Module 3: Basic Statistical Analyses using R

The purpose of this handout is to introduce you to basic statistical analyses using R. This module assumes you are using the `RxP.clean` dataset that was created in the Module 2.

This handout will cover the following topics:

- Assessing data normality
- Student's t-test
- One-way Analysis of Variance (ANOVA)

1 Determining What Type of Analysis to Do

There are several questions to ask yourself as you prepare to analyze your data.

- What kind of data do you have?
  - Is the response variable normal or non-normal?
  - Is the response varible continuous or discrete?
  - Is the predictor varible continuous or discrete?
  - Do you have only one predictor or multiple predictors?

For now, we will be 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. Module 5 will introduce what to do with non-normally distributed data.

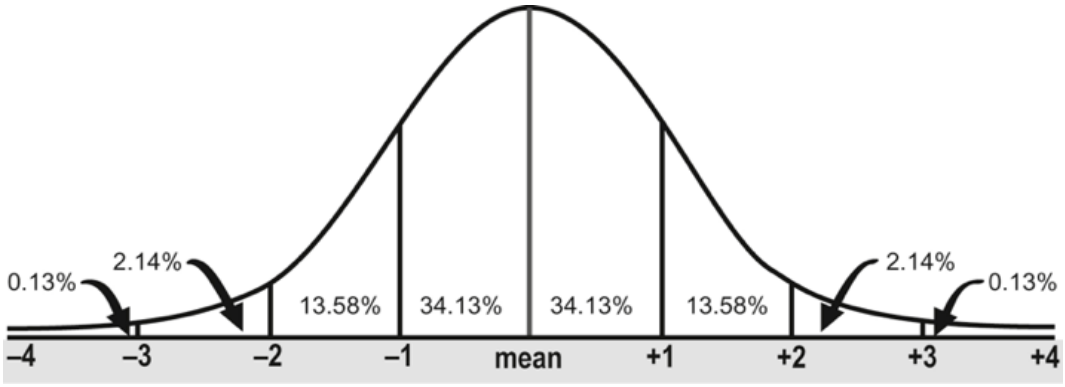

Figure 1: A Gaussian, or normal, distribution showing the even spread of values around the mean

Many standard statistics rely on data being approximately normally distributed. Statistics that require "normal" data are also called parametric statistics. All of these models are coded with the same basic design.

- `lm(Response ~ Predictor1 + Predictor2 +..., data)`

The code above indicates 1) we have a function called `lm` (short for "linear model") that specifies what type of analysis we are going to do, 2) we use a ~ to separate the response (AKA dependent variable) written on the left side of the equation from one or multiple predictors (AKA independent variables) written on the right side of the equation, and 3) we specify where `R` will find the data. This syntax should seem familiar, as we used it both to create basic plots of data and to use the `aggregate()` function to summarize our data. Table 1 shows how different combinations of continuous and discrete predictor variables determine which type of model you might do.

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

2 The Data

This section serves as a reminder of the RxP experiment, which come from the following article.

- 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.

In addition, several outliers were removed in Module 2. The analyses conducted below assume that you have already been through that handout and have removed the outliers and are using the `RxP.clean` dataset.

3 Avoiding Pseudoreplication

Although we have data on nearly 2500 individual metamorphs, those data are not all independent. This is because tadpoles were raised in common environmnents (mesocosms, AKA tanks) and variation between tanks may make certain individuals more similar than others. If we treat all individuals as independent, we are committing pseudoreplication, which is when you artificially inflate your sample size of independent observations. For more details, see the classic article Hurlbert, S.H. 1984. "Pseudoreplication and the design of ecological field experiments." *Ecological Monographs* 54(2):187-211.

