Module 4: Linear Models using R

The purpose of this handout is to introduce you to the variety of different linear models you can do in R. This module assumes you are using the `RxP.byTank` dataset that was created in the Module 3.

This handout will cover the following analyses and techniques:

- Multi-way Analysis of Variance (ANOVA)
- Linear regression
- Analysis of Covariance (ANCOVA)
- Plotting regressions

As a reminder, for now we are concerned only with "normal" data. By normal data, we are referring to data that fit a normal distribution, with an approximately even distribution of values around the mean. Table 1 shows the variety of models that we can conduct under the `lm` framework. Note that all we do is change the number and type of predictor variables to do different types of models.

Table 1: Linear models and how they relate to different types of response and predictor variables

Data Distribution	Response variable	Predictor variable	# Predictors	Test
Normal	continuous	discrete	one	One-way ANOVA
Normal	continuous	discrete	multiple	Multi-way ANOVA
Normal	continuous	continuous	one	Linear regression
Normal	continuous	continuous	multiple	Multiple regression
Normal	continuous	discrete & continuous	multiple	ANCOVA

1 The Data

This section serves as a reminder of the RxP experiment, which come from the following article. You can probably skip it.

- Touchon, J.C., McCoy, M.W., Vonesh, J.R., and Warkentin, K.W. 2013. Effects of hatching plasticity carry over through metamorphosis in red-eyed treefrogs. *Ecology*. 94(4): 850-860.
- The experiment consisted of 96 400 L mesocosm tanks arrayed in an open field in the northwest corner of Gamboa. The mesocosms were spatially arranged in 12 blocks of 12 tanks each. Each block consisted of 1 tank from each of 12 unique treatment combinations. Each tank began with 50 tadpoles and the experiment ended when all tadpoles reached metamorphosis or had died.
- **Predictor variables**
  - Hatching age (Early [4 days post-oviposition] or Late [6 days post-oviposition])
  - Predators (Control, Nonlethal dragonfly larvae or Lethal dragonfly larvae)

  - Resources (Low [0.75 g] or High [1.5 g] food level, added every 5 days)
  - Block where the metamorph was reared
  - Tank where the metamorph was reared

- We wanted to know how when a frog embryo hatches might affect its development to metamorphosis under various combinations of predators and resources.
- **Response variables**
  - Age at hatching, both in terms of time since eggs were oviposited and time since emergence began (definied as Day 1)
  - Snout-vent length at emergence
  - Tail length at emergence
  - Snout-vent length at completion of tail resorption
  - Mass at completion of tail resorption
  - Number of days needed for each metamorph to fully resorb the tail
- During the course of the experiment disease broke out in 18 of the mesocosms containing Nonlethal predators and thus those tanks have been excluded from the dataset.

2 Linear Models

The model commonly referred to as a linear model, or `lm()`, is one of the most flexible and useful in all of statistics. We will discuss generalized linear models (GLMs) in Module 5, which allow us to model non-normal data.

2.1 Multi-way Analysis of Variance - ANOVA

The linear models from Module 3 had a single categorical predictor, which is also referred to as a one-way ANOVA. But what about when we have multiple categorical predictors that may be interacting? This is generally referred to as a multi-way ANOVA (two predictors would be a two-way ANOVA, three predictors would be a three-way ANOVA, etc). For example, we might expect that the effects of predators on SVL or age at metamorphosis or mass at metamorphosis would differ under different resource conditions. To code an interaction between two effects in `R` you use a colon (`:`). However, `R` also provides us with a shortcut for writing both individuals effects (e.g.,`Pred` or `Res`) and the interaction effect (e.g., `Pres:Res`). Using the asterisk, `*` tells `R` to look at both individual and interaction effects between whatever predictors you have provided. Thus, `Pred*Res` is the same as writing `Pred+Res+Pres:Res`. For example,

```
> lm2<-lm(log.SVL.final~Pred*Res, data=RxP.byTank)
> summary(lm2)

Call:
lm(formula = log.SVL.final ~ Pred * Res, data = RxP.byTank)
```

```
Residuals:
      Min        1Q    Median        3Q       Max
-0.128839 -0.031148 -0.001991  0.034934  0.137993

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)   2.93082    0.01392 210.507  < 2e-16 ***
PredNL        0.03254    0.02524   1.289    0.201
PredL         0.08936    0.01969   4.539 2.22e-05 ***
ResLo        -0.03204    0.01969  -1.627    0.108
PredNL:ResLo -0.00214    0.03569  -0.060    0.952
PredL:ResLo   0.02231    0.02785   0.801    0.426
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1

Residual standard error: 0.05569 on 72 degrees of freedom
Multiple R-squared:  0.4446,  Adjusted R-squared:  0.4061
F-statistic: 11.53 on 5 and 72 DF,  p-value: 3.4e-08
```

Just like in Module 3, we can use the `anova()` or `Anova()` functions to calculate summary statistics for each of our predictors and the interaction, which is what you would likely report in a manuscript.

```
> anova(lm2)
Analysis of Variance Table

Response: log.SVL.final
          Df   Sum Sq  Mean Sq F value   Pr(>F)
Pred       2 0.165021 0.082511 26.5616 2.29e-09 ***
Res        1 0.010734 0.010734  3.4555  0.06713 .
Pred:Res   2 0.002566 0.001283  0.4131  0.66316
Residuals 72 0.223660 0.003106
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1
```

This shows that we have a very strong effect of Predator treatment, a marginal effect of Resource treatment, and no interaction between the two. In other words, the effect of predators on metamorph SVL does not change with different Resource levels. Thus, we can say that there is no interaction between Predators and Resources on the SVL of metamorphosing froglets.

