FES 524 Winter 2022 Lab 8

Contents

Generalized linear models for count data

In Lab 8 we'll continue to work with generalized linear models, this time for count data. For the first time this quarter we'll be working with continuous explanatory variables.

Our response variable this week is the number of colonizing insect species on mangrove islands after all insects were previously eradicated. The question of interest is how the mean number of colonizing insect species is related to island size (m^2), after accounting for island distance from shore (km).

Load R packages for data manipulation and plotting

```
library(dplyr)
library(ggplot2)
```

Read in dataset

We will be working with the dataset `lab8.example.data.csv`, so make sure you have this file and have changed your working directory appropriately. As usual when we read in a dataset we'll take a look at the structure and make any necessary changes. Notice this week all variables in the dataset are numeric or integers.

```
mangrove = read.csv("lab8.example.data.csv")

head(mangrove) # Look at the first 6 lines of the dataset
```

```
  numspp size  dist
1     41  102 0.051
2     59   97 0.028
3     13   54 0.104
4      3   32 0.036
5     91  113 0.055
6     13   82 0.160
```

Look at the structure of the dataset in the Environment pane.

```
'data.frame':   25 obs. of  3 variables:
 $ numspp: int  41 59 13 3 91 13 5 14 55 6 ...
 $ size  : int  102 97 54 32 113 82 49 67 109 34 ...
 $ dist  : num  0.051 0.028 0.104 0.036 0.055 0.16 0.199 0.179 0.056 0.09 ...
```

Initial data exploration

Summary statistics with skimr

Getting summary statistics of all variables in a dataset can be useful, especially for continuous variables. We've used `summary()` in the past, which works OK for small datasets, but the output isn't particularly tidy. Today we'll use package **skimr** to do something similar with a tidier output. You may need to install this package if you haven't used it before.

```
library(skimr)
```

Once **skimr** is loaded we can use `skim()` to get a quick look at the variables in the dataset. We get a few more summary statistics than from `summary()`, including the standard deviation, but also get more readable information on the number of rows missing and a simple histogram showing the distribution of each variable. (The histograms do not show in this PDF document.)

```
skim(mangrove)
```

```
-- Data Summary ------------------------
                           Values
Name                       mangrove
Number of rows             25
Number of columns          3
-----------------------
```

```
Column type frequency:
  numeric                 3
------------------------
Group variables           None

-- Variable type: numeric -------------------------------------------------------------------------
# A tibble: 3 x 10
  skim_variable n_missing complete_rate   mean      sd     p0    p25    p50    p75  p100
* <chr>             <int>         <dbl>  <dbl>   <dbl>  <dbl>  <dbl>  <dbl>  <dbl> <dbl>
1 numspp                0             1 32.7   30.3     2     10     17      49    118
2 size                  0             1 76.4   27.0    32     54     80     102    117
3 dist                  0             1  0.106  0.0580  0.028  0.055  0.104   0.16   0.2
```

If you are using R Markdown instead of a script the output from **skimr** is put into a nicely formatted table. However, if working on windows you may have problems with histograms so skip them using `skim_without_charts()`. Here's an example of what the output looks like in PDF R Markdown output.

```
skim_without_charts(mangrove)
```

Table 1: Data summary

Name	mangrove
Number of rows	25
Number of columns	3
Column type frequency:	
numeric	3
Group variables	None

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100
numspp	0	1	32.72	30.31	2.00	10.00	17.0	49.00	118.0
size	0	1	76.36	26.98	32.00	54.00	80.0	102.00	117.0
dist	0	1	0.11	0.06	0.03	0.06	0.1	0.16	0.2

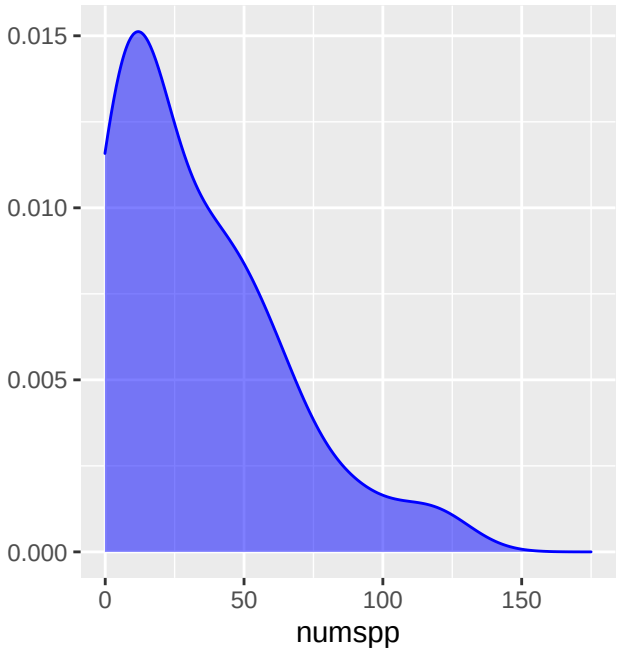
Graphical data exploration of the response variable

We'll start our graphical data exploration by looking at the overall distribution of the response variable via a density plot. We can see the shape of the distribution as well as possible values that the counts take.

This plot doesn't tell us all that much about what sort of model we could use, but it does demonstrate what observed count distributions can look like. Counts are non-negative (0 or greater) and typically show right skew.

