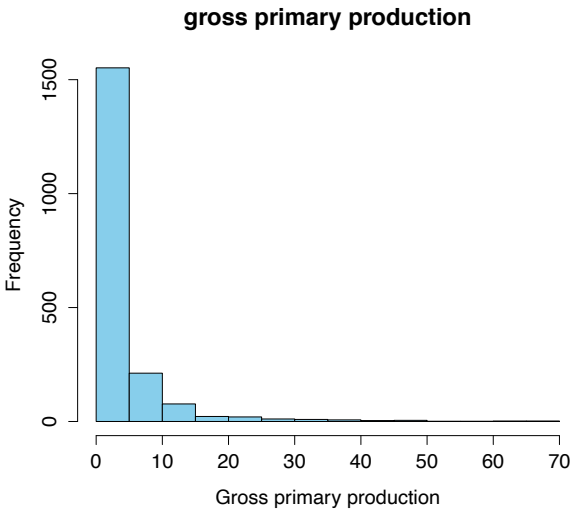
Positive continuous data

Some kinds of data are always greater than zero. Sizes are a good example, e.g. size of organisms or sizes of parts of organisms. Also concentrations (e.g. nutrient concentration, toxin concentration) are in principle always positive, though due to detection limits there may be values recorded as zero as well. Rates are another common instance, e.g. ecosystem rates of gross primary production or respiration are almost always greater than zero.

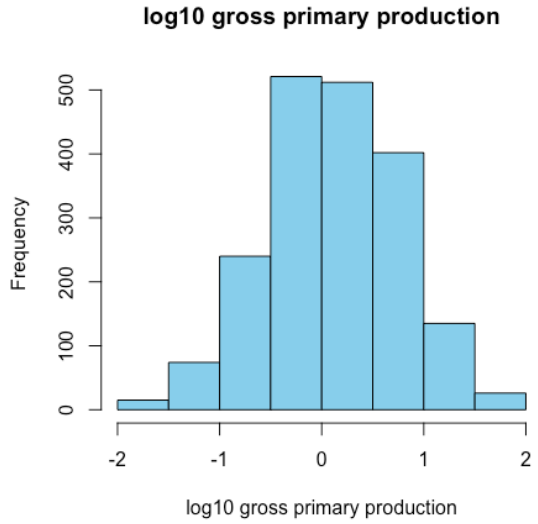
Positive continuous data is somewhat similar to count data, except that the values are continuous instead of discrete, and there are no zeros. But similar to count data, the shape of the distribution often depends on how close the distribution is to zero. For example, in the first lecture we looked at the length of isopods from different populations, and the distribution of isopod sizes was approximately normal:



The Poisson distribution becomes approximately normal for large values of λ, but for small λ the distribution is increasingly skewed. The same is often true for positive continuous data, for example here is data from a compilation of the rate of gross primary production at different locations in the ocean:



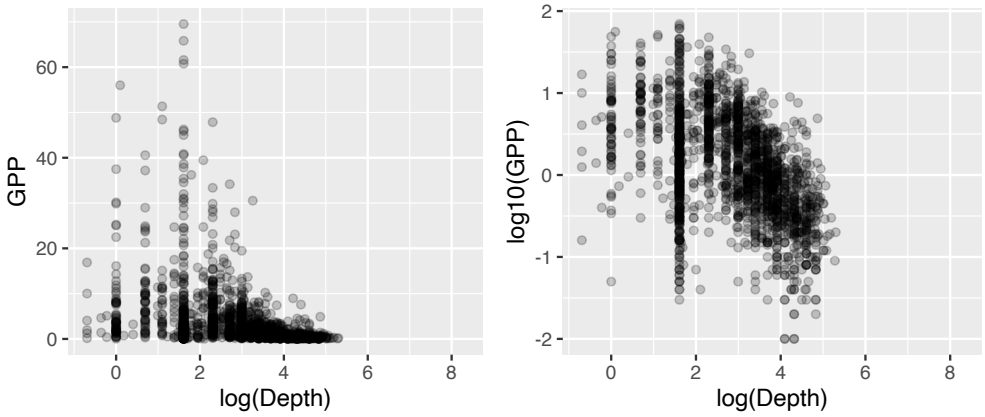
These values are all positive, but the distribution is smashed against zero and has a long tail of large values. This kind of data is often more normally distributed on a log scale:



Or in other words, the data appears to be *lognormally distributed*. The prior example of isopod lengths is also consistent with a lognormal distribution, because the lognormal (like the Poisson) is approximately normal if the mean is large (away from zero) and if the variance is not too big (i.e., the data doesn't hit the lower bound at zero).

If data are approximately lognormally distributed, and we want to make a linear-type model, how do we do that? The easiest option is to just log-transform the data and then use a normal distribution. For example, let's say we wanted to see if GPP is related to the depth in the ocean where the measurement was taken. This is

plausible because primary production depends on irradiance, which diminishes exponentially with depth in the ocean.