One easy way to avoid psuedoreplication is to utilize the mean value for each tank instead of that of individuals. We can summarize our entire dataset with relative ease using the `summarise()` function that we introduced in Module 2. We will necessarily have to get rid of column 1 on Individual ID and column 3 which shows the tank within each block. Recall that we 1) define our dataset, 2) define the various variables we want to group our data by with the `group_by()` function and then 3) use `summarise()` to create the variables that are the mean values from the raw dataset. At each of the three steps you 'pipe' your data from one line to the next. You could also use the `aggregate()` function by defining the 7 columns we want to summarize (our response variables) and binding them together in a single object using the function `cbind()` which *binds* two or more *c*olumns together, then give it the various predictor variables we want to use to summarize the data. Here, we will use `summarise()`. Recall that these functions are found in the `dplyr` package.

```
RxP.byTank<-RxP.clean %>%
            group_by(Block,Tank.Unique,Pred,Hatch,Res) %>%
            summarise(Age.DPO = mean(Age.DPO),
            Age.FromEmergence = mean(Age.FromEmergence),
            SVL.initial = mean(SVL.initial),
            Tail.initial = mean(Tail.initial),
            SVL.final = mean(SVL.final),
            Mass.final = mean(Mass.final),
            Resorb.days = mean(Resorb.days))
RxP.byTank<-as.data.frame(RxP.byTank)
> str(RxP.byTank)
'data.frame': 78 obs. of  12 variables:
 $ Block            : int  1 1 1 1 1 1 1 1 1 1 ...
```

```
$ Tank.Unique      : int  1 2 3 4 5 6 7 8 9 10 ...
$ Pred             : Factor w/ 3 levels "C","L","NL": 3 1 1 2 3 2 3 1 2 1 ...
$ Hatch            : Factor w/ 2 levels "E","L": 2 1 2 2 1 1 2 1 2 2 ...
$ Res              : Factor w/ 2 levels "Hi","Lo": 1 1 1 2 1 1 2 2 1 2 ...
$ Age.DPO          : num  47.2 45.4 53.8 56.9 64.8 ...
$ Age.FromEmergence: num  13.2 11.4 19.8 22.9 30.8 ...
$ SVL.initial      : num  19.4 18.4 18.9 18.8 19.7 ...
$ Tail.initial     : num  4.83 5.37 4.8 4.63 5.43 ...
$ SVL.final        : num  19.7 19 19.1 19.1 20.1 ...
$ Mass.final       : num  0.418 0.382 0.412 0.382 0.486 ...
$ Resorb.days      : num  3.49 3.79 3.51 3.65 4.22 ...
```

Summarizing our data in this way is nice because now our observations are independent (although we might want to think about the fact that tanks come from within blocks). However, we have reduced our dataset of nearly 2500 individuals to just 78 observations! That is throwing out or ignoring a tremendous amount of data that we worked very hard to obtain. In Module 6 on Mixed Effects Models, we will discuss how you can utilize all of your data while controlling for the non-independence of certain observations.

4 Testing for Normality in Your Data

How do you know if your data fit a normal distribution? That is an essential but somewhat difficult question to answer. There are multiple ways to try and address that question. The first and most important thing to do is simply to plot a histogram of your data. In addition, we will discuss two different approaches: 1) normality tests and 2) using maximum likelihood to estimate the most appropriate error distribution.

4.1 Looking at the data

The first and easiest thing to do if you want to see if your data are normal or not is to plot the histogram of values. You could use either the `hist()` function or the `qplot()` function found in the `ggplot2` package. Either way, a histogram should give you a general sense of what your data look like. Remember that data which are Normally distributed will have a relatively even spread of values above and below the mean. Lets look at the variables `Age.FromEmergence` and `SVL.final` and plot them using `qplot()`. Remember, you have to load the package first using the `library()` function.

```
> qplot(x=Age.FromEmergence, data=RxP.byTank, geom="histogram", bins=10)
> qplot(x=SVL.final, data=RxP.byTank, geom="histogram", bins=8)
```

Looking at these two histograms is quite informative. The SVL data appear almost normal, although the data have a slight tail to the right. The Age data are extremely skewed; most individuals emerged very early in the period of metamorphosis but the tail is very long with individuals still emerging from tanks more than 100 days after the first metamorphs. Many biological patterns can be made normal with log-transformation, so we should see what effect that has on our data. We will just look at SVL.final, but feel free to explore other variables as well. Data that can be made normal upon log-transformation