Remember from lecture that we think the response variable follows some distribution *conditional on the variables in the model*. This isn't something we can easily explore. Always be careful with making guesses on distributions only by looking at the raw response variable.

```
# Density plot, changing color and fill and alpha (transparency)
qplot(x = numspp, data = mangrove,
     color = I("blue"),
     fill = I("blue"),
     alpha = I(.5),
     geom = "density",
     xlim = c(-.1, 175) )
```

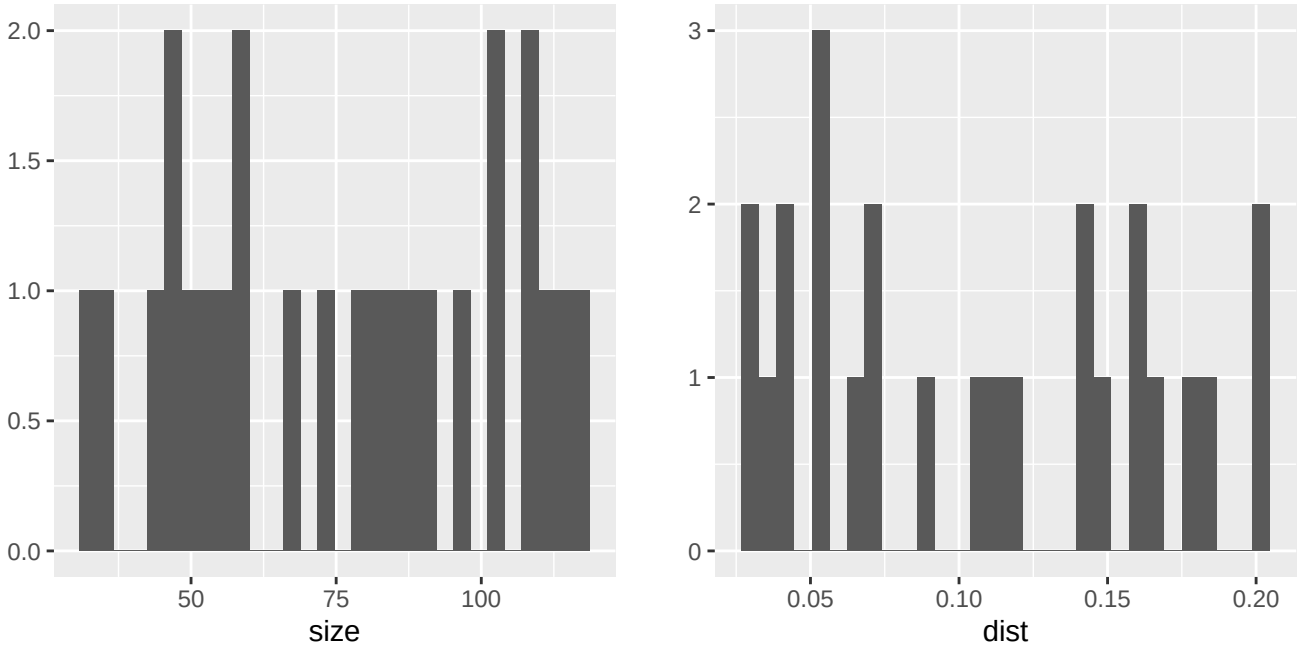


Graphical data exploration of the explanatory variables

We will also take a look at the distribution of the explanatory variables. We are looking at the range as well as the shape of the distributions. This helps us think about how practically meaningful the variable might be. A variable that takes on very few values, for example, or has a very narrow range might not be useful for modeling our response even if the two variables should be related.

In problems where we have many variables, going through them and thinking hard about ones that obviously don't cover a range that would be practically meaningful is an important step. Exploring them via boxplots and/or histograms is therefore warranted.

```
qplot(x = size, data = mangrove, geom = "histogram")
qplot(x = dist, data = mangrove, geom = "histogram")
```



Data exploration on the original scale of the response variable

With multiple continuous explanatory variables, we'll want to make plots of each variable against the response variable as well against each other. We'd also want to calculate pairwise correlations for explanatory variables. One option, which we won't review, is to use the built-in R function `pairs()` for plotting and `cor()` for calculating correlations.

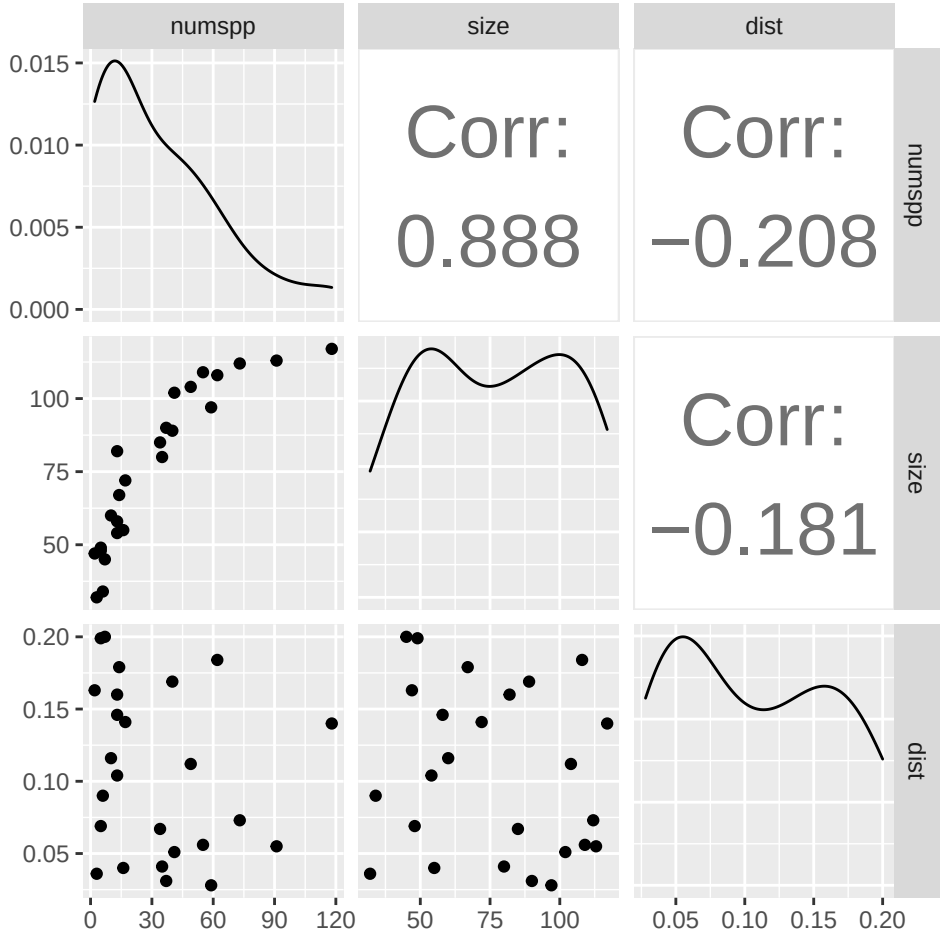
Combined scatterplot matrix and correlations via `ggpairs()`