2.1.1 Interpreting an output with interactions

Once you have multiple, interacting predictors the model output becomes increasing difficult to interpret. However, it is worth understanding how to interpret the output. Just to review, our Predator treatment has three levels (Control, Nonlethal and Lethal) and our Resource treatment has two level (Low and High), giving us 6 total combinations of Predators-by-Resources (e.g., Control-Low, Control-High, Nonlethal-Low, etc.). There are six lines in the `Coefficients` section of the `Summary` output. Just like with a one-way ANOVA, `R` works alphabetically (or at least it works in the order of how you've set up

your factor levels, if you have changed the order from the default of alphabetical). Thus, the baseline listed as the `(Intercept)` is the Pred Control-Resource High treatment combo. The subsequent lines all describe changes in the effect sizes in relation to that baseline. Similar to the one-way ANOVA we saw above, the treatment listed in the left most column of the `Coefficients` section shows what has changed and only what has changed (thus, if only one thing is listed, that means the other thing is still the same as the baseline). Thus, if the baseline is the Control-High, the second line (listed as `PredL`) is the Lethal-High treatment combo and the third line (listed as `PredNL`) is the Nonlethal-High treatment combo. Similarly, the fourth line (listed as `ResLo`) is the Control-Low treatment combo. The last two lines are the Lethal-Low and Nonlethal-Low treatment combos.

Calculating the effect sizes follows similar logic, but with the addition that any modifications to effect sizes include all lower effects. What does that mean? If the baseline (Control-High) is 2.93082, then the Lethal-High is 2.93082+0.08936, just like for the one-way ANOVA example described earlier. The calculations for NL-Hi and C-Lo treatments are the same. However, when we get to the last two lines of the output (listed as `PredL:ResLo` and `PredNL:ResLo`), the correct effect sizes have to be calculated including earlier modifications. Thus for `PredL:ResLo`, we have the baseline of 2.93082, plus the effect of `PredL` alone (0.08936) plus the effect of `ResLo` alone (-0.03204) plus the combined effect of `PredL:ResLo` (0.02231). Similar logic applies for calculating the `PredNL:ResLo` effect. The calculations are shown below. We can confirm that these treatment estimates are indeed correct by calculating the raw treatment means using `tapply`.

```
> exp(2.93082)#(Intercept) aka Control High
[1] 18.74299
> exp(2.93082+0.08936)#PredL aka Lethal High
[1] 20.49498
> exp(2.93082+0.03254)#PredNL aka Nonlethal High
[1] 19.36292
> exp(2.93082-0.03204)#ResLo aka Control Low
[1] 18.15199
> exp(2.93082+0.08936-0.03204+0.02231)#PredL:ResLo aka Lethal Low
[1] 20.29653
> exp(2.93082+0.03254-0.03204-0.00214)#PredNL:ResLo aka Nonlethal Low
[1] 18.71228

> exp(tapply(RxP.byTank$log.SVL.final, paste(RxP.byTank$Pred, RxP.byTank$Res), mean))
    C Hi     C Lo     L Hi     L Lo    NL Hi    NL Lo
18.74300 18.15203 20.49510 20.29674 19.36289 18.71230
```

This might seem like an unnecessarily cumbersome way to calculate the means of your treatment groups, and it is. However, working from simpler to more complex model structures provides us with a way to really understand the output that `R` gives us, which will be very useful later on.

2.1.2 Posthoc comparisons when you have interactions?

Is it possible to use the `glht()` function described earlier to do multiple comparisons, to find out which of our six treatment combinations are significantly different from one another? It is possible, but not the way the model is currently written (with an interaction). We must first create a dummy variable that has the six levels, each of our treatment Predator x Resource treatment combinations. Let's call the

dummy variable `PredRes`. We can do this with the `paste()` function, which simply allows us to paste two groups of text together. We can nest the `paste()` command within the function `as.factor()` to ensure that our new dummy variable is a factor. The `paste()` function takes 3 arguments: the two vectors you are pasting together, and some text you will use to separate the text from the vectors. In the code below I've separated the Pred and Res values with a hyphen, but you could easily use something else (e.g., a space, a period, an underscore, nothing at all, etc).

```
> RxP.byTank$PredRes<-as.factor(paste(RxP.byTank$Pred, RxP.byTank$Res, sep="-"))
```