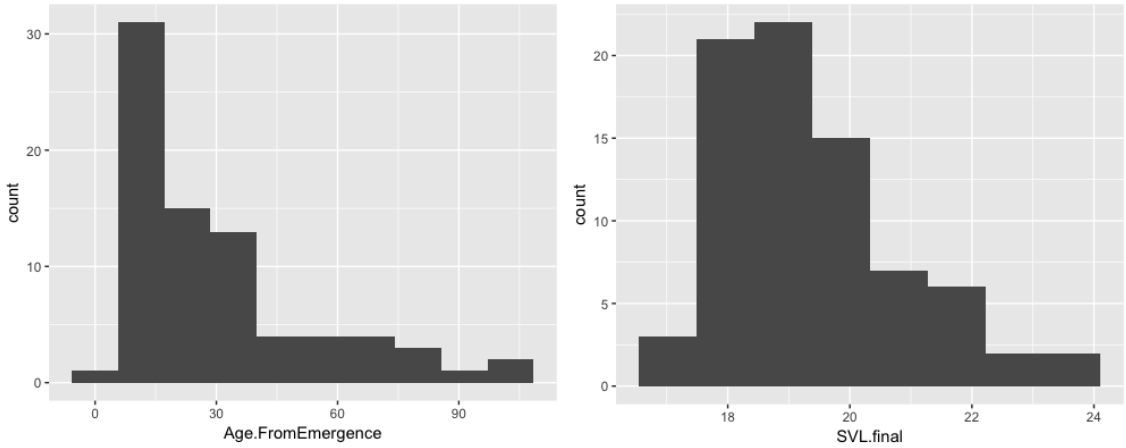

Figure 2: Histograms of SVL.final and Age.FromEmergence for *A. callidryas* metamorphs

are referred to as "lognormal". As we explored in Module 2, plotting data as a density plot can also be useful, and so both are shown below.

```
> qplot(x=SVL.final, data=RxP.byTank, geom="histogram", bins=8)
> qplot(x=log(SVL.final), data=RxP.byTank, geom="histogram", bins=8)
> qplot(x=SVL.final, data=RxP.byTank, geom="density")
> qplot(x=log(SVL.final), data=RxP.byTank, geom="density")
```

Log-transformation does not appear to have made a *huge* difference, but does look like it helped reduce the extended tail on the right side, making the data appear rather normal. That implies that maybe our data are indeed lognormally distributed. We can use a normality test to measure this effect statistically.

4.2 Normality tests

A classic way to identify if your data are suitable for parametric statistics is through the use of normality tests. These tests compare your dataset to a normal distribution with a similar mean and variance and assign a p-value of the probability that your data are normal. Smaller p-values (e.g., $p < 0.05$) indicate that the data are less likely to be normal. This is an important distinction: a small p-value does not mean your data "are not normal", it means that it is unlikely they come from a larger pool of data that are normally distributed. The most widely used normality test is the Shapiro-Wilks test, executed in `R` using the function `shapiro.test()`. This is an old function in `R` and does not accept the `data=` argument, so you have to just specify the vector of data you are interested. Recall that you can specify a particular column in a data frame with by using the `$` operator (i.e., `df$column`).

```
> shapiro.test(RxP.byTank$Age.FromEmergence)

  Shapiro-Wilk normality test

data:  RxP.byTank$Age.FromEmergence
```