Package **GGally** has a scatterplot matrix function called `ggpairs()` that is useful when exploring continuous variables. While it is very slow when you have a large number of variables, it gives pairs plots, pairwise correlations, and density plots of each variable in a single figure. This allows a lot of data exploration at one time.

In the example below, I make the size of the correlation text larger for increased visibility in the output plot with the `upper` argument. I also remove the default stars, since it is silly to do hypothesis tests of correlations without checking any assumptions.

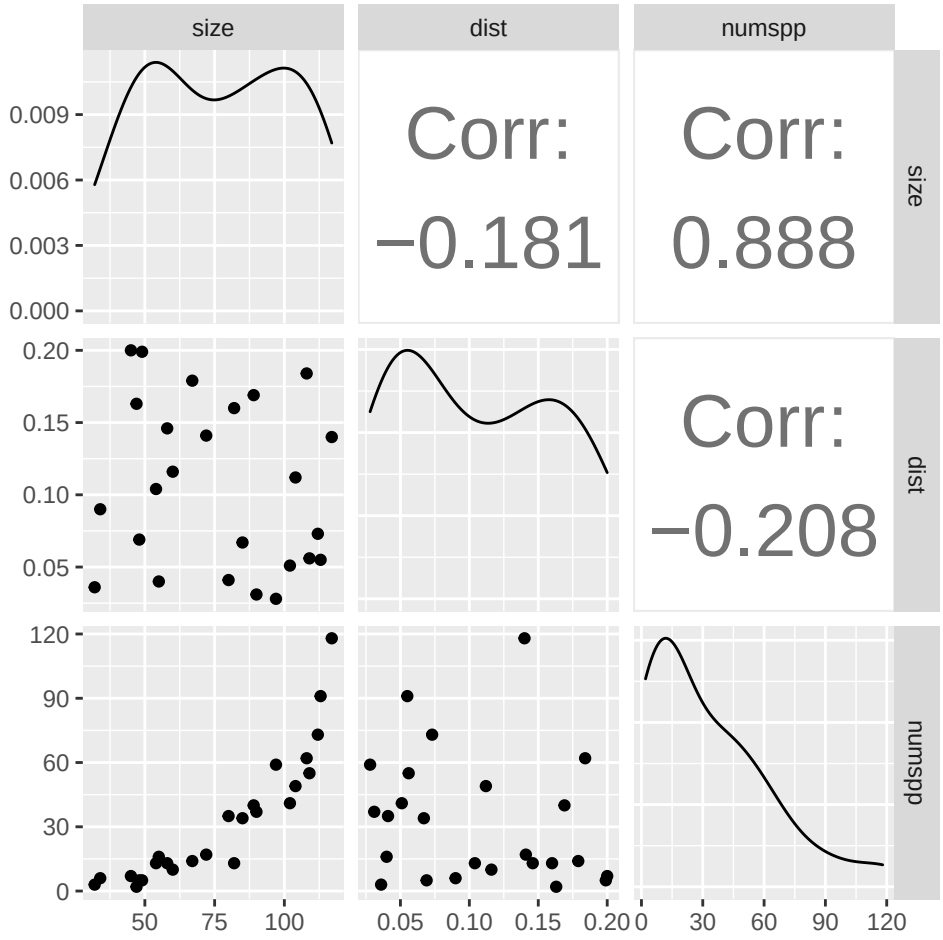
```
library(GGally)

ggpairs(mangrove,
       upper = list(continuous = wrap("cor",
                                size = 10,
                                stars = FALSE) ) )
```



When we pass a dataset to `ggpairs()` the resulting plot is in the same order as the variables are in the dataset. I find it easiest to interpret things when the response variable is on the y axis. It would help me to have `numspp` on the y axis in the lower triangle of plots. We can use `select()` from package **dplyr** to get the variables in the order we want for plotting (putting `numspp` last).

```
ggpairs( dplyr::select(mangrove, size, dist, numspp),
        upper = list(continuous = wrap("cor",
                                       size = 10,
                                       stars = FALSE) ) )
```



Data exploration on the link scale for the response variable

Now we want to explore the dataset on the scale of the link. For count data, the most common link is the natural logarithm link.

When working with generalized linear models we work directly with the original data and don't use any transformations of the response variable. For data exploration purposes, though, we need to transform the response variable. Remember, this is *only for the purposes of data exploration*, so be careful to always go back to the raw data when you start the generalized linear model analysis.

The empirical natural logarithm of number of species

In order to make a variable on the log scale for data exploration, we'll need to check for and deal with any 0 values prior to transformation.

Do we have any 0 values in the dataset?

```
min(mangrove$numspp)
```

```
[1] 2
```

In this case, no, so the log transformation is going to be easy. However, if we did have 0 values we'd need to add a small value to the entire distribution in order to shift it away from 0. One commonly used rule is to add half of the smallest non-zero value to the entire dataset. Here's how we would check what the smallest non-zero value was in the dataset.

```
min(mangrove$numspp[mangrove$numspp > 0])
```

```
[1] 2
```

We can make our new variable via `mutate()`. Like last week, I add the letters "el" to the new variable name to remind myself that this is the empirical log of the response variable that I will be using only for exploratory graphics.