The plot on the left shows GPP vs log depth, and the plot on the right shows log GPP vs log depth. For now ignore the fact that I've log transformed the x-axis as well; we're concerned here with the consequence of log-transforming the response variable. If we fit a linear regression to the data on the right, then that model can be written like this:

$$\mu_{\log,i} = a + b * X_i$$
$$\log(Y_i) \sim \text{Normal}(\mu_{\log,i}, \sigma_{\log})$$

where *Y* is GPP, and *X* is log depth. I've given μ and sigma the subscript 'log' to indicate that these are on the log scale. If we rewrite this model on the scale of the original data (i.e. not log-transformed), it looks like this:

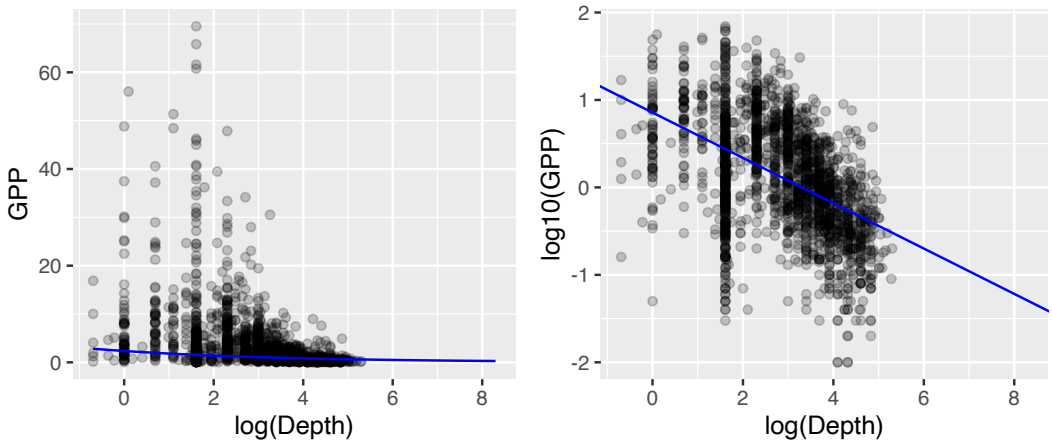
$$\mu_i = \exp(a + b * X_i)$$
$$Y_i \sim logNormal(\mu_i, \sigma)$$

This should look somewhat familiar, because it is similar to a GLM for a Poisson or negative binomial distribution. Essentially this is a GLM with a log link function, i.e. the mean of the data is an exponential function of the predictors, and the data is distributed according to a lognormal distribution with a mean determined by the predictors, and with a variance to be estimated from the data. We can fit such a model to the GPP data by using lm() on the log-transformed data:

```
mod = lm(log(GPP) ~ logDepth, data = resp)
```

Then we can plot the fitted curve on the log scale, as a straight line predicted by lm(), or we can plot the back-transformed exponential curve on the original scale, using geom_function():

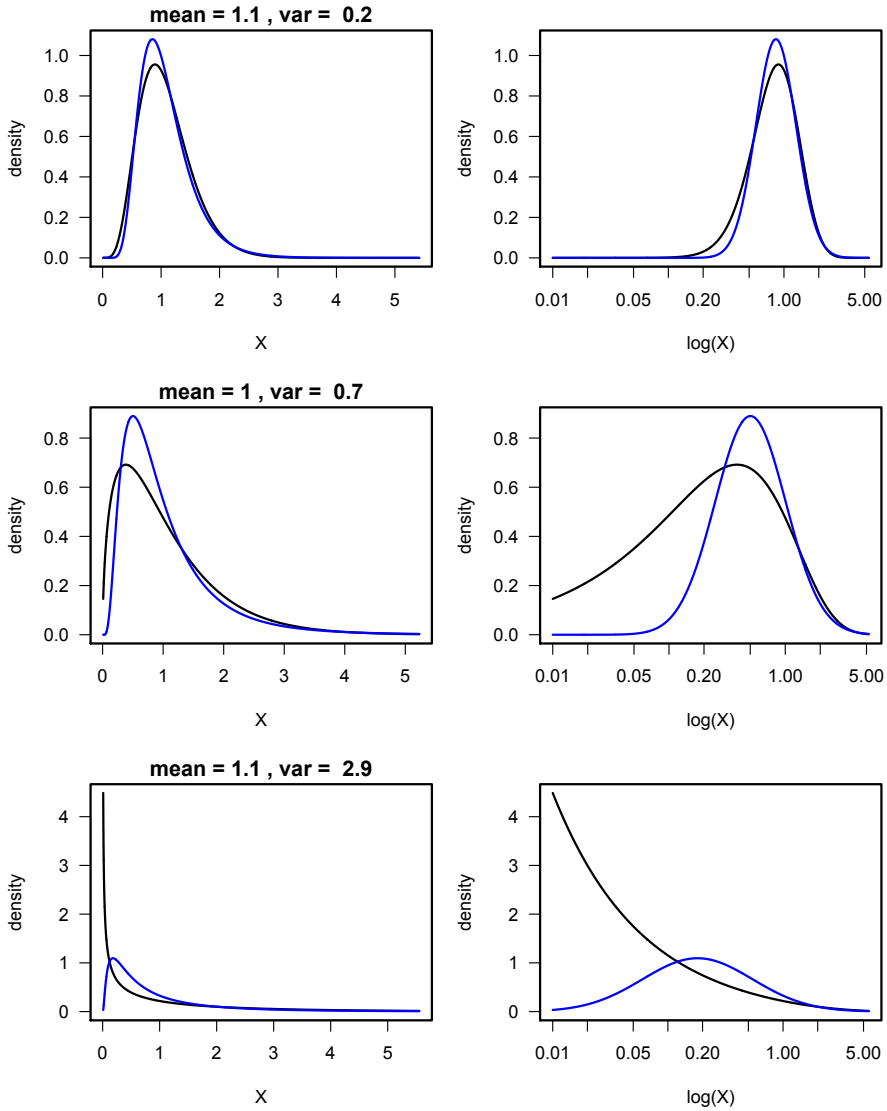
```
p1 = ggplot(resp, aes(log(Depth), GPP)) + geom_point(alpha = 0.2) + geo
m_function(fun = ~ exp(coef(mod)[1] + coef(mod)[2]*.x), col = 'blue')

p2 = ggplot(resp, aes(log(Depth), log10(GPP))) + geom_point(alpha = 0.
2) + geom_abline(intercept = coef(mod)[1], slope = coef(mod)[2], col =
'blue')

grid.arrange(p1, p2, nrow = 1)
```



We can see from these plots that there is probably some nonlinearity we aren't accounting for here, because GPP appears to flatten out between -1 and 1 on the log depth axis. However, this model is a good first pass and shows how the lognormal can be used to model positive continuous data in a linear model framework.