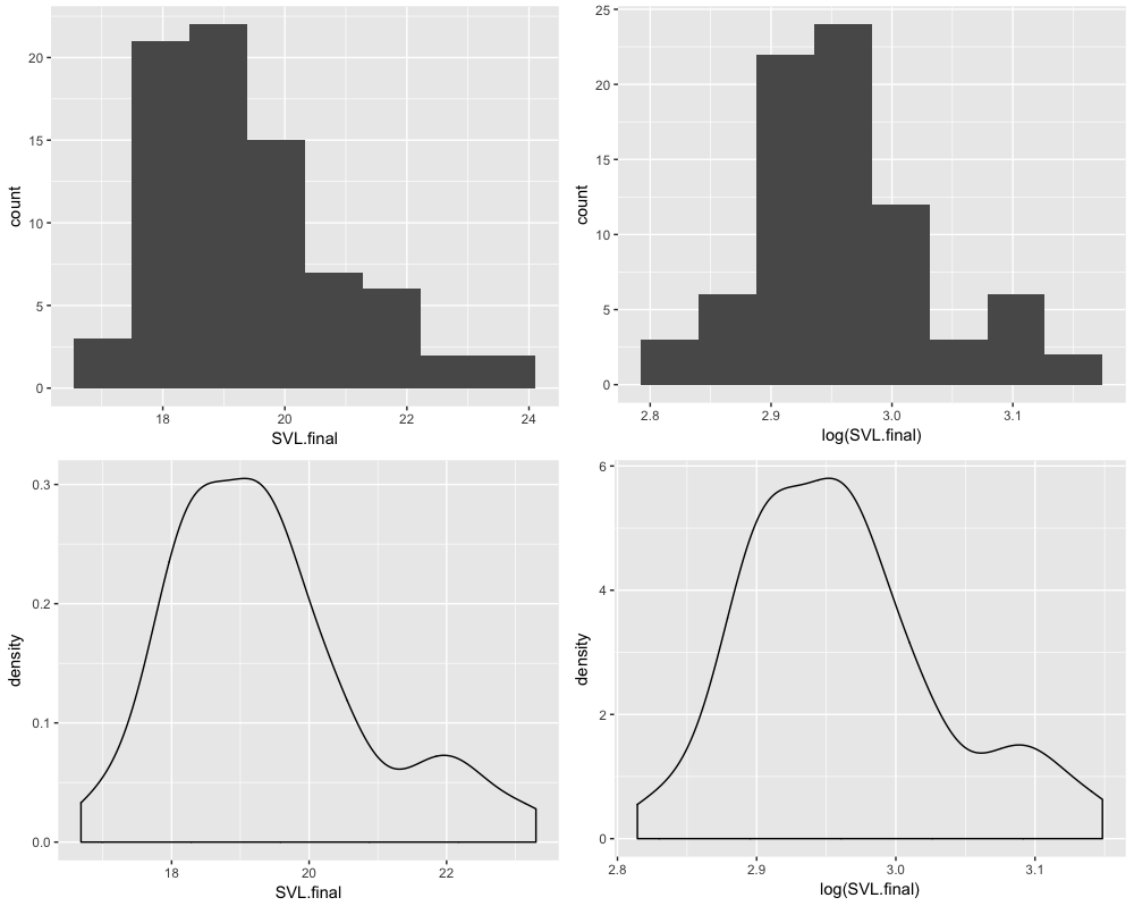

Figure 3: Histograms and density plots of SVL.final and log-transformed SVL.final for *A. callidryas* metamorphs

```
W = 0.8237, p-value = 3.143e-08

> shapiro.test(RxP.byTank$SVL.final)

Shapiro-Wilk normality test

data:  RxP.byTank$SVL.final
W = 0.9373, p-value = 0.0008234
```

The Shapiro-Wilks test indicates that neither of these variables are normally distributed. However, what about when we log-tranform them, as we did above?

```
> shapiro.test(log(RxP.byTank$Age.FromEmergence))

Shapiro-Wilk normality test
```

```
data:  log(RxP.byTank$Age.FromEmergence)
W = 0.973, p-value = 0.09666

> shapiro.test(log(RxP.byTank$SVL.final))

  Shapiro-Wilk normality test

data:  log(RxP.byTank$SVL.final)
W = 0.9542, p-value = 0.006943
```

Log-transformation of the tank-summarized data normalizes Age.FromEmergence and greatly improves the normality of SVL.final. Although the p-value is still substantially smaller than 0.05, clearly log-transformation has helped to normalize the SVL.final data. There are several other normality tests found within the `nortest` package if you wish to explore them.

4.3 How to estimate the most appropriate error distribution using maximum likelihood

Data in the real world are often messy, that is a fact of life for a biologist. This means that data often do not fall exactly within the confines of certain data classifications, such as "normal" vs. "non-normal". An alternative to normality tests is to simply estimate which predefined error distribution most accurately reflects your data. Thus, instead of trying to define what kind of data you have, you are instead trying to understand what your data are closest to. This can be done using the `fitdistr()` function (short for "fit distribution") found within the `MASS` package.

What is an error distribution you ask? In essence, it is a way of describing the shape of the data surrounding a mean (more or less). As we saw earlier, 'Normal' is an error distribution. It describes a dataset where the spread of data is relatively even above and below the mean. There are many different error distribution and we will explore them more in Module 5.

The `fitdistr()` function uses maximum likelihood to assess the fit of your data against a predefined error distribution and assigns an AIC score for the fit. This allows you to compare the fit of different error distributions using AIC. The full list of possible error distributions can be found in the help file for `fitdistr()`.

In short, AIC stands for Akaike's Information Criterion and is a measure of how well a model fits, i.e., how well a model explains the variance in the data. Three important things to know about AIC scores are 1) AIC scores are only comparable on the *exact* same response variable. 2) Although AIC scores are unitless, smaller AIC scores are better. 3) AIC values within 2 of one another are generally not considered to be pretty equivalent.