It is important to note that this model is mathematically identical to `lm2`, coded with the interaction between Predator and Resource treatments. Now, we can create a new `lm` object and run post-hoc tests on the six treatment combinations. Remember that `glht()` is in the `multcomp` package.

```
> lm3<-lm(log.SVL.final~PredRes, data=RxP.byTank)
> ph3<-glht(lm3, linfct=mcp(PredRes="Tukey"))
> summary(ph3)

 Simultaneous Tests for General Linear Hypotheses

Multiple Comparisons of Means: Tukey Contrasts


Fit: lm(formula = log.SVL.final ~ PredRes, data = RxP.byTank)

Linear Hypotheses:
                     Estimate Std. Error t value Pr(>|t|)
C-Lo - C-Hi == 0   -0.032038   0.019690  -1.627  0.57747
L-Hi - C-Hi == 0    0.089365   0.019690   4.539  < 0.001 ***
L-Lo - C-Hi == 0    0.079640   0.019690   4.045  0.00173 **
NL-Hi - C-Hi == 0   0.032538   0.025237   1.289  0.78554
NL-Lo - C-Hi == 0  -0.001640   0.025237  -0.065  1.00000
L-Hi - C-Lo == 0    0.121403   0.019690   6.166  < 0.001 ***
L-Lo - C-Lo == 0    0.111678   0.019690   5.672  < 0.001 ***
NL-Hi - C-Lo == 0   0.064576   0.025237   2.559  0.11804
NL-Lo - C-Lo == 0   0.030398   0.025237   1.205  0.82958
L-Lo - L-Hi == 0   -0.009726   0.019690  -0.494  0.99612
NL-Hi - L-Hi == 0  -0.056827   0.025237  -2.252  0.22272
NL-Lo - L-Hi == 0  -0.091005   0.025237  -3.606  0.00697 **
NL-Hi - L-Lo == 0  -0.047102   0.025237  -1.866  0.42466
NL-Lo - L-Lo == 0  -0.081279   0.025237  -3.221  0.02203 *
NL-Lo - NL-Hi == 0 -0.034178   0.029768  -1.148  0.85614
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1
(Adjusted p values reported -- single-step method)

> cld(ph3)
 C-Hi  C-Lo  L-Hi  L-Lo NL-Hi NL-Lo
  "a"   "a"   "b"   "b"  "ab"   "a"
```

Hopefully the output above demonstrates why the `cld()` function is so useful! We can easily see that the two Lethal predator treatments are different from all other treatments except the Nonlethal-High treatment, which does not differ from any other treatments. We can visualize the differences amongst treatments by plotting the means and standard errors. The figure below uses a modified version of the code from Module 2.

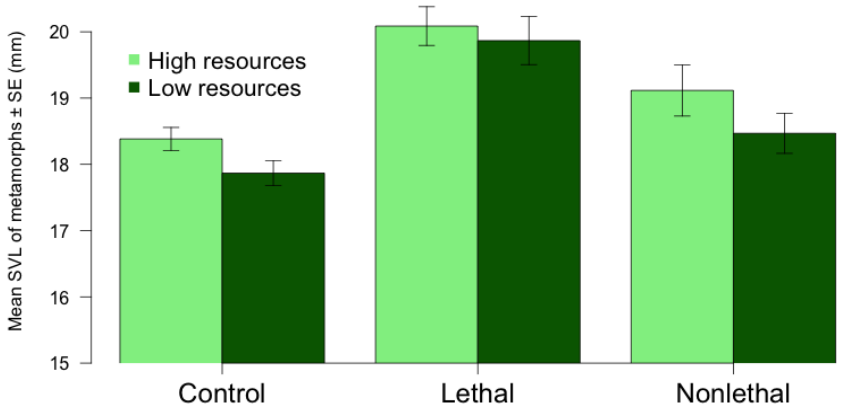

Figure 1: A barplot showing the means and standard error of six Predator x Resource treatment combinations

2.2 Linear regression

So far we have discussed scenarios where you have categorical predictors. But what about when you have continuous predictors? As long as your response variable is normally distributed, that is a linear regression. As mentioned above, in `R` a linear regression is just another form of `lm`. For example, with a linear regression we can ask if SVL at metamorphosis is influenced by the length of the larval period? Here we will use Age at metamorphosis in terms of DPO (days post-oviposition) as our predictor, which is the length of time from when eggs were laid until the froglet crawled out of the water.

```
> lm4<-lm(log.SVL.final~log.Age.DPO, data=RxP.byTank)
> summary(lm4)

Call:
lm(formula = log.SVL.final ~ log.Age.DPO, data = RxP.byTank)

Residuals:
      Min        1Q    Median        3Q       Max
-0.182298 -0.049070  0.000516  0.038342  0.159057

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)   3.44001    0.09211  37.345  < 2e-16 ***
log.Age.DPO  -0.11624    0.02232  -5.208 1.58e-06 ***
```

```
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1

Residual standard error: 0.06244 on 76 degrees of freedom
Multiple R-squared:  0.263,  Adjusted R-squared:  0.2533
F-statistic: 27.12 on 1 and 76 DF,  p-value: 1.581e-06

> anova(lm4)
Analysis of Variance Table

Response: log.SVL.final
             Df  Sum Sq  Mean Sq F value    Pr(>F)
log.Age.DPO   1 0.10576 0.105757  27.123 1.581e-06 ***
Residuals    76 0.29633 0.003899
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1
```

Indeed, `log.Age.DPO` has a highly significant effect on `log.SVL.final`. We can also see in the `Coefficients` section of the `summary()` output that the regression has a negative slope, indicating that as Age increases SVL decreases (more on this below). As above, we should look at the diagnostic plots to evaluate how well the model fits.