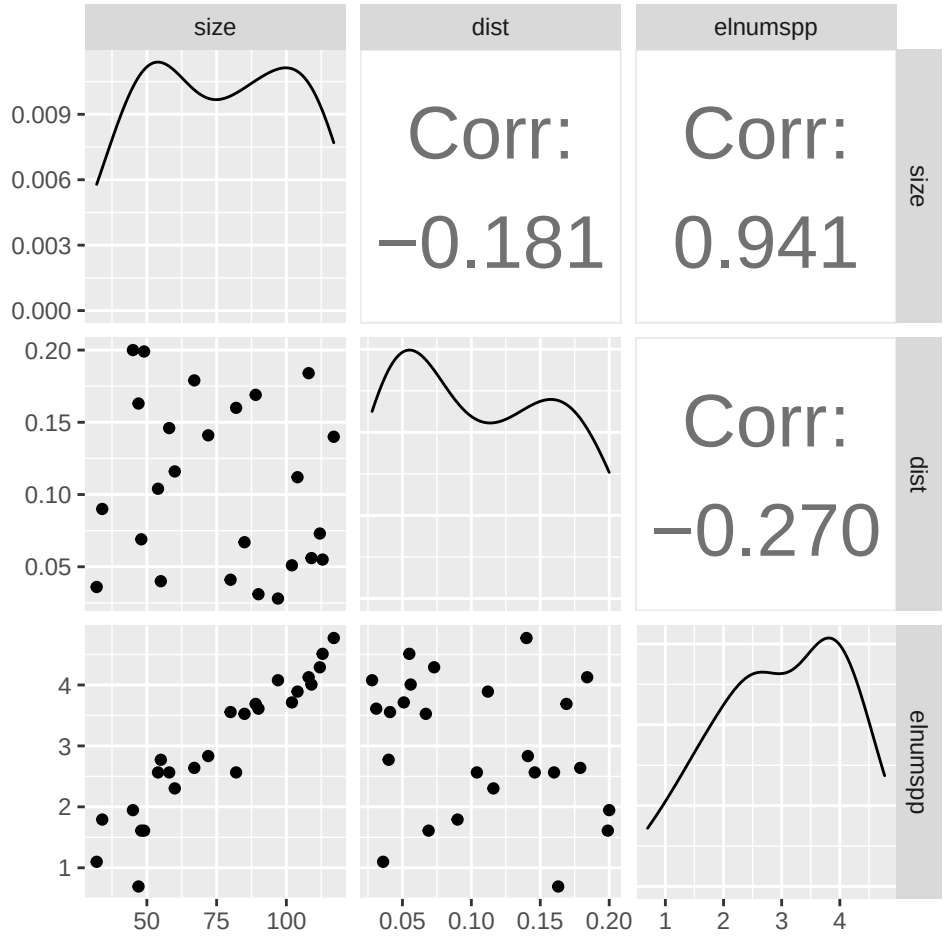
If we had 0 values in the dataset we'd log the response variable only after adding half the smallest non-zero value to it. For example, if our minimum no-zero value was 1 we'd do `log(numspp + 0.5)`.

```
mangrove = mutate(mangrove, elnumspp = log(numspp) )
```

Pairs plot on the log scale for the response

Now we can make the pairs plot with `numspp` on the log scale. Don't forget that we assume a linear relationship between the response and explanatory variables on the scale of the link. These plots can help us think about this assumption of linearity; however, since these are exploratory plots we should be cautious in basing any modeling decisions on what we see if doing confirmatory research.

```
ggpairs( dplyr::select(mangrove, size, dist, elnumspp),
        upper = list(continuous = wrap("cor",
                                       size = 10,
                                       stars = FALSE) ) )
```



Generalized linear models for count data

Count data are discrete and integer-valued (including 0). There are only have a couple of distributions that are regularly used for modeling count data in ecology, the negative binomial distribution and the Poisson distribution. The generalized Poisson distribution is also getting to be more popular.

The negative binomial distribution

The negative binomial distribution has become the common "go-to" distribution for analyzing count data in ecology. See, e.g, Stroup's 2014 paper "Rethinking the analysis of non-normal data in plant and soil science" for more background.

The negative binomial distribution has discrete, integer values that are non-negative (so include 0). It can be extremely right skewed, so it is useful for modeling situations where there are many observations of small counts and a few observations of very large counts. This is the kind of distribution that would make sense for an organism that tend to be clustered together.

Parameters of the negative binomial distribution

This distribution as two parameters, λ and θ. The variance and mean of the distribution are related, as they both rely on λ. The variance increases as the mean increases.

The mean of the distribution is λ.

The variance is $\lambda + \lambda^2/\theta$; $\theta \geq 0$.

The Poisson distribution

The Poisson distribution was the original distribution we had available for modeling counts in generalized linear models.

The Poisson distribution has discrete, integer values that are non-negative (so include 0). It is often right skewed, but not extremely so.

Parameters of the Poisson distribution

The Poisson distribution has a single parameter, λ. The variance and mean of the distribution are both defined by λ, so the variance increases as the mean increases.

The mean of the distribution is λ.

The variance is also λ.

Relationship between the negative binomial and Poisson distributions

Note that as θ goes to infinity, the negative binomial variance becomes identical to the Poisson variance.

This variance-mean relationship of the Poisson distribution tends to be overly strict for a lot of ecological count data. The variance of the Poisson distribution almost invariably underestimates the variance of the data, leading to *overdispersion*. See week 7 lecture and lab material for more discussion on overdispersion. Starting out by modeling count data via a negative binomial distribution is therefore a reasonable strategy. An estimate of θ that is extremely large (>1000) would indicate that the Poisson variance-mean relationship could be sufficient for the particular data at hand.