Let's continue to explore the `SVL.final` and `Age.FromEmergence` variables. We will create a new object for each `fitdistr()` object and compare them with the `AIC()` function. You may need to load the `MASS` package into the active memory using the `library()` command. For the full list of different possible error distributions you can choose from see the help file for `fitdistr()`. This function just takes two arguments: the vector of data you are examining and the distribution you want to compare it to.

```
> fit1<-fitdistr(RxP.byTank$SVL.final, "normal")
> fit2<-fitdistr(RxP.byTank$SVL.final, "lognormal")
```

```
> AIC(fit1,fit2)
     df      AIC
fit1  2 280.7438
fit2  2 276.4907
```

The AIC scores indicate that as suspected, the lognormal distribution is a better fit to the `SVL.final` data than a normal error distribution. Similar to what we saw with the histograms, the effect is not huge, but lognormal is slightly better. The original data on `Age.FromEmergence` are integer values and the long tail indicates that the data might be Poisson or negative binomially distrbuted. However, now that we have taken tank averages, our values are no longer integers. `fitdistr()` requires integers for either Poisson or negative binomial distributions, so we will add a `round()` command to make the values whole numbers.

```
> fit1<-fitdistr(RxP.byTank$Age.FromEmergence, "normal")
> fit2<-fitdistr(RxP.byTank$Age.FromEmergence, "lognormal")
> fit3<-fitdistr(round(RxP.byTank$Age.FromEmergence), "Poisson")
> fit4<-fitdistr(round(RxP.byTank$Age.FromEmergence), "negative binomial")
> AIC(fit1,fit2,fit3,fit4)
     df       AIC
fit1  2  718.7014
fit2  2  665.9925
fit3  1 1607.0213
fit4  2  675.6300
```

For `Age.FromEmergence`, the AIC scores once again indicate that the lognormal distribution is the best fitting error distribution. A negative binomial error distribution is also good, and substantially better than a normal distribution, but not quite as good as the lognormal. The Poisson distribution is by far the worst.

To make our lives easier, let's go ahead and create new variables in the `RxP.byTank` dataset that are the log-transformed values of SVL.final and Age.FromEmergence. To add new variables to an existing data frame, simple create a new object that refers to the data frame and `R` will automatically tack it on the end. While we are at it, we might as well also do `Age.DPO` too.

```
> RxP.byTank$log.SVL.final<-log(RxP.byTank$SVL.final)
> RxP.byTank$log.Age.FromEmergence<-log(RxP.byTank$Age.FromEmergence)
> RxP.byTank$log.Age.DPO<-log(RxP.byTank$Age.DPO)
```

You can confirm that this worked with the `str()` function.

5 Student's t-test

One of the simplest statistical analyses to conduct is Student's t-test, and there are multiple uses for the t-test (as well as a really cool backstory: `https://en.wikipedia.org/wiki/Student%27s_t-test#History`). The t-test is particularly useful for small sample sizes ($N < 30$). With a large sample size, the t-test is equivalent to a linear model (more on that below). With a t-test, you can:

1. Compare the means of two independent groups of normally distributed data.

2. Compare the means of two groups of paired data (e.g., before and after measurements).
3. Compare the mean of a single set of data with a hypothetical mean (e.g., from a previous study).

The function to conduct a t-test in R is simply `t.test()`. Although there are several ways to code the model, the most straightforward is what we presented above `t.test(response~predictor)`. For example, maybe we would like to know if there is a significant difference in the final SVL of metamorphs based on the resource treatment they experienced as tadpoles. One might expect that tadpoles fed more would grow to a larger size. In the code below, notice that we no longer call each variable directly from the data frame using the $ operator, but instead we use `data=` to specify where the model should look for any variables we give it.

Note that by default R will assume that the variances in your two groups are unequal and will adjust the degrees of freedom accordingly. This can be confusing for some folks because it will give slightly different results than what you might calculate by hand. If you are reasonably sure the variances are similar, add the argument `var.equal=T`, which will conduct a straightforward t-test like you were taught in school.

```
> t.test(log.SVL.final~Res, data=RxP.byTank, var.equal=T)

Two Sample t-test

data:  log.SVL.final by Res
t = 1.4315, df = 76, p-value = 0.1564
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -0.009104467  0.055640861
sample estimates:
mean in group Hi mean in group Lo
        2.973323         2.950055
```

What does all of that output tell us? First, R tells us what we did (a two sample t-test) and reminds us of what we are analyzing, the log.SVL.final variable by the Resource treatment. Next, we get the t-statistic, the degrees of freedom, and the p-value for the test. The p-value tells us probability of the H_0, that the difference in the means between our two groups are not different (the flip side is the shown H_1, that the difference in the means is not 0). Since the p-value is greater than 0.05, by convention we take this as evidence that *it is somewhat probable* that the difference in the means is actually 0. Most people would just say "they are not different" which is technically incorrect but practically correct in most cases. The 95% confidence interval is the upper and lower limits on what the model estimates as the difference in the means. You can see that it spans 0, and thus our belief that there is no practical difference between the two groups.