It would be convenient if the lognormal could be specified in the glm() function as well, but there is apparently some debate about whether the lognormal counts as an 'exponential family' distribution, and therefore whether this kind of model is a GLM *per se*, or just looks like a GLM. In any case these are arcane matters.

An alternative to the lognormal is the gamma distribution. These two distributions have a similar shape for some parameter values, and distinct shapes for other parameter values:



Each plot compares a probability density function for the gamma distribution (black) and lognormal distribution (blue), with the same mean and variance. The different rows have different variances, and the plots on the right show the same pdf as the plots on the left, but with the x-axis on a log scale.

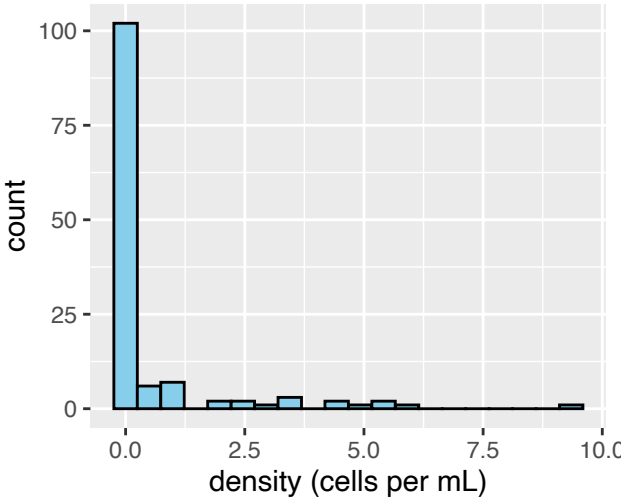
When the variance is small/medium, the distributions have similar shape. Note that on a log scale, the lognormal distribution is always symmetric (because it is a normal distribution on the log scale). When the variance gets large, the gamma distribution no longer has a unimodal shape (it has no mode), but continues to increase as x -> 0. This suggests that if the data have such a monotonic (not unimodal) shape, a gamma distribution may be more appropriate. To use a gamma instead of lognormal for the example above, use glm() on the untransformed data with family = gamma(link = 'log') [note the default link function for the gamma distribution is the inverse function, but log may make more sense in many situations). One way to assess the appropriateness of lognormal vs. gamma would

be to fit a lognormal model and see to what extent the (log) residuals violate normality. I usually start with the lognormal option because when we start doing mixed models, using normally distributed error is computationally easier than using one of the other probability distributions.

Semi-continuous data

One of the trickiest data types I've encountered is data that is continuous but also includes zero. So this is the same kind of data as the positive continuous case I just looked at, but the data includes zero as well. You could also call this non-negative continuous data. A common example of this data is densities, e.g. of organisms or other quantities. The examples I gave for positive continuous data, such as nutrient concentrations or rates of some process, could also be semi-continuous if they include zeros.

Let's look at example of organism densities. In principle, if organisms are counted, then you could just analyze the count data with a Poisson or negative binomial distribution, using an offset to account for any variation in sampling effort. However, I've encountered many instances of large surveys or time series where count data was converted to densities, and the original counts weren't saved. For example, the English Channel phytoplankton data we looked at previously is recorded as densities:



This is the same Eucampia data we looked at before, but previously I had turned all the positive densities to 'present', and we analyzed the data as presence/absence. So in reality the data is about half zeros, while the other half is a wide range of densities, which is typical for phytoplankton which can 'bloom' at relatively high density but effectively disappear at other times/places.

Similar to Poisson and lognormal data, semi-continuous data is often highly skewed, especially if there are a lot of zeros. Because this data arises commonly, people have come up with many methods to analyze it. Ideally we could analyze

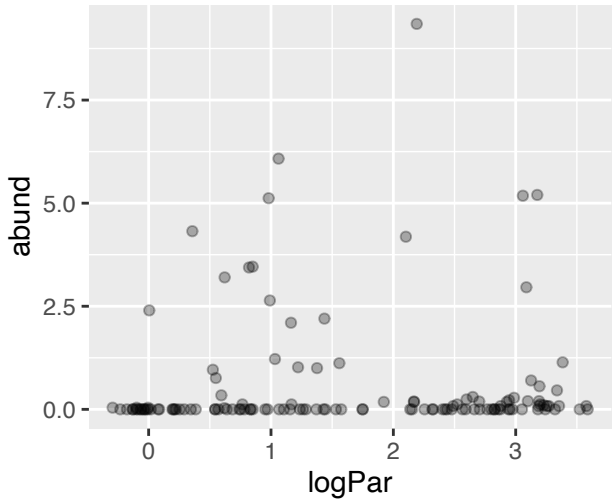
this data using a probability distribution that ranges from zero to infinity and takes on a continuous range of values. I am aware of one such distribution, called the Tweedie distribution, but it is difficult to work with because the probability density function does not have a closed formula and has to be quantified using infinite series. Another approach is to assume that the data have been truncated: in reality the density of *Eucampia* is always positive, but when the density drops below some threshold it is recorded as 0, because we didn't see any *Eucampia* in our counts. A simple model for such truncated data is a Tobit model.

