Lab 2: Mechanisms of Evolution

1. Getting Started

Make a new project in a new folder and open a new script.

It is standard practice to first load all libraries you will need. In this case, we will be using `ggplot2` again to make our graphs. We are also entering data into Google Sheets, so we will need `gsheet` to pull that data into R.

For this lab we will provide you with the directions of what to do by showing you the comments you should enter in you R script.

- Copy the comment lines into your script
- Enter the necessary code based on the previous lab. In any case where you have not seen the necessary code it is provided. **In cases where you should know the code already you should check assignment 1 if you need help.**

```r
# Load the libraries ggplot2 and gsheet
```

2. Get Data into R

We will use a similar procedure as the last lab to load data into R so that we can work with it.

Next, load in the data (i.e. tell R where your dataset is). Today's data is found at:

https://docs.google.com/spreadsheets/d/1t63u8_5onAk4PNK1bATVSGVxwOcMzvPLuIInP1UgBzE

```r
# Assign website address for the data to a variable
```

```r
# Load the data from google sheets and store in a variable
```

Once we have our data in, it is always good to check to make sure the data was imported correctly. To view ONLY the first few lines of the dataset use the command `head`.

```r
# Check your data looks right (first lines only)
head(snail_data)
```

3. Find The Average Snail Count in the Oystercatcher Data

We are interested in differences in population size between red and white snails over time. To graph the average snail population size over time, you first need to calculate the class averages for each snail type at each generation time for each experiment. Start with the oystercatcher data (experiment 1).

Because we are focusing on the oystercatcher experiment first, we need to filter out the data from the first experiment. One way to do this is the `filter` function found in the library `dplyr`.

```r
# Load the dplyr library to provide a method of filtering the data
library(dplyr)
```

Now filter the data.

```r
# Filter data: experiment is equal to 1
exp1 <- filter(snail_data, exp==1)
```

This code filters for the snails in experiment 1. The `==` checks for equality (a `=` is the same as `<-` and assigns a value to a variable - that is not what you want here).

We actually want to take the average of the class data for each experiment / generation / color combination.

- Get the rows of data for white snails in experiment 1 in generation 0.

```
# Filter data: experiment 1, gen 0, col white
exp1gen0colW <- filter(snail_data, exp==1 & generation==0 & snailcolor=="white")
```

- When you use words in R (e.g. "white"), put the word in quotes. Words that are not in quotes are interpreted as variables. There is no 'white' variable.
- Calculate the average (mean) number of snails for these rows

```
# Calculate mean of the snails column of exp1gen0colW
mean(exp1gen0colW$snails)
```

Here we asking R to give us the mean of the subset of snails we found earlier. The `$` in the code points R to the the column of our data that we are interested in.

We could keep doing this for each experiment, at each generation time, for each color of snail. Now I bet you're saying "Wait…there are 4 generation times, two colors, and two experiments, which would mean 16 lines of code!…this will take forever!", and you'd be correct! Let's let the computer do all this calculating for us.

4. Automatically perform calculations on groups of data

Rather than calculate the mean for each group manually we will use the `group_by` function from `dplyr` to split the data into groups. `group_by` takes the data, followed by each column you want to group by.

```
# Group data by exp, generation, snailcolor
grouped_snail_data <- group_by(snail_data, exp, generation, snailcolor)
```

Next we calculate the mean for each group, and put this into a new data table using the `summarise` function. This function takes the data and the value you want to calculate. The latter is assigned to a column name using `=`. Here the name of the column containing the mean number of snails (calculated using the function `mean`) is "mean".

```
# Calculate the mean number of snails for each group
snail_data_means <- summarise(grouped_snail_data,mean=mean(snails))
```

Check the first few lines of your data using `head`.

```
head(snail_data_means)
```

5. Graphing the OysterCatcher Data

Just like last time, we are going to use `ggplot` to graph the data. (Remember, `ggplot2` is a library you load; `ggplot` is the function you use for graphing).