Thus, we can conclude that the mean final SVL of metamorphs did not differ if the rearing tank received high or low resource levels. The last two lines of the output tell us the means of each group. Since we log-transformed the data, we can use the `exp()` function to exponentiate the values back into real values (mm).

```
> exp(2.973323)#mean of the Hi treatment
[1] 19.5568
> exp(2.950055)#mean of the Lo treatment
[1] 19.107
```

So the means are only slightly different; metamorphs from the High resource tanks were just over 0.4 mm longer, but they are not signficantly different. How do you think this would have been different if we had not averaged the values for each tank? We can similarly test if the Age.FromEmergence differs based on Resource treatment, or perhaps Hatching treatment.

```
> t.test(log.Age.FromEmergence~Res, data=RxP.byTank, var.equal=T)

Two Sample t-test

data:  log.Age.FromEmergence by Res
t = -4.6246, df = 76, p-value = 1.511e-05
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -0.9493877 -0.3778055
sample estimates:
mean in group Hi mean in group Lo
        2.835664         3.499261

> exp(2.835664);exp(3.499261)#Means of Hi and Lo resource levels
[1] 17.04171
[1] 33.09099

> t.test(log.Age.FromEmergence~Hatch, data=RxP.byTank, var.equal=T)

Two Sample t-test

data:  log.Age.FromEmergence by Hatch
t = -1.0807, df = 76, p-value = 0.2832
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -0.4952698  0.1468414
sample estimates:
mean in group E mean in group L
       3.080356        3.254570

> exp(3.080356);exp(3.254570)#Means of Early and Late hatching ages
[1] 21.76615
[1] 25.90847
```

It looks like Resource level had a really strong influence on the time needed to reach metamorphosis. After exponentiating the mean values, we see that metamorphs from the Low resource tanks needed nearly twice as long, on average, to get out of the water, in comparison to individuals from the High resource tanks (remember that these data are age at emergence *after the first metamorph had emerged*). Hatching age, on the other hand, did not appear to have as strong effect on age at metamorphosis (just a 4 day difference).

One thing we should think about is if it is even appropriate to use the t-test in this scenario. As mentioned above, with a large enough sample size a t-test becomes numerically identical to a linear

model. The linear model is the common framework for conducting models commonly referred to as ANOVA, ANCOVA, and linear regression. Since we have 78 tanks (i.e., 78 independent observations) our sample size is certainly large enough to warrant using a linear model.

6 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. This flexibility leads the model to be called the *general* linear model by some. This is a bit confusing with the *generalized* linear model which we will discuss in Module 5, and so we will use R's preferred lingo, the linear model. The constraint on a linear model is that the response variable must be normally distributed, but the predictor variable (or variables) can be either continuous or discrete (i.e., categorical). See Table 1 for the lay terms which are commonly applied to linear models with various combinations of predictor variables.

6.1 Useful functions for linear models

There are several functions that are very useful for any kind of linear model. These include the following:

- `summary()` - Summarizes your model and gives you important information such as adjusted R-squared and treatment means, as well as the F-statistic and p-value for the model. For linear regression and ANCOVA, provides slope and intercept of best fit regressions.
- `anova()` - Provides a very brief summary of the overall model but does not provide information on individual levels within factors. Statistical significance of each predictor is calculated in a stepwise manner, adding each factor one-by-one.
- `Anova()` - Provides similar summary information to the `anova()` function, but statistical significance of each predictor is calculated assuming all other factors are in the model. Found within the `car` package. Completely identical to `anova()` for most models, but is better in some cases, such as for ANCOVA.
- `plot()` - Provides diagnostic plots of the residuals of the model. Useful for assessing model fit and balance.

6.2 One-way analysis of variance - ANOVA

One of the most common types of statistical analyses is the Analysis of Variance, called ANOVA for short. In its simplest form, an ANOVA is a statistical test of whether or not the means of multiple (more than two) groups are equal. Thus, it is essentially the same as the t-test but for more than two groups (in fact, you can use this to do the exact same hypothesis test as the t-test). The resulting p-value and test statistic give us some idea of the probability that those group means are indeed different. For example, perhaps we want to know if SVL.final differs across our three Predator treatments (Control, Nonlethal, and Lethal). Here, we will actually make our model into an `R` object and then examine it using the `summary()` function. Note that we could have done that in the t-test example, we just chose not to. Also, remember to use the log-transformed version of your variable so that it is normally distributed (or at least relatively close to it).

```
> lm1<-lm(log.SVL.final~Pred, data=RxP.byTank)
> summary(lm1)

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

Residuals:
      Min        1Q    Median        3Q       Max
-0.125260 -0.043283 -0.001346  0.036501  0.133130

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 2.914801   0.009924 293.715  < 2e-16 ***
PredL       0.100521   0.014035   7.162  4.6e-10 ***
PredNL      0.031468   0.017989   1.749   0.0843 .
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1

Residual standard error: 0.05614 on 75 degrees of freedom
Multiple R-squared:  0.4122,  Adjusted R-squared:  0.3965
F-statistic: 26.29 on 2 and 75 DF,  p-value: 2.223e-09
```