If you have to deal with semi-continuous data and you're not sure how best to model it, I recommend exploring the Tweedie distribution and the Tobit model. But I'm not going to use them here, because such specialized models don't allow you to incorporate other extensions such as random effects or additive smoothers for nonlinearity, and these extensions are often important in my experience. So let's consider two other options: 1) transforming the data, and 2) modeling the zeros separately from the non-zeros.

Transforming the data

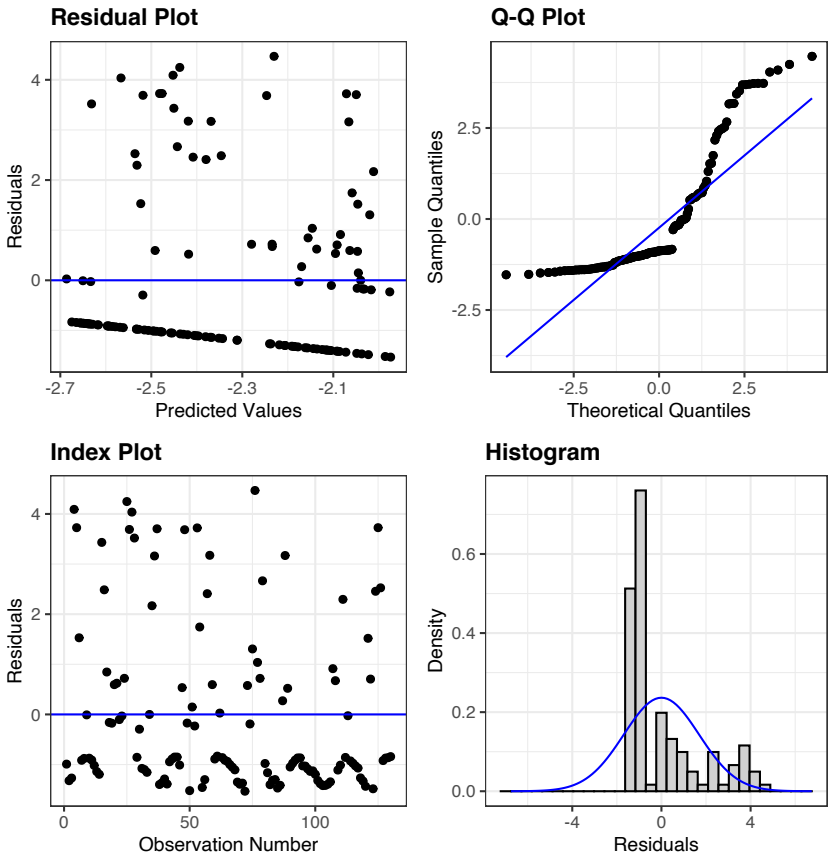
The general philosophy of modern statistics is to avoid transformation when possible. Classically, transformation was necessary to meet assumptions that 1) the data are normally distributed, and 2) the variance in the data is constant (homoscedasticity). Now that GLMs and extensions have been developed, it is often possible to pick a probability distribution that matches the data well, and this means that models will fit the data better and provide more accurate and sensitive tests of hypotheses. Earlier I log-transformed data that was always positive, but that was because the data fit a lognormal distribution, and fitting lm() to log-transformed data is equivalent to fitting a lognormal distribution. However, situations still arise where there is no appropriate/convenient distribution to use, and transformation is the best option.

Can we transform the *Eucampia* data to make it look normal? Specifically, I'd like to quantify how Eucampia response to irradiance (PAR), as part of a project to see how different species respond differently to the environment. It's pretty clear from looking at the data that a linear model with normally distributed data is a bad idea:



A log transformation is not a good option because there are many zeros, and the log of zero is undefined. One option that is common is to add a small value to the data so that there are no zeros. E.g. adding the smallest non-zero value in the dataset to all the data. Let's try it:

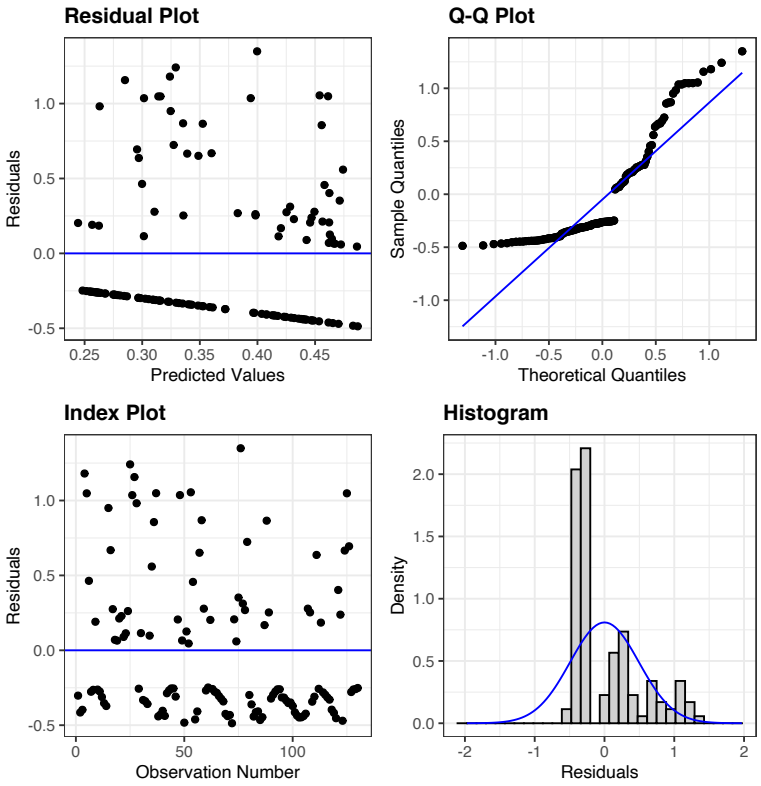
```
#log transform and add a small number
datause$abund.fixed = datause$abund
datause$abund.fixed = datause$abund.fixed + min(datause$abund.fixed[dat
ause$abund.fixed > 0])

mod.log = lm(log(abund.fixed) ~ logPar, data = datause)
resid_panel(mod.log)
```