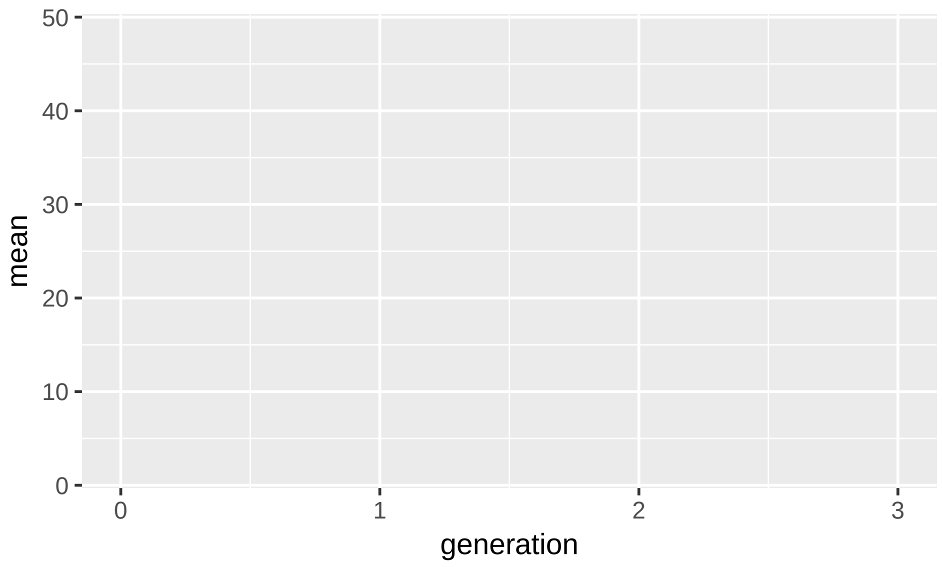
- Filter your new dataset (containing the means) for just experiment 1 (otherwise, you would graph both the oystercatcher and the driftlog data, making a very confusing graph to look at).
- Assign the filtered data to a new variable.

```
# Filter data to include only averages for exp 1
```

5a. Create the base layer of your plot

- Use your new filtered dataset

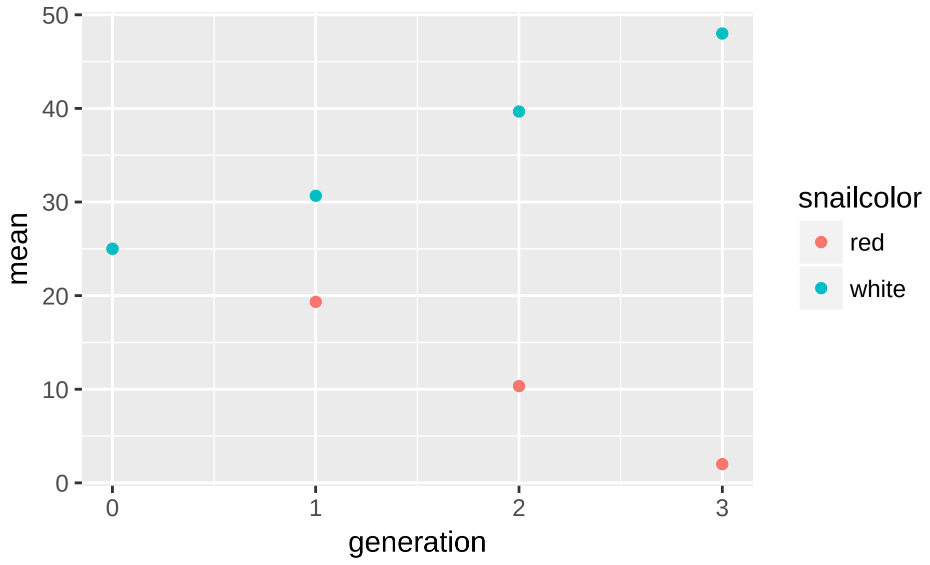
```r
# Make the base layer of your plot
ggplot(snail_data_means_ex1,
       aes(x=generation, y=mean, color=snailcolor))
```



Notice the addition of `color` to aes. The `color` variable separates the data by each value of `snailcolor` and then plots these separately on the same graph.

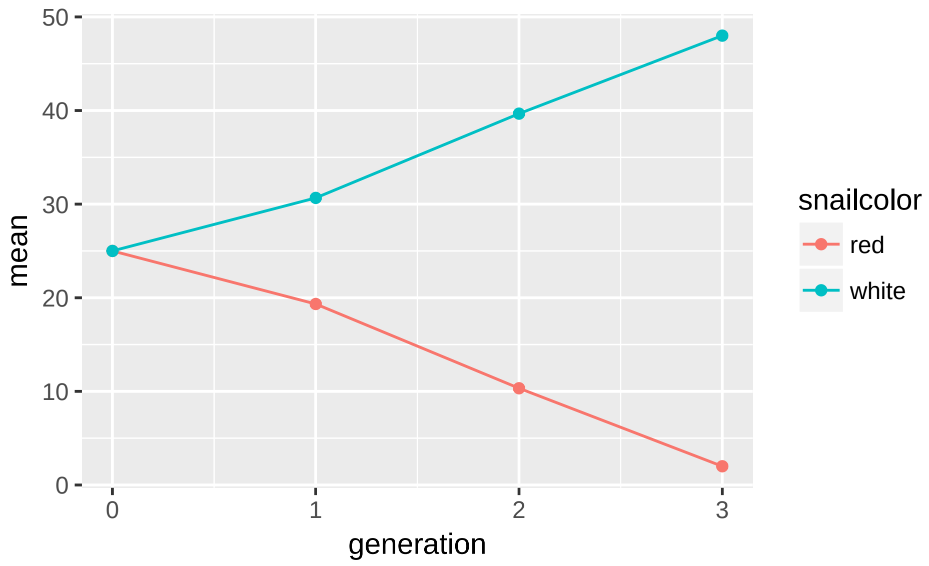
5b. Add the points to your plot

- Make your comments relevant to your graph



5c. Add a line to your plot to connect points

The layer to add a line is `geom_line()` and it is added just like you add `geom_point()`