The assumption of constant variance

As you can see from the distributions above, we expect the variance to depend on the mean when modeling count data. We no longer have the assumption of constant variance. The fact that for counts the variance usually depends on the mean is one of the reasons that trying to use linear models (i.e., assuming normality of errors) for count data tends to fail.

Package messages and masking

To fit a negative binomial generalized linear model in R we'll need the function `glm.nb()` from package **MASS**.

When you load **MASS**, you will get a series of messages. One of those messages is that **MASS** has a `select()` function that has *masked* the `select()` function from **dplyr**. In order to continue to use `select()` from **dplyr** you can either unload **MASS** or you can call the function explicitly via `dplyr::select()`.

If loading the packages all together at the start of your script you may decide to load **dplyr** *after* **MASS** to avoid this `select()` problem. If you've already loaded these packages you must unload them before you can reset the package loading order.

The take home here is that the order you load packages can matter, and those messages you get when loading packages can be very important.

```
library(MASS)
```

```
Attaching package: 'MASS'

The following object is masked from 'package:dplyr':

    select
```

Fitting a negative binomial generalized linear model

Take a look at the help page to see how the `glm.nb()` function works, including the defaults for the link function.

```
?glm.nb
```

The default link function is the `log` link, which is what we'll be using today. Alternatives are listed under "Arguments". Remember to work with the original raw data as we don't use transformed response variables when working with generalized linear models.

Our research question today is about the relationship between mean number of colonizing insect species and island size, after accounting for the effect of island distance from shore. Our model will use `numspp` as the response variable and include both `size` and `dist` as explanatory variables. This is our fullest model.

```
mod1nb = glm.nb(numspp ~ size + dist, data = mangrove)
```

Checking for model fit (negative binomial)

Checking variance-mean relationship

We'll start by checking that the variance-mean relationship defined by the negative binomial distribution adequately captures the variance of the data. Like last week, we do this by summing the squared Pearson residuals and dividing by the residual degrees of freedom.

Values close to 1, like the estimate below, indicates no problem. A large value would indicate a problem with the model. We'd need an alternative modeling strategy since the negative binomial model wouldn't fit our data well and so wouldn't be valid.

```
sum( residuals(mod1nb, type="pearson")^2 )/mod1nb$df.res
```

```
[1] 1.068099
```

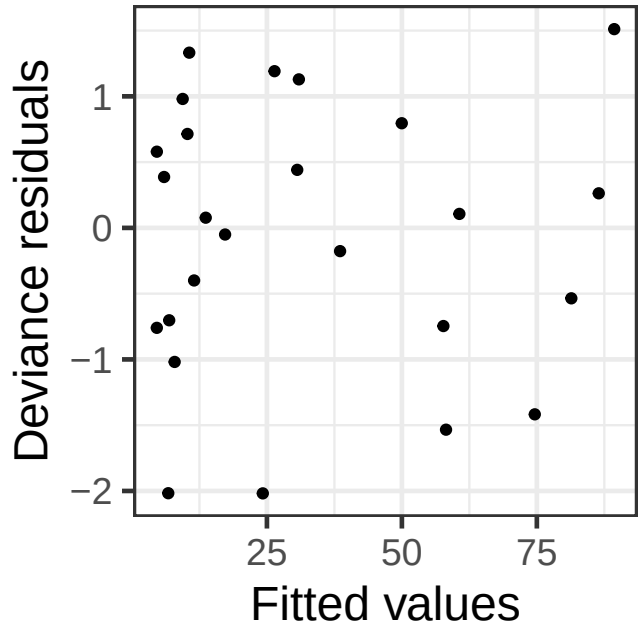
Residual plots

We have no evidence of problems with the variance-mean relationship, so let's look at the standard residual vs fitted value plots to check for any odd patterns. This is where we might find, e.g., evidence of non-linearity. I've seen other patterns from complicated negative binomial generalized linear mixed models that indicated the variance was *decreasing* with the mean. This ultimately led me to reject the model fit. Seeing patterns and deciding if they are OK or not is not an easy task. If you are working on these kinds of models in your own work, package **DHARMa** can be very useful for checking model fit.

We'll be using deviance residuals today, as they can help us spot unusual points. We expect about 95% of the residuals to be between +/-2.

Nothing stands out in the plots below, and it appears the model fits these data adequately.

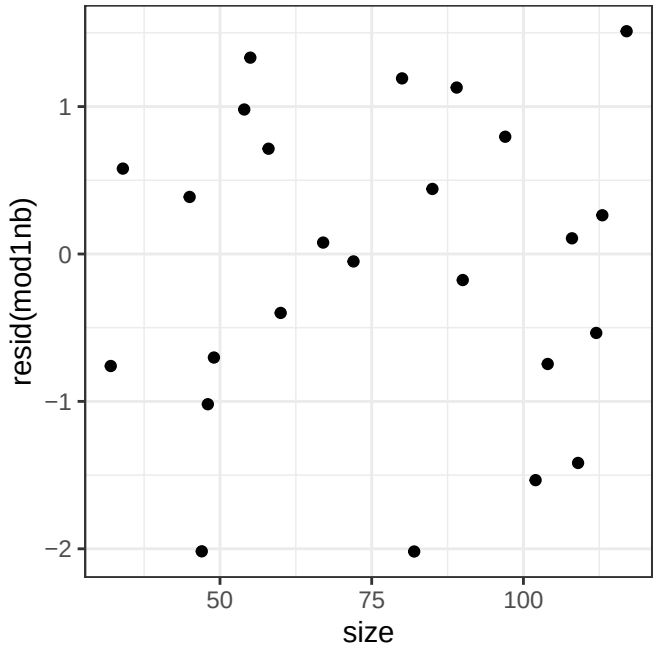
```
qplot(x = fitted(mod1nb), y = resid(mod1nb) ) +
    theme_bw(base_size = 18) +
    # change base size of text so is bigger
    labs(y = "Deviance residuals",
        x = "Fitted values")
```