The residual plots look pretty funky. There is a band of values along the bottom of the residuals vs fitted plot, because those are the former zeros. The normal quantile plot shows a pretty large deviation at the upper values, because the large densities are much larger than expected, even after the log transformation. In general, if there are a small number of zeros then doing a log(x + small.number) transformation can work pretty well, but a large proportion of zeros is problematic.

Another transformation option is to take the quarter-root of the data (or some other root). The quarter-root is similar to the log transform, in that it spreads out the distance between small values and shrinks the distance between large values. And it has the advantage that the quarter root of 0 is 0, so you can include zeros. Let's try this one:

```
#quarter root transform
datause$quarter.abund = datause$abund^0.25
mod.quarter = lm(quarter.abund ~ logPar, data = datause)
resid_panel(mod.quarter)
```



The residuals vs fitted looks similar, though the quantile-quantile plot is a little better. If I were desperate and really wanted to find a normalizing transformation, I'd probably go with this model. Though I would take the coefficient estimates and p-values with a grain of salt.

A two-part conditional model for semi-continuous data

A final option, and the one I actually used when analyzing this data, is to use a two-part conditional model. The logic is similar to the zero-inflated models we've discussed, though with an important difference. The logic works like this:

1) Make a model for whether the data is positive (presence) or zero (absence). Use a binomial GLM, and include any predictors of interest.

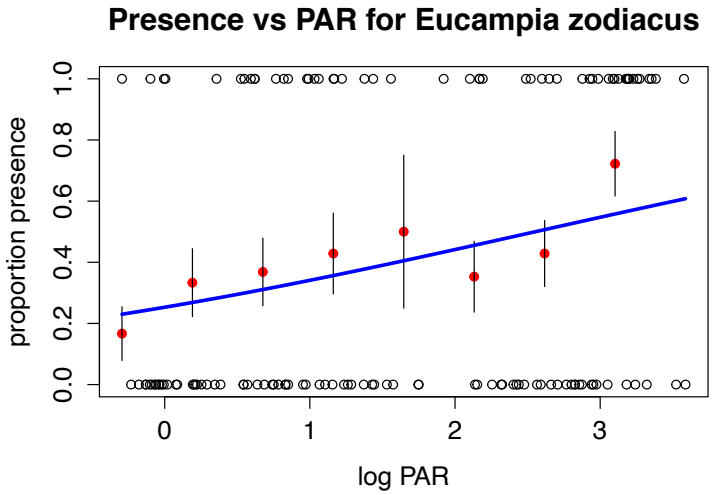
2) Make a second model that only analyzes the positive densities. Model these as lognormally distributed, i.e. log-transform and then use lm().

This approach is called conditional because model 2 is saying 'how do these predictors affect density, conditional on the fact that the species is present?'. This approach is also called a *hurdle model*, because model 1 is modeling whether the species crosses the hurdle from absence to presence.

Similar to a zero-inflated model, a two-part model allows for the fact that the zeros may be controlled by different processes than the positive densities. When there are many zeros, as in the *Eucampia* data, this is certainly plausible. The main contrast with a zero-inflated model is that the zeros in a zero-inflated model could come from the binomial 'extra' zeros, OR from the count distribution (Poisson, negative binomial). That's why the zero-inflated model is a kind of mixture model. The two-part model literally cleaves the data in two, which is why you can just fit two separate models to do the analysis. It is also possible to fit a two-part model (aka hurdle model) to discrete count data, though I think a zero-inflated model usually makes more sense.

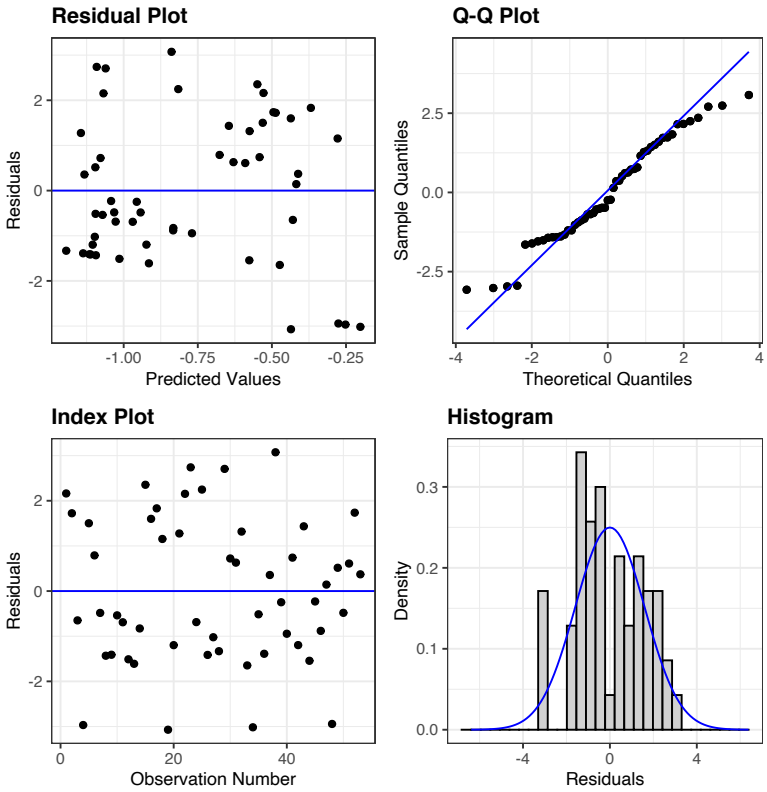
OK let's fit a two-part model to the *Eucampia* data. The first part (the binomial model) was fit in lectures 8 and 9, and looks like this:

```
mod = glm(presence ~ logPar, data = datause, family = binomial)
```