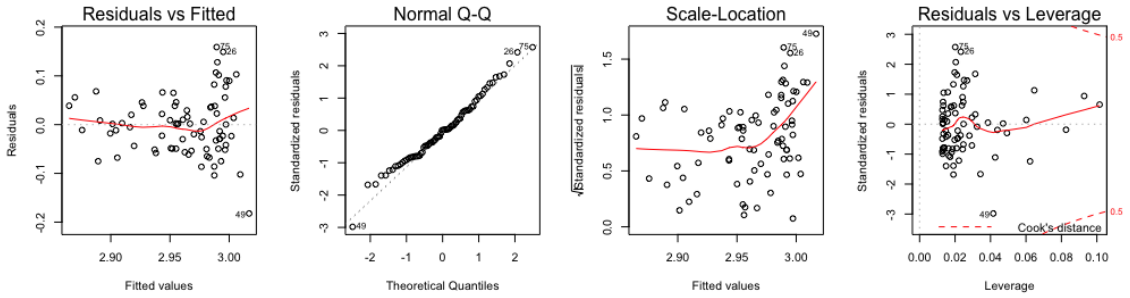

Figure 2: Four diagnostic plots of model `lm4`

Overall, this looks pretty good. One nice things about these plots is that `R` will label observations that may be problematic. Looking across the first plots, we can see that observations 26, 49 and 75 have been flagged. The location on the x-axis (fitted values) tells us where the model expects those points to fall according to the regression, and the residuals tells us how far above or below that expectation those points are. We can see that obs 49 is much smaller than expected, whereas obs 75 and 26 are larger than expected. Are these problematic? These points actually look pretty good in the Q-Q plot and do not look like a big deal according to the leverage plot (panel 4), so it's somewhat hard to say. To know if these observations are indeed influencing our model, we should re-run the model while excluding them and see if it changes our results in any appreciable way.

2.2.1 Removing potentially problematic data?

What is the easiest way to exclude a few data points? As we've seen in previous models, we could use square brackets ([]'s) to to subset out the rows we do not want to include in our dataset. However, whereas we previous used indexing to create a new data frame in the memory, here we can use indexing to exclude the potentially problematic data directly in the model we are running. By changing the `data=` argument in our model specification from `data=RxP.byTank` to `data=RxP.byTank[-c(26,49,75),]` we can run the model but without rows 26, 49 and 75. One reason to do this is that we do not have to permanently modify our dataset or create a new object in the memory. We just run the model but say to exclude the three rows we want to get rid of.

```
> lm4.1<-lm(log.SVL.final~log.Age.DPO, data=RxP.byTank[-c(26,49,75),])
> summary(lm4.1)

Call:
lm(formula = log.SVL.final ~ log.Age.DPO, data = RxP.byTank[-c(26,
    49, 75), ])

Residuals:
      Min        1Q    Median        3Q       Max
-0.102504 -0.047131  0.001842  0.038959  0.129485

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)   3.44064    0.08238   41.77  < 2e-16 ***
log.Age.DPO -0.11681    0.01990   -5.87 1.19e-07 ***
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1

Residual standard error: 0.05431 on 73 degrees of freedom
Multiple R-squared:  0.3207,  Adjusted R-squared:  0.3114
F-statistic: 34.46 on 1 and 73 DF,  p-value: 1.19e-07

> anova(lm4.1)
Analysis of Variance Table

Response: log.SVL.final
             Df  Sum Sq Mean Sq F value   Pr(>F)
log.Age.DPO   1 0.10164 0.10164  34.457 1.19e-07 ***
Residuals    73 0.21533 0.00295
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1
```

Removing those three observations does not have a dramatic effect on our model. Without them, our R-squared is admittedly about 6% greater, and our F-statistic is larger, but the slope is still in the same direction and the fundamental relationship being described has not changed. Thus, it is not a problem to leave the three observations in the model.

2.2.2 Interpreting the regression

Since we have determined our original model, `lm4`, is not problematic, we can proceed with interpreting the output. Recall that the formula for a straight line is y = *slope**x + *intercept* (you might have learned it as $y=mx+b$ back in high school). When you look at the `summary()` output, you can see it looks exactly the same as what we saw when we had categorical predictors (i.e., the one-way and multi-way ANOVA's described above). However, now our `(Intercept)` is the intercept for the regression line, and the parameter listed for `log.Age.DPO` is our slope. The summary output tells us that for every unit increase in the log of Age.DPO, we have a unit decrease in the log of SVL.final. Another way to write this would be as follows:
`log.SVL.final = 3.44001 - 0.11624 * log.Age.DPO`.

Since this is a log-log model (i.e., both the predictor and response variables were log-transformed), we know that the relationship between Age and SVL is fundamentally nonlinear. We can plot this in two ways. The easiest thing to do is to keep the axes and values on a log-scale. However, without much difficulty we can plot the regression back on the values of the original data (days and mm), which may make the model more intuitive visually.

2.2.3 Plotting the regression

The function `abline()` plots a straight line on an existing scatterplot. `abline()` needs two arguments, an intercept (a) and a slope (b). You can also pass other optional plotting parameters if you want to change the line width, style, color, etc.

```
> plot(log.SVL.final~log.Age.DPO, data=RxP.byTank)
> abline(3.44001,-0.11624)
```

A shortcut for when you have a simple linear regression (i.e., you only have a single continuous predictor) is to just type `abline(lm4)`. The function knows to get the intercept and regression from the model. This does not work with models with multiple predictors.