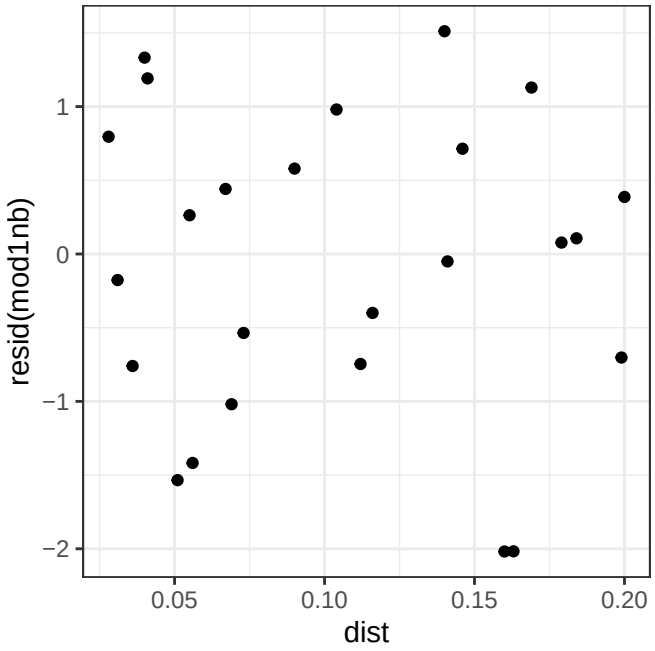


Don't forget to look at residual vs explanatory variable plots. This is another place where we could spot unusual patterns caused by an explanatory variable (including non-linear relationships). I see nothing interesting to note here.

```
# Size vs deviance residuals
qplot(x = size, y = resid(mod1nb), data = mangrove) +
    theme_bw()

# Distance vs deviance residuals
qplot(x = dist, y = resid(mod1nb), data = mangrove) +
    theme_bw()
```





Normality and non-Gaussian generalized linear models

As we discussed in Lab 7, there is no assumption of normality in non-Gaussian generalized linear models. The deviance residuals *could* be normally distributed in certain situations and you may see folks making quantile-quantile normal plots of the deviance residuals. However, this is not an actual assumption we are making with this model and not seeing normality in

the deviance residuals does not necessarily indicate a problem.

How to fit a Poisson generalized linear model

Before we wrap up our analysis, let's take a moment to learn how to fit a Poisson generalized linear model in R. We do this with the `glm()` function, defining the family and link. See `?family` for a list of possible families and links.

```
mod1p = glm(numspp ~ size + dist, data = mangrove,
            family = poisson(link = "log") )
```

Checking model fit (Poisson)

Checking for overdispersion

Since the Poisson is a single parameter distribution we most often expect to have overdispersion. We check for overdispersion using the sum of the squared Pearson residuals divided by the residual degrees of freedom. We get an estimate of ~2, indicating there is overdispersion.

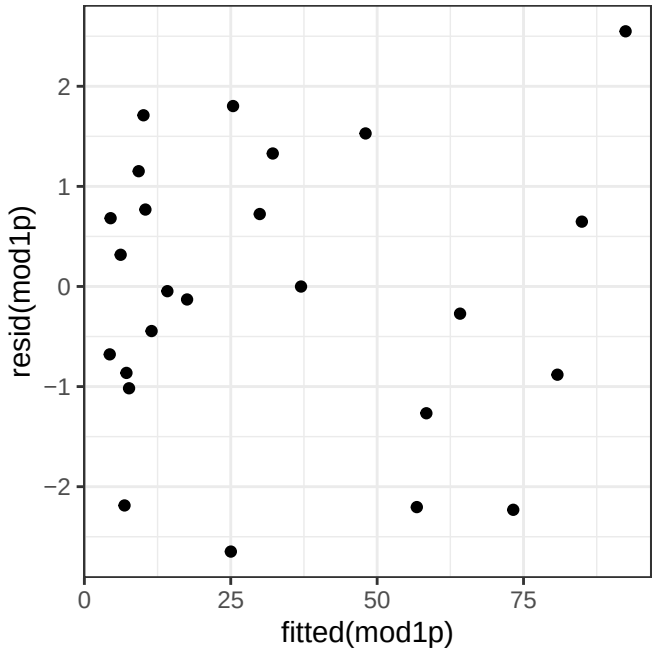
```
sum( residuals(mod1p, type="pearson")^2 )/mod1p$df.res
```

```
[1] 2.043972
```

Residual plots

We'll focus on the residual vs fitted values plot today, as this section is just to see a basic example of a Poisson GLM. You should notice that we have 5 points that are outside +/-2 in the residual vs fitted plot below. Having many points outside of +/-2 when plotting deviance residuals is another possible indication of overdispersion. I see no other patterns of note.

```
qplot(x = fitted(mod1p), y = resid(mod1p) ) +
    theme_bw()
```



Alternatives to the Poisson generalized linear model with overdispersion

The negative binomial is, of course one alternative to using the Poisson distribution for modeling count data. There are some other options, as well.

Observation-level random effect

We could fit what's called a Poisson-log normal model by adding in an observation-level random effect. We could never do this with Gaussian linear models because the observation-level random effect is the residual error term. When working with a

one parameter distribution, where the variance is solely a function of the mean, we can add observation-level random effects. This would put us into the world of generalized linear *mixed* models.

Generalized Poisson

We could use the *generalized Poisson* distribution. This distribution allows for a different variance-mean relationship than the negative binomial and data from this distribution can be heavier tailed (so can have more 0 values as well as larger counts compared with negative binomial). The generalized Poisson can also address problems with underdispersion. You can fit generalized Poisson GLM's in package **VGAM** (GLM's) or **glmmTMB** (GLM's or GLMM's).