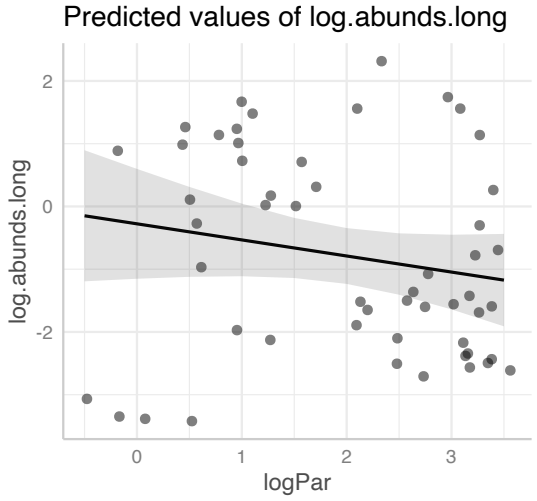


Now for the second part, the lognormal model:

```
datapos = subset(datause, presence > 0)
mod.pos = lm(log(abunds.long) ~ logPar, data = datapos)
resid_panel(mod.pos)
```

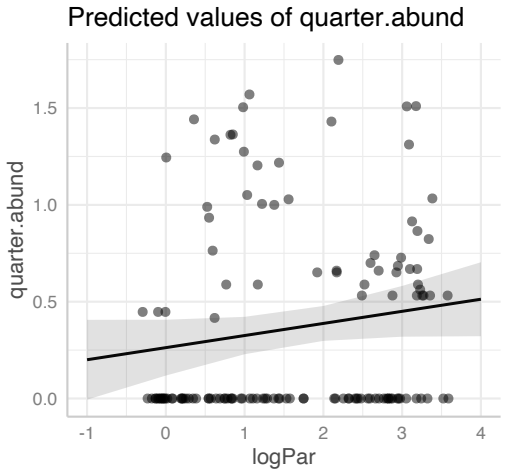


First I inspect the residuals, and they look much better than when we were fitting all the data together. The positive part of the data is approximately lognormal. What does the fitted relationship look like?



Looks like the positive part of the data doesn't show a relationship with logPar, and indeed if we do an F-test on the model it is nothing near significant.

What's the conclusion of this analysis? It appears that the presence of *Eucampia* increases substantially as irradiance increases, but once it is present its abundance is not affected by irradiance. Biologically this seems reasonable because once there is sufficient light in the mixed layer for a phytoplankton population to have positive growth, phytoplankton are more likely to be limited by nutrients or grazers than by irradiance. It's also worth noting that the quarter root model find a nonsignificant positive trend with logPar, consistent with the fact that the postive densities are showing a different pattern than the zeros:

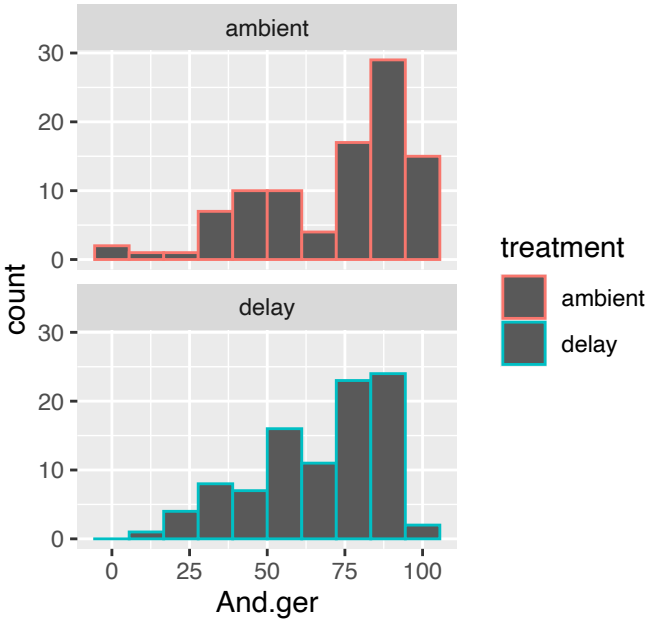


Modeling this kind of data in two parts is a double-edged sword. On the one hand, it is possible to discover different processes controlling presence/absence vs abundance when present, and the model assumptions are well fit by the data. On the other hand, if presence/absence and abundance are controlled by similar processes, then a two-part model may reduce our ability to detect patterns. The presence-absence model is removing information by converting abundances to presence; the lognormal model is also removing information by dropping the lowest values from the model, thereby reducing the range of the response and also using only half of the data. The best approach will depend on the circumstances and the data. If you really want to fit a single relationship, then something like the quarter-root transform, or a Tweedie or Tobit model is worth considering.