So what all does that mean? We created an object called `lm1` and then used the `summary()` function to look at it. The top of the summary output tells us the model we made and gives us information about the distribution of the residuals around the mean. The section entitled `Coefficients` provides information about the estimated mean for each level (C, L and NL) within the factor (Pred). The way `R` reports estimates of means is that a baseline is established alphabetically, and everything else is in relation to that baseline. Thus, in the output for `lm1`, what is labeled as `(Intercept)` is the mean for the Control treatment (because C comes before L or NL, alphabetically). All of the values reported are *on the scale of your response variable*. In this case, that means the numbers are log-transformed and to make sense of them we have to exponentiate them. Thus, the SVL of metamorphs from the Control tanks is:

```
> exp(2.914801)
```

or 18.45 mm. The subsequent `Estimates` are how much those treatments differ from the baseline. Thus, the `PredL` level is 0.100521 larger than the Control treatment, or

```
> exp(2.914801 + 0.100521)
```

or 20.40 mm. Lastly, the SVL of metamorphs from the NL treatments are the last line, and are

```
> exp(2.914801 + 0.031468)
```

or 19.03 mm. The bottom three lines of the output provide the summary statistics that you might include in a manuscript. The adjusted R^2, the F-statistic, the degrees of freedom, and the p-value for the ANOVA. (Just FYI, you should report the adjusted R^2 and not the multiple R^2, as the adjusted R^2 is more conservative and has been *adjusted* for the number of predictors in your model). As you can see,

the Predator treatment had a very significant effect on metamorph SVL. We can confirm the treatment means with the `tapply()` or `aggregate()` functions. We can even nest that within the `exp()` function to expedite transforming our values back into the real scale of mm.

```
> exp(tapply(RxP.byTank$log.SVL.final, RxP.byTank$Pred, mean))
       C        L       NL
18.44515 20.39568 19.03482
```

Another way to view the summary statistics for the model as a whole is with the `anova()` function, which gives just the necessary info that you might include in a manuscript: degrees of freedom (both numerator and denominator), F-statistic and p-value.

```
> anova(lm1)
Analysis of Variance Table

Response: log.SVL.final
          Df  Sum Sq  Mean Sq F value    Pr(>F)
Pred       2 0.16573 0.082865  26.294 2.223e-09 ***
Residuals 75 0.23636 0.003151
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1
```

6.2.1 Diagnostic plots

It is always a good idea to look at the diagnostic plots for your model. By default, the `plot()` function will give you four diagnostic plots of your model.

- The residuals plotted against the fitted values (treatment means). This is used to check for consistency of variance across your treatments.
- A quantile-quantile normal distribution plot (called a Q-Q plot for short), plotting standardized residuals against the theoretical quantiles from a normal distribution. This is used to check for normality of errors.
- The square root of the absolute value of the standardized residuals plotted against the fitted values (treatment means). Similar to the first plot, this is used to check for constancy of errors across treatments.
- "Cook's distance", which is a measure of the influence of each observation on the parameter estimates.

```
> plot(lm1)
```

These plots look good. The first and third show a relatively even spread of points across our three threatment means. The second plot shows a good similarlity between our standardized residuals and the straight line derived from a normal distribution. This indicates good model fit. If the points in the Q-Q plot deviate considerably from the line, your model does not fit a normal distribution well. The last plot does not show us much in this case, but that is good. If there were points with too much leverage, it would be obvious.