In order to plot the regression back on the original data, we have to do some work ourselves. Recall the equation for the regresssion listed above (`log.SVL.final = 3.44001 - 0.11624 * log.Age.DPO`). If we wanted to know the predicted log of the final size of a metamorph emerging at, for example, 40 days post-oviposition, we can easily do the following.

```
> 3.4401 - 0.11624 * log(40)
[1] 3.011305
```

That answer is the log SVL, so we can put that back into real units (mm) by exponentiating it.

```
> exp(3.011305)
[1] 20.31389
```

Thus the predicted size of a metamorph emerging 40 DPO is 20.3 mm. The same logic applies for any time DPO we might choose. For example, to calculate the predicted size of metamorphs emerging at 60, 80 or 100 days post-oviposition, we can easily type the following:

```
> exp(3.4401 - 0.11624 * log(60))
[1] 19.37868
```

```
> exp(3.4401 - 0.11624 * log(80))
[1] 18.74137
> exp(3.4401 - 0.11624 * log(100))
[1] 18.2615
```

You can see how as DPO increases, the predicted SVL of metamorph decreases. Now let's take this a step further. Instead of merely passing R a single value for DPO, we can pass it a vector. For example, to predict the sizes of metamorphs for every day between 40 and 60 DPO, we can do the following.

```
> exp(3.4401 - 0.11624 * log(40:60))
 [1] 20.31389 20.25566 20.19900 20.14383 20.09007 20.03766 19.98653 19.93663
 [9] 19.88790 19.84029 19.79375 19.74824 19.70372 19.66014 19.61747 19.57567
[17] 19.53471 19.49457 19.45519 19.41657 19.37868
```

The colon (:) in the above example is a shortcut that R knows to make a vector of whole integers between two values (go ahead and type two numbers with a colon between them into the console and see what happens. what happens if the first number you type is larger than the second one?). Let's make a vector of those predicted values for every day between 35 and 145 DPO, which is essentially the length of our x-axis. Once we have that vector, we can use the function `lines()` to plot them. `lines()` is a function that requires only two things: a vector of values for x-coordinates and a matching vector of y-values (they must be the same length). Since we used `35:145` to define the range of predicted SVL's, that will also be the range of our x-coordinates.

```
> plot(SVL.final~Age.DPO, data=RxP.byTank) #plot the raw values
> SVL.line<-exp(3.44001-0.11624*log(35:145)) #make the vector of the predicted
  SVL's and save it as an object
> lines(x=35:145, y=SVL.line, lwd=2) #Use the predicted values to plot a line
```

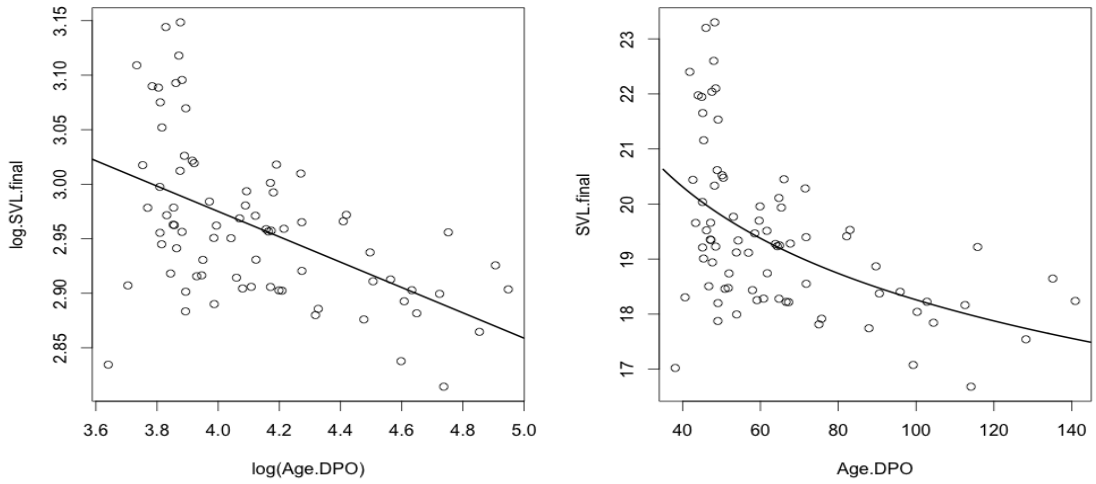

Figure 3: Two scatterplots of SVL against age DPO, both with a regression line plotted from `lm4`. The figure on the left is plotted using a log-log scale, whereas the figure on the right uses the original values. Doesn't the figure on the right look nice?

2.3 Analysis of covariance - ANCOVA

What happens when we have a combination of continuous and categorical predictors? For example, perhaps we would like to know if the relationship we just found between age DPO and SVL differs when tadpoles are raised in different predator environments? Note that now it is important to use the `Anova()` function from the `car` package to calculate summary statistics. You can download the car package by using the Package Installer from the menu, or simply by typing `install.packages("car")` at the prompt.

```
> lm5<-lm(log.SVL.final~log.Age.DPO*Pred, data=RxP.byTank)
> summary(lm5)

Call:
lm(formula = log.SVL.final ~ log.Age.DPO * Pred, data = RxP.byTank)

Residuals:
      Min        1Q    Median        3Q       Max
-0.122214 -0.029839  0.002357  0.036826  0.122903

Coefficients:
                  Estimate Std. Error t value Pr(>|t|)
(Intercept)        3.08467    0.12387  24.902   <2e-16 ***
log.Age.DPO       -0.03974    0.02890  -1.375   0.1733
PredNL            -0.05606    0.21020  -0.267   0.7905
PredL              0.67183    0.24458   2.747   0.0076 **
```

```
log.Age.DPO:PredNL  0.02000    0.04982   0.401   0.6893
log.Age.DPO:PredL  -0.14883    0.06090  -2.444   0.0170 *
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1

Residual standard error: 0.05227 on 72 degrees of freedom
Multiple R-squared:  0.5107,  Adjusted R-squared:  0.4767
F-statistic: 15.03 on 5 and 72 DF,  p-value: 4.324e-10
```