Quasipoisson

We could fit a quasilikelihood model by using `family = quasipoisson`. This simply multiplies all standard errors from the Poisson generalized linear model by the square root of the estimated dispersion. This can be simple and useful, although we no longer have the standard likelihood tools available to us. Traditionally, we do t and F tests when using quasilikelihood distributions instead of the standard z and χ^2 tests.

Reporting model results

Let's go back to our negative binomial model, `mod1nb`, to finish up the analysis.

Likelihood ratio test

We'll start by doing a likelihood ratio test (aka drop-in-deviance test) for evidence against the null hypothesis that the slope between log(mean number of colonizing insect species) and island size is 0. We can do this by comparing our full model and a reduced model without `size` in it.

First we need to fit the reduced model. I will do this via `update()`, which we saw in lab 6, so you can see what it looks to remove a variable from the model.

This model now only has `dist` in it.

```
mod1nb_red = update(mod1nb, . ~ . - size)
mod1nb_red
```

```
Call:  glm.nb(formula = numspp ~ dist, data = mangrove, init.theta = 1.246146061,
    link = log)

Coefficients:
(Intercept)         dist
      3.861       -3.710

Degrees of Freedom: 24 Total (i.e. Null);  23 Residual
Null Deviance:      28.88
Residual Deviance: 27.68    AIC: 229.5
```

We can now do the statistical test via `anova()`. The resulting likelihood ratio test is based on the Chi-squared distribution.

As usual, you need to report the test statistic and the degrees of freedom for test if you report any p-values. In this case you can see from the output this is a 1 degree of freedom test. Be sure to use the "chi" symbol: $\chi^2_1 = 63.3, p < 0.0001$.

```
anova(mod1nb, mod1nb_red)
```

```
Likelihood ratio tests of Negative Binomial Models

Response: numspp
        Model     theta Resid. df    2 x log-lik.   Test    df LR stat.      Pr(Chi)
1        dist  1.246146        23       -223.5495
2 size + dist 36.952920        22       -160.2724 1 vs 2     1 63.27715 1.776357e-15
```

Another option for doing likelihood ratio tests is by using the `drop1()` function. However, this *does not work correctly* for negative binomial models in R and so the full-reduced model approach is the correct one to use.

Note also that you have to be careful using `anova()` on a single model when using continuous variable. `anova.glm` uses *sequential* tests, so changing the order of the variables in the model changes the results. Only the last variable listed is a test after accounting for everything else in the model.

Finally, the output from `summary()` shows marginal tests instead of sequential tests (which is what we want) but these are Wald tests and likelihood ratio tests are preferred.

Since we did not have any research questions about `dist` it is not relevant to report any hypothesis tests for this variable.

Estimates and confidence intervals

Today we are making inference about an estimated relationship between our response variable and a continuous explanatory variable. When estimating slopes with `glm()` we can pull out the coefficient of interest using the `coef()` function, and get confidence intervals via `confint()`. These results are on the scale of the link (i.e., here are on the log scale).

The `confint.glm` function makes confidence intervals by profiling the likelihood by default, which are the gold standard for confidence intervals in generalized linear models.

```
coef(mod1nb)
```

```
(Intercept)        size        dist
 0.40730377  0.03651228 -1.33460082
```

```
confint(mod1nb)
```

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

```
                 2.5 %     97.5 %
(Intercept) -0.08712399 0.88638660
size         0.03202136 0.04114217
dist        -3.23816405 0.56828551
```

Biologically relevant change in X

The slope coefficient for island size, which is the one we are interested in, is for a 1 m^2 change in island size by default. However, a change in island size that small is not considered large enough to lead to a meaningful change in the mean number of colonizing insect species. It turns out a 1 m^2 change isn't even detectable given the precision of island size measurements. Researchers defined a 5 m^2 change in island size to be a biologically relevant change in island size.

To make inference at the correct scale of island size, we'll need to multiply the estimate and confidence interval limits by the biologically relevant unit change. We do this on the scale of the model.

We're only interested in the coefficient for island size, so we'll pull that one out and multiply it by 5.

```
coef(mod1nb)[2]*5
```

```
     size
0.1825614
```

```
confint(mod1nb)[2, ]*5
```

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

```
    2.5 %    97.5 %
0.1601068 0.2057108
```

Exponentiating back to original scale for interpretation

Now that we have the coefficient based on a biologically relevant change in island size, we'll need to use the inverse link, `exp()`, to exponentiate back to the scale of the original data.

```
exp( coef(mod1nb)[2]*5 )
```

```
    size
1.200288
```

```
exp( confint(mod1nb)[2, ]*5 )
```

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

```
   2.5 %   97.5 %
1.173636 1.228398
```

Interpreting the estimated slopes

The estimated relationship between mean number of colonizing insect species and island size is multiplicative, as the relationship we modeled is on the log scale. The results can be expressed either as X-percent increases (or decreases, if relevant) or X-fold changes.

Here area a couple of options for language for reporting the results for our generalized linear model with a log link:

1. After accounting for distance of island from shore, the evidence against the null hypothesis that there is no relationship between island size and mean number of colonizing insect species is $\chi^2_1 = 63.3, p < 0.0001$. For every 5 square meter increase in island size the mean number of insect species colonizing an island increases by 20.0% (95% CI 17.4% to 22.8%).
2. After accounting for distance of island from shore, associated with a 5 square meter increase in island size is a 1.20-fold increase in the mean number of insect species colonizing an island (95% CI 1.17 to 1.23).

While I won't plot the result and confidence interval today, I would still interpret this result based on the biologically important value. Researchers expected at least a 20% increase in mean number of colonizing insect species for a 5 square meter increase in island size.