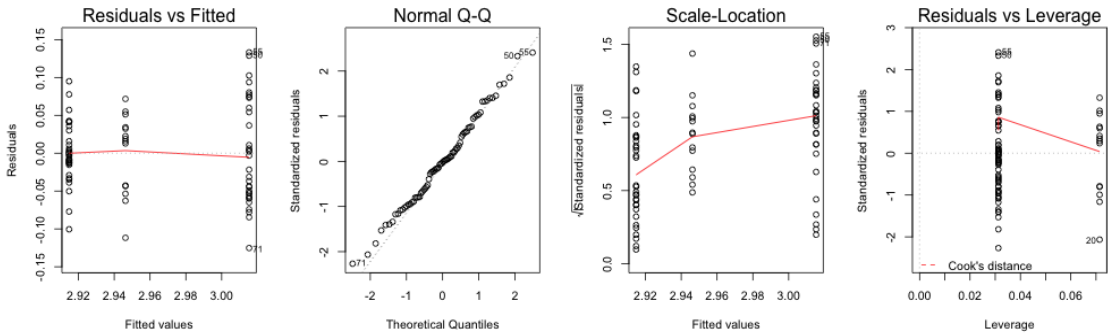

Figure 4: Four diagnostic plots of model `lm1`

6.2.2 A note of caution

Looking back at the summary output for `lm1`, notice that the right side of the `Coefficients` section shows several p-values. It is tempting to look at these and think "wow, my model is really significant!" That would be incorrect for several reasons. The first line, which above shows a t-value of 293.715 and an extremely small p-value, is simply a measure of the difference between the baseline level (here the Control treatment) and zero. The subsequent lines are essentially pairwise t-tests comparing the baseline (C) and the given treatment level (L or NL). However, the p-values reported here are not adjusted for the fact that multiple tests are being conducted, and therefore should not be used as such. Furthermore, it does not tell you all pairwise comparisons (i.e., the comparison of NL and L treatments is missing), which makes it unhelpful.

6.2.3 Multiple comparisons

It is often useful (and necessary) to be able to compare the different levels within a single categorical variable. For example, in the example here we might want to know how our three Predator treatments differ from one another. This is accomplished with the `glht()` function in the `multcomp` package. `multcomp` is short for "multiple comparisons" and `glht()` stands for "general linear hypothesis test". Again, we will make an object and then used the `summary()` function to view it.

```
> ph1<-glht(lm1, linfct=mcp(Pred="Tukey"))
> summary(ph1)

   Simultaneous Tests for General Linear Hypotheses

Multiple Comparisons of Means: Tukey Contrasts


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

Linear Hypotheses:
            Estimate Std. Error t value Pr(>|t|)
L - C == 0   0.10052    0.01403   7.162  < 1e-04 ***
```

```
NL - C == 0  0.03147    0.01799   1.749 0.191766
NL - L == 0 -0.06905    0.01799  -3.839 0.000671 ***
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1
(Adjusted p values reported -- single-step method)
```

Here, we have used `glht()` to conduct a Tukey test (all pairwise comparisons) of the three treatments within our Predator treatment. The post-hoc comparisons here allow us to conclude that log.SVL.final differs significantly for Lethal vs Control treatments, and Lethal vs Nonlethal treatment, but that Control and Nonlethal treatments did not differ. You can see that the reported t-statistics for the comparisons between L and C treatments, or NL and C treatments, are the same as those listed in the summary output of the linear model. However, here the p-values are adjusted for the multiple tests being run, and so are more appropriate to use. You can also use `glht()` to conduct a Dunnett's test (comparing treatment against a single control) or any other specific pairwise comparison you desire. A handy function (particularly if you have many treatments) is `cld()`, which stands for "compact letter display". It gives you a very simple depiction of which treatments are statisically different from one another.

```
> cld(ph1)
  C   L  NL
"a" "b" "a"
```

Treatments with the same letter are not different from one another. Thus, in this case, we can see that C and NL are not different (both are `"a"`) and both differ from L (which is `"b"`).

7 Assignment!

Here are some things to do on your own, to build on the analysis and plotting skills you have just been working on.

1. For this weeks assignment, you are to conduct a series of one-way ANOVAs, analyzing the final Mass of metamorphs by each of the three predictors in the dataset Hatching age, Predator treatment, and Resource level and write a *very* brief report about it.
    - Make sure you check the final mass data for normality.
    - Make sure you check the diagnostic plots to see if things look okay or really out of whack (if the model really doesn't fit well, it should be obvious).
    - Make a bar graph to go with the analysis of each individual predictor.
    - Thus, for each ANOVA, you should give me a figure and report the F-statistic, R^2 value, and P-value. For the analysis of Predator treatment, conduct a post-hoc Tukey test to find out which of the three treatments are different from one another. Also, write a sentence or two interpreting your results.
2. Write a *very* short summary paragraph about this process. Which predictors were significant or not? Do you think it is appropriate to analyze the predictors independently? Tell me any other thoughts you have about this process.

Turn in to me (via Moodle) a word doc or pdf with both figures. The assignment is due the evening of Wednesday 2/14/2018.