For the sake of comparison, below I have the output from both the `anova()` and `Anova()` functions.

```
> Anova(lm5)
Anova Table (Type II tests)

Response: log.SVL.final
                   Sum Sq Df F value    Pr(>F)
log.Age.DPO       0.019898  1  7.2824   0.00867 **
Pred              0.079871  2 14.6155 4.706e-06 ***
log.Age.DPO:Pred  0.019731  2  3.6105   0.03204 *
Residuals         0.196732 72
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1
> anova(lm5)
Analysis of Variance Table

Response: log.SVL.final
                  Df   Sum Sq  Mean Sq F value    Pr(>F)
log.Age.DPO        1 0.105757 0.105757 38.7050 2.908e-08 ***
Pred               2 0.079871 0.039935 14.6155 4.706e-06 ***
log.Age.DPO:Pred   2 0.019731 0.009865  3.6105   0.03204 *
Residuals         72 0.196732 0.002732
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1
```

Notice that the significance of log.Age.DPO is much higher in the `anova()` output than the `Anova()` version. This is because the `anova()` version calculates significance of each predictor in a step-by-step manner, so the significance of the first predictor (in this case log.Age.DPO) is considered the only thing in the model. The `Anova()` output, on the other hand, calculates significance assuming everything else in the model, which is why the latter predictors have the same stats in both summary outputs. In general, the approach used by `Anova()` is more conservative and therefore preferrable.

2.3.1 Interpreting the ANCOVA

What does this model tell us? In essence, this is saying that the relationship between metamorph SVL and the length of the larval period differs depending on the presence of lethal, nonlethal (or no) Predators in the environment. Another way to phrase this is that the slopes of the three regressions (one for each of our three Predator treatments) are different. We don't really know which of the three are different,

simply *that there is a difference amongst the three treatments* and because we have two predictors we cannot use `glht()` to conduct post-hoc comparisons. Instead, the most logical thing to do is plot the data and see what is going on.

2.3.2 Plotting the ANCOVA

Recall earlier how we calculated the parameter estimates for a two-way ANOVA. The `summary()` output gives us the coefficients for each of our predictor variables in relation to a baseline, and the baseline is the alphabetically first treatment level (which in our case is C, Control). The same logic applies here, but now we have three regressions, each corresponding to our three Predator treatments.

The first two lines of the `summary()` output labeled `(Intercept)` and `log.Age.DPO` correspond to the intercept and slope for our baseline regression, the Control. Anything subsequently labeled with `log.Age.DPO` will be a slope, and anything that just lists the name of a Predator treatment will be an intercept. Thus, for example, the coefficient labeled `PredL` is the modified intercept for the Lethal Predator treatment, and the coefficient labeled `log.Age.DPO:PredL` is the corresponding modified slope. We can use `abline()` to plot the three regressions on the log-log scales as before.

In the code below, we have used the additional plot argument `pch` to define a new plot character (`pch=21` gives you circles with a colored background, or `bg`). Do a Google Images search for "R pch" to see the 25 different built in plotting characters. We've told `R` to color the background (`bg`) of the points based on the Predator treatment, which will once again rank them alphabetically and assign colors in order. Color 1 is black, 2 is red and 3 is green. We've also added a legend, and told R to place it in the top right corner. Note that the code below is for the left figure in Figure 8. Can you modify it to make the figure on the right?

```
plot(log.SVL.final~log.Age.DPO, data=RxP.byTank, bg=RxP.byTank$Pred, pch=21)
abline(3.08467, -0.03974, lwd=2, col=1)#Regression for Control
abline(3.08467+0.67183, -0.03974-0.14883, lwd=2, col=2)#Regression for Lethal
abline(3.08467-0.05606, -0.03974+0.02000, lwd=2, col=3)#Regression for Nonlethal
legend(x="topright", legend=c("Control","Nonlethal","Lethal"),
text.col=c(1,3,2), cex=2, bty="n")
```

2.3.3 Interpreting the plot of the ANCOVA

As mentioned above, we cannot use post-hoc comparisons to try to understand the nature of the significant interaction between log.Age.DPO and Predator treatment (we could try, but it would not really make much sense to do so). Instead, we plotted the data. So what does the plot tell us? In essence, what we can glean from the plot is that the presence of Lethal predators fundamentally changes the relationship between larval period and size at metamorphosis. Nonlethal predators slightly increase the size of metamorphs, but the way that size decreases the longer tadpoles are in the water doesn't really change. Lethal predators, on the other hand, cause a drastic increase in SVL and a general shortening of the larval period and this results in a different relationship between size and larval period.