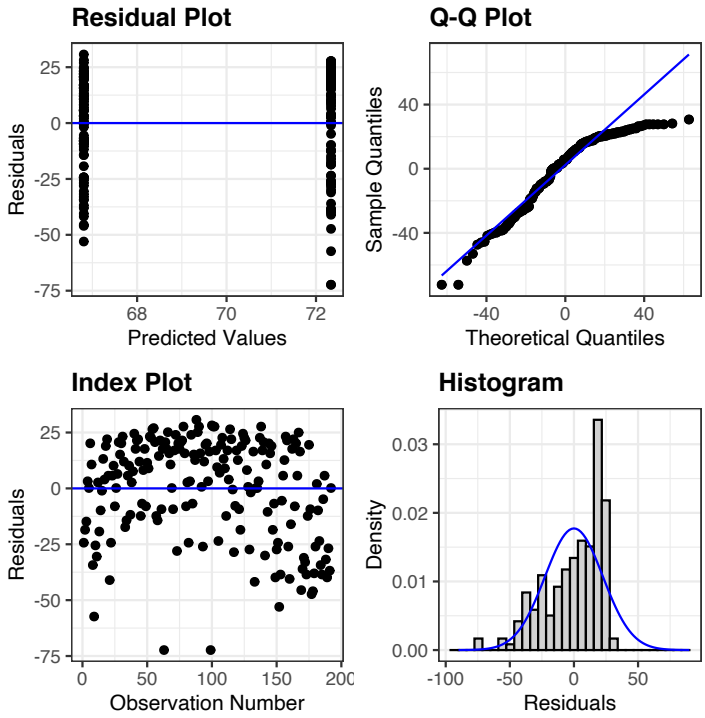
Proportions that aren't binomial

The last misfit data type I'll review is proportions that aren't binomial. When data has the form "X successes out of Y total", it can be modeled with the binomial distribution. However, there are cases where data is some kind of proportion or percentage that does not arise from some process of binomial trials. In biology common examples are percent cover in benthic marine systems or terrestrial plants; the percentage of something that is consumed by predators or lost due to disturbance.

For example, here are some data from a long-term rainfall manipulation experiment at Konza Prairie, where two treatments were imposed: rainout shelter, with collected rain added back to the plots; rainout shelter, with collected rain added back to make the delay between rain events 50% longer, while keeping the total rain over time constant. These are the 'ambient' and 'delay' treatments, and the percent cover of prairie plants was measured as the response. Let's look at the cover of one of the more common species, the tall grass *Andropogon gerardii*, aka big bluestem.

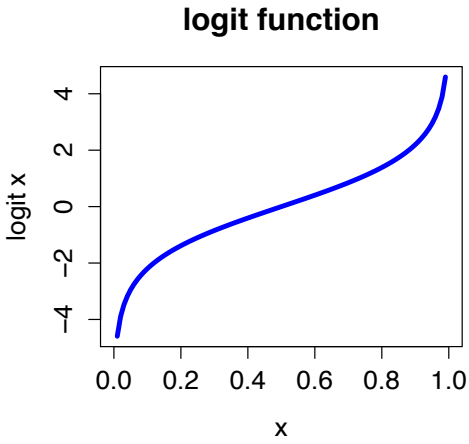


These are histograms of percent cover of big bluestrem in all samples of the two treatments. These data are from a time series, and so the time component should be accounted for (this is also the case with the English Channel data from before), but that will come later in the course. For now let's deal with modeling percent cover. It's pretty clear that using a normal distribution isn't a great idea, especially looking at the ambient treatment where the distribution is pushed up against 100% cover. Looking at the residuals for a normal model confirms this:



Most of the data isn't too bad, but there is a long tail on the quantile-quantile plot where the real data is too small compared to the theoretical expectation, and that is because the data is bounded at the upper end.

Is there a good normalizing transformation for this data? A common recommendation in textbooks is the arcsine square root transformation, but a paper by Warton and Hui called "The arcsine is asinine" has convinced me to avoid it. A simpler option is the logit transform. We already know about the logit from binomial GLMs. The formula is $\text{logit}(x) = \log\frac{x}{1-x}$. This will take a value between zero and one, and transform it so that it ranges from –Infinity to Infinity.