Plotting the results with an added variable plot

Visualizing the results from a model with continuous explanatory variables is pretty different than what we've been doing throughout the term. One option that I often use is to show the estimated relationship between the response and the explanatory variable of interest while holding the other variables fixed. This type of plot is called an *added variable* or *partial regression* plot.

To make such a plot we'll need to get predicted values from the model and build approximate 95% confidence intervals of those predictions. We'll be doing this "by hand", but see also packages **visreg**, **effects** and **rms**.

Today we will be plotting the estimated relationship as a line with confidence intervals as a ribbon. The observed values for the explanatory variable are added via a rug plot, which are lines along the axis that indicate locations of observed values. This allows us to see where we had data to estimate the line. Using rug plots avoids the confusion of trying to draw a line estimated from a model with multiple variables in it on top of the raw data when the raw data hasn't been adjusted for all the variables in the model.

Making a dataset for predictions

We need to get predicted values from the model *with the other variable held fixed*. It is most common to fix the other variable(s) to the median or mean. It is important to be clear about what value the other variables in your model are fixed to, which I put in the caption of the graphic.

The dataset we will use for prediction will have the original data for `size` but `dist` fixed at its median.

Here's the median of `dist` in the original dataset.

```
median(mangrove$dist)
```

```
[1] 0.104
```

The dataset for prediction must contain all explanatory variables used in the model *with the exact same names*. If you forget to include a variable or use different names for the variables in the prediction dataset you will get errors.

I'll name the dataset for prediction `preddat`.

```
preddat = data.frame(size = mangrove$size,
                     dist = median(mangrove$dist) )

head(preddat)
```

```
  size  dist
1  102 0.104
2   97 0.104
3   54 0.104
4   32 0.104
5  113 0.104
6   82 0.104
```

Predictions from `glm()` objects

We use the `predict()` function to make predictions from our model. The `predict()` function works differently with different model types, so let's take a look at the help page for `predict.glm`. The `newdata` and `se.fit` arguments are particularly useful.

```
?predict.glm
```

By default the predictions will be on the model scale (on the scale of the link). That's appropriate for us because we will need to calculate approximate confidence intervals on the model scale. We'll then exponentiate the estimates and confidence interval limits back to the original scale.

We are going to use the dataset we made above for making predicted values. By default we get fitted values from the model using the original dataset, so to make predictions using an alternative dataset we make use of the `newdata` argument.

We can get standard errors from `predict.glm`. These are what we'll use for building confidence intervals, so we'll save these along with the predictions via `se.fit = TRUE`.

```
predsize = predict(mod1nb,
                   newdata = preddat,
                   se.fit = TRUE)
```

Calculate approximate confidence intervals

We can make approximate 95% confidence intervals by adding/subtracting 2 times the standard error from each predicted value. I use `with()` to avoid lots of dollar sign notation.

```
uppersize = with(predsize, fit + 2*se.fit)
lowersize = with(predsize, fit - 2*se.fit)
```

Everything above is on the link scale. We'll exponentiate and add these to the original dataset for plotting. This works because `preddat` is the same length and in the same order as `mangrove`. Check for these changes to the dataset in the Environment pane.

```
mangrove = mutate(mangrove,
                  predsize = exp(predsize$fit),
                  uppersize = exp(uppersize),
                  lowersize = exp(lowersize))

head(mangrove)
```

```
  numspp size  dist elnumspp  predsize uppersize lowersize
1     41  102 0.051 3.713572 54.204184 61.139754 48.055372
2     59   97 0.028 4.077537 45.159320 50.473861 40.404363
3     13   54 0.104 2.564949  9.394915 11.468159  7.696477
4      3   32 0.036 1.098612  4.207625  5.635458  3.141556
5     91  113 0.055 4.510860 80.995469 94.273978 69.587240
6     13   82 0.160 2.564949 26.115068 29.223993 23.336878
```

```
'data.frame':   25 obs. of  7 variables:
 $ numspp   : int  41 59 13 3 91 13 5 14 55 6 ...
 $ size     : int  102 97 54 32 113 82 49 67 109 34 ...
 $ dist     : num  0.051 0.028 0.104 0.036 0.055 0.16 0.199 0.179 0.056 0.09 ...
 $ elnumspp : num  3.71 4.08 2.56 1.1 4.51 ...
```

```
 $ predsize : Named num  54.2 45.16 9.39 4.21 81 ...
  ..- attr(*, "names")= chr [1:25] "1" "2" "3" "4" ...
 $ uppersize: Named num  61.14 50.47 11.47 5.64 94.27 ...
  ..- attr(*, "names")= chr [1:25] "1" "2" "3" "4" ...
 $ lowersize: Named num  48.06 40.4 7.7 3.14 69.59 ...
  ..- attr(*, "names")= chr [1:25] "1" "2" "3" "4" ...
```

Plotting the estimated line with a confidence ribbon

Now that we have everything together in one dataset we are ready to make the plot. I use `geom_line()` to add the line for the mean and `geom_ribbon()` to add a confidence interval. I make the ribbon transparent via `alpha`. The lines in the rug show the raw data values for island size.

We would need to define all the elements of the graph in the caption, including the line, ribbon, and rug. We would also want to indicate what values any other variables are fixed to (in this case, `dist` is fixed at it's median of 0.104 km).

```
ggplot(mangrove, aes(x = size, y = numspp) ) + # Establish x and y axes
    geom_rug(sides = "b") + # Add observed X from dataset
    geom_line(aes(y = predsize) ) + # Add a line from the predicted values
    geom_ribbon(aes(ymin = lowersize, ymax = uppersize), alpha = .25) +
    # Add an approximate confidence interval "ribbon"
    labs(x = expression(paste("Size of island ", "(", m^2, ")", sep = "") ),
        y = "Mean number of\ncolonizing insect species") + # change axis labels
    theme_bw(base_size = 16) # change theme b&w, increase font size
```