3 The `predict()` function

Now that we have calculated our curves by hand, let's calculate them using a function built in to `R`: `predict()`. The most useful reason to use `predict()` is that it provides an easy way to calculate 95%

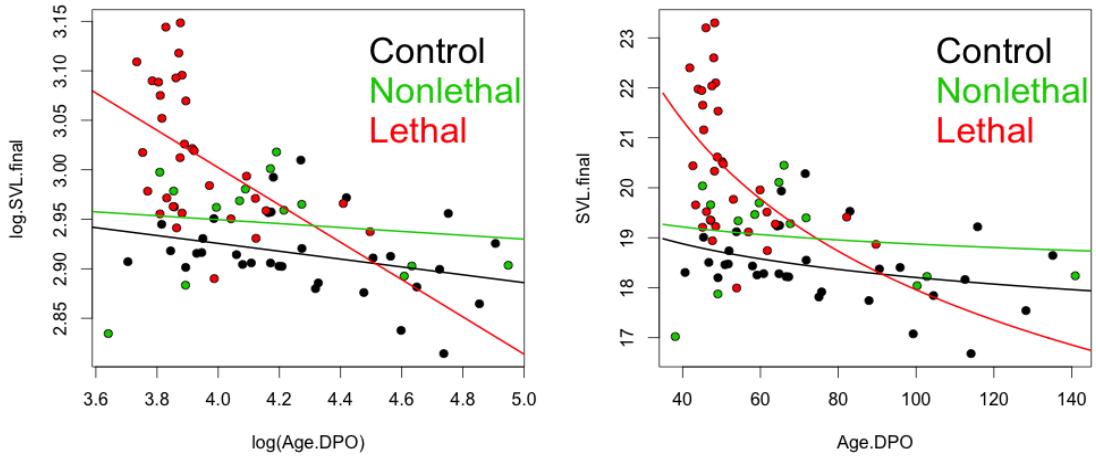

Figure 4: Two scatterplots of metamorph SVL against age DPO, with regression lines plotted from `lm5` for each predator treatment. The figure on the left is plotted using a log-log scale, whereas the figure on the right uses the original values. The figures are identical except that the axes on the right are in the original units and are not log-transformed.

confidence intervals around regression lines.

Using the `predict()` function takes a little getting used to, but it is very useful once you get the hang of it. `predict()` essentially requires just two arguments, but in practical use you will most likely want to provide it with three arguments. You need to provide `predict()` with 1) the model you are predicting values from, 2) a set of "new data" to populate with new values, and 3) if you want to calculate confidence intervals (the default is to not provide them). In Module 5, we will add a fourth argument, which is the type of output to provide.

The most confusing of these is the "new data" argument. If you look back at our model `lm4`, we have one predictor, the `Age.DPO` predictor (on the log scale of course). The "new data" is essentially what we did in section 2.2.3, where we provided a length of x-axis values for the regression equation to use to calculate the y-axis values of final SVL. Now we can let `predict()` do that math for us. Let's create a new data frame with hypothetical values for our x-axis (the log of age at metamorphosis post-oviposition), and a blank column for `predict()` to fill in for the corresponding final SVL. First, let's figure out the smallest and largest values of our x-axis values.

```
> min(log(RxP.byTank$Age.DPO))
[1] 3.640506
> max(log(RxP.byTank$Age.DPO))
[1] 4.948068
```

Okay, our smallest value is around 3.64 and our largest is about 4.94. To make a sequence of numbers that spans that range, we can use the function `seq()`, which allows you to specify a minimum, a maximum, and either how many numbers you want it to output or the interval between the numbers you want. Thus, if we want to make a sequence between 3.64 and 4.94 every 0.01 values, we can type:

```
> seq(3.64,4.94,0.01)
  [1] 3.64 3.65 3.66 3.67 3.68 3.69 3.70 3.71 3.72 3.73 3.74 3.75 3.76 3.77
```

```
 [15] 3.78 3.79 3.80 3.81 3.82 3.83 3.84 3.85 3.86 3.87 3.88 3.89 3.90 3.91
 [29] 3.92 3.93 3.94 3.95 3.96 3.97 3.98 3.99 4.00 4.01 4.02 4.03 4.04 4.05
 [43] 4.06 4.07 4.08 4.09 4.10 4.11 4.12 4.13 4.14 4.15 4.16 4.17 4.18 4.19
 [57] 4.20 4.21 4.22 4.23 4.24 4.25 4.26 4.27 4.28 4.29 4.30 4.31 4.32 4.33
 [71] 4.34 4.35 4.36 4.37 4.38 4.39 4.40 4.41 4.42 4.43 4.44 4.45 4.46 4.47
 [85] 4.48 4.49 4.50 4.51 4.52 4.53 4.54 4.55 4.56 4.57 4.58 4.59 4.60 4.61
 [99] 4.62 4.63 4.64 4.65 4.66 4.67 4.68 4.69 4.70 4.71 4.72 4.73 4.74 4.75
[113] 4.76 4.77 4.78 4.79 4.80 4.81 4.82 4.83 4.84 4.85 4.86 4.87 4.88 4.89
[127] 4.90 4.91 4.92 4.93 4.94
```

Okay, now we are ready to make our data frame. We should name the new variables in our data frame *exactly* what they are called in the model `lm4` so that the `predict()` function will know what they are.

```
> lm4.newdata<-data.frame("log.Age.DPO"=seq(3.64,4.94,0.01))
> head(lm4.newdata)
  log.Age.DPO
1        3.64
2        3.65
3        3.66
4        3.67
5        3.68
6        3.69
```

Now for the last step, to use `predict()` to calculate our regression curve and associated confidence intervals. To calculate 95% CI's, add the argument `se.fit=TRUE`. Note that the default is a 95% CI, but you can change it to be whatever you would like.

```
> lm4.predicted<-predict(lm4, newdata=lm4.newdata, se.fit=T)
> str(lm4.predicted)
List of 4
 $ fit           : Named num [1:131] 3.02 3.02 3.01 3.01 3.01 ...
  ..- attr(*, "names")= chr [1:131] "1" "2" "3" "4" ...
 $ se.fit        : Named num [1:131] 0.0127 0.0126 0.0124 0.0122 0.012 ...
  ..- attr(*, "names")= chr [1:131] "1" "2" "3" "4" ...
 $ df            : int 76
 $ residual.scale: num 0.0624
```