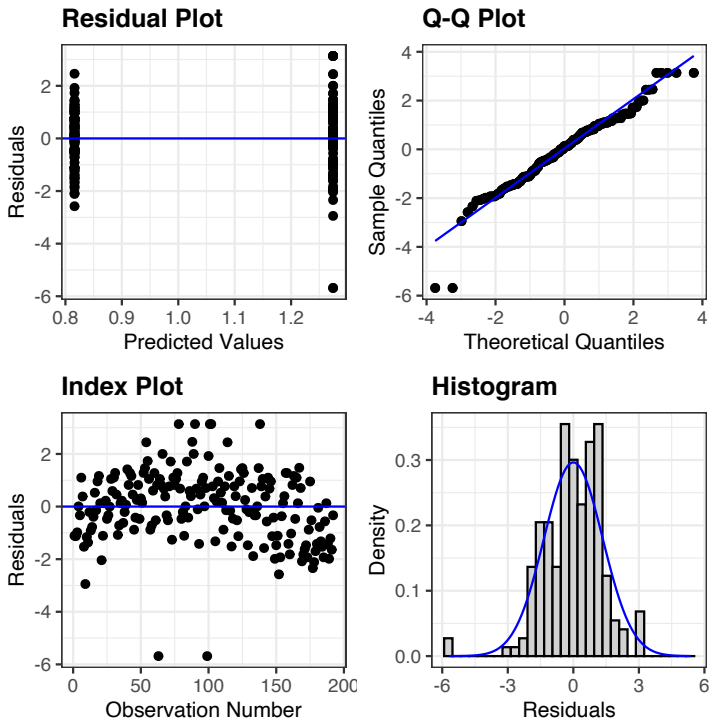


This function will also tend to 'stabilize' the variance, i.e. make it more homoscedastic, by stretching out the distances between values that are near 0 or 1. The only problem is that the logit is undefined when the data equals zero or one. The typical solution is to rescale the data by pushing the zeros away from 0 and the ones away from 1, by small amounts. It's often a good idea to see if your results are sensitive to the value used. In the big bluestem example, the value closest to either 0% or 100%, without actually hitting those boundaries, is 98.8%. So I will convert any 1.0 values to 0.988, and convert any 0 values to 0.012, and all the values in between will be squeezed proportionally. Let's look at a model fit to the logit transformed data:

```
konza$And.ger.logit = logit(konza$And.ger/100, adjust = 0.012)
```

The handy logit() function in the 'car' package lets you specify the amount you want to adjust 0's and 1's. Note that I've divided the percentage cover by 100 to get values between 0 and 1.

```
mod.logit = lm(And.ger.logit ~ treatment, data = konza)
resid_panel(mod.norm)
```



Based on the residual plots, the transformation looks quite good. What does the fitted model say?

```
summary(mod.logit)
```

```
##
## Call:
## lm(formula = And.ger.logit ~ treatment, data = konza)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -5.685 -0.879  0.126  0.953  3.136
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)       1.274      0.138    9.25   <2e-16 ***
## treatmentdelay   -0.458      0.195   -2.35     0.02 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.35 on 190 degrees of freedom
## Multiple R-squared:  0.0283, Adjusted R-squared:  0.0232
## F-statistic: 5.54 on 1 and 190 DF,  p-value: 0.0196
```

The treatment effect says that the delayed rainfall reduces percent cover of big bluestem by -0.458 on the logit scale. Let's convert this back to the raw data scale. The inverse of the logit is the logistic function. That means the fitted treatment mean for the ambient treatment is exp(1.27)/(1 + exp(1.27)) = 0.78, and the mean for the delay treatment is exp(1.27-0.46)/(1 + exp(1.27-0.46)) = 0.69. So the delay treatment reduces percent cover by about 9%, which is not huge but not nothing. An F-test says this difference is significant:

```
Anova(mod.logit)
```

```
## Anova Table (Type II tests)
##
## Response: And.ger.logit
##           Sum Sq  Df F value Pr(>F)
## treatment     10   1    5.54   0.02 *
## Residuals    346 190
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

It's worth noting that if this data were analyzed with a normal distribution, the difference between the treatments is estimated to be ~6%, and is not significant:

```
mod.norm = lm(And.ger ~ treatment, data = konza)
```

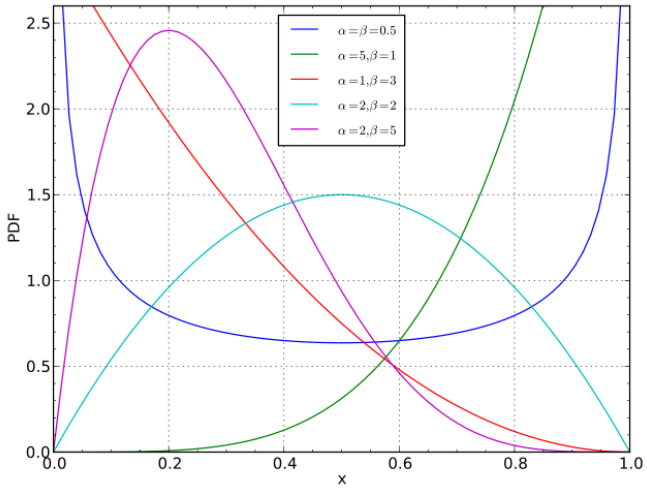
```
Anova(mod.norm)
```

```
## Anova Table (Type II tests)
##
## Response: And.ger
##           Sum Sq  Df F value Pr(>F)
## treatment   1467   1    2.88  0.091 .
```

```
## Residuals  96704 190
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

So if you put a lot of stock in the p<0.05 threshold (you shouldn't, but people do anyways), then this shows how using an appropriate transformation can be important.

So the logit transformation seems alright for non-binomial proportion data. Is there no probability distribution that can actually model values between 0 and 1. In fact there is, and it is called the beta distribution. It has two parameters and can take on a variety of shapes:



There is an R package called 'betareg' which will let you model proportional data with a beta distribution, where the mean of the distribution is modeled with a logit link as in a binomial GLM. glmmTMB can do the same. I won't cover it here, but it is worth considering as an alternative to the logit transformation if you have to deal with proportional data.

Table of data types

Phew, we're done covering different kinds of distributions and how to model them. On to new topics. Here is a table summarizing what we've covered:

Data type	**Distribution or transformation**
Continuous, unbounded	Normal distribution
Counts	Poisson, negative binomial, quasi-Poisson, zero-inflated P or NB
Binary or proportion (X out of Y)	Binomial, quasi-binomial
Continuous, >0	Lognormal, gamma

Continuous, ≥0	Two-part model (hurdle); quarter root or log transform; Tweedie distribution; Tobit model
Non-binomial proportions	Logit transform; beta regression