If you look at the structure of the output from `predict()` you see that it is a list. The first item `fit` is the regression curve itself and the second item `se.fit` is the CI. Let's plot it all using the functions `plot()` and `lines()`. To plot the CI's, we just add or subtract the amount of the CI from the predicted regression line. We can also make the CI's dashed lines by specifying `lty=2`, which changes the line type.

```
> plot(log.SVL.final~log.Age.DPO, data=RxP.byTank)
> lines(x=lm4.newdata$log.Age.DPO, y=lm4.predicted$fit)
> lines(x=lm4.newdata$log.Age.DPO, y=lm4.predicted$fit+lm4.predicted$se.fit, lty=2)
> lines(x=lm4.newdata$log.Age.DPO, y=lm4.predicted$fit-lm4.predicted$se.fit, lty=2)
```

What if we wanted to plot this back on original, more intuitive values? Can we just exponentiate the predicted values? Yes we can!

```
> plot(SVL.final~Age.DPO, data=RxP.byTank)
> lines(x=exp(lm4.newdata$log.Age.DPO), y=exp(lm4.predicted$fit))
> lines(x=exp(lm4.newdata$log.Age.DPO), y=exp(lm4.predicted$fit-lm4.predicted$
  se.fit), lty=2)
> lines(x=exp(lm4.newdata$log.Age.DPO), y=exp(lm4.predicted$fit+lm4.predicted$
  se.fit), lty=2)
```

In a later module we will see how to make these sorts of plots using the `ggplot2` package. Both approaches are useful. `ggplot2` can make nice figures quite easily, but it is very useful to know what is going on with the predict function.

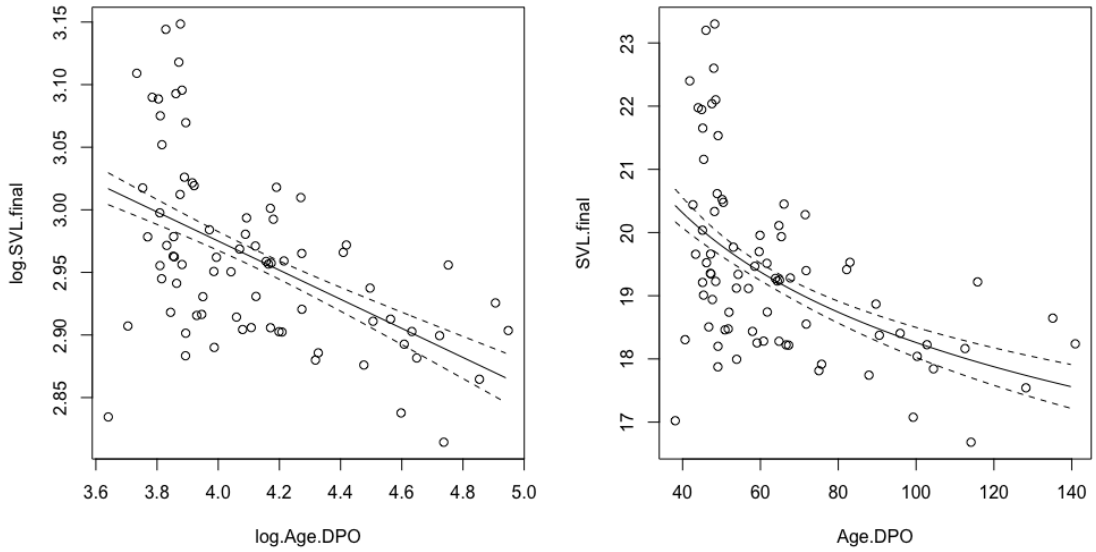

Figure 5: Two scatterplots of metamorph SVL against age DPO, with regression lines plotted from `lm4` and predicted regression lines with 95% confidence intervals. The figure on the left is plotted using a log-log scale, whereas the figure on the right uses the original values. The figures are identical except that the axes on the right are in the original units and are not log-transformed.

4 Assignment!

Here are some exercises to try on your own, to build on the data analysis and plotting skills you have just worked on. Put together a very brief report that completes the following tasks and answers the associated questions.

1. Last week we analyzed the log of final mass by each of the three predictors independently. What happens if you include all three predictors in the model at the same time? Does the significance of any of individual predictors change when we include more variables? If so, what does that tell you about how your interpretation of one variable might change when you include other variables in the model?

2. Make the scatterplot shown on the right of Figure 5. The process is essentially the same as that described in the linear regression section, with regards to how to make the curved plots.

3. Try to use `predict()` to get curves for `lm5`, when you have both continuous and categorical predictors. There are multiple ways of course, but here is one hint: What if the "newdata" object had multiple columns, one for each predictor in the model?

4. Explore the analyses shown here but with different interactions. Pick three different questions to ask. For example does Hatching age interact with Age.DPO to influence the size of tail when froglets crawl out of the water? How is age at metamorphosis influenced by predators and resources, or predators, resources and hatching age? These are just examples, there are lots of questions you can ask. Make sure one of your questions is an ANCOVA style analysis and plot the data as in Figure 4.

Turn in to me (via Moodle) a word doc or pdf with both figures. The assignment is due the evening of Wednesday 2/21/2018.