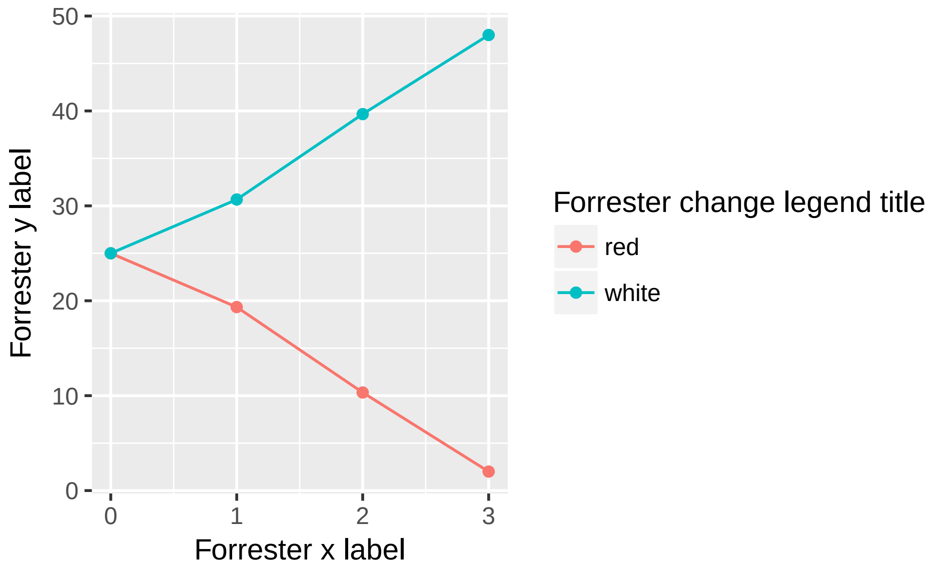


Note: `geom_line()` and `geom_point()` both are followed by empty parentheses. Because each `geom` is a function it can take arguments (i.e. particular instructions or information). These arguments need to go in the parentheses associated with the function. In this case, these functions do not take additional arguments, but we will see additional layers that do.

5d. Add labels to your plot and change axis titles

Just like last class, you'll want to clean up your graph and make it look professional.

- To label the axes make a new layer using `labs(x="", y="", color="")`. Enter labels for the x axis, y axis, and legend in the quotes. Note that x, y, and color are all arguments for the `labs` function.

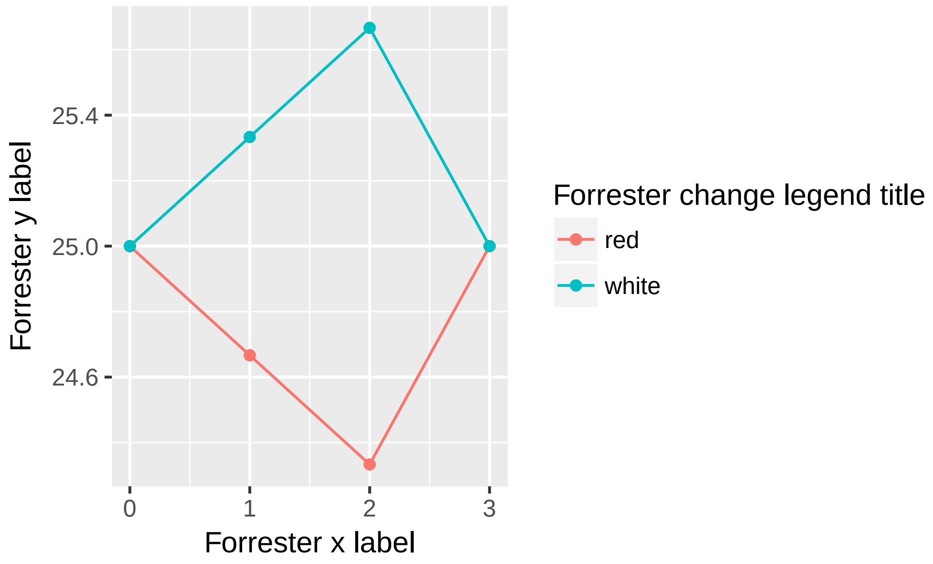


6. Graphing The Driftlog Experiment

Repeat everything you did for experiment 1 for experiment 2. Make sure your axis and legend titles are appropriate.

```
# Filter data to include only averages for exp 2

# Graph exp 2 data
```



Take a look at the scale of your plots. In many cases, your Oystercatcher data will have a much larger y axis than your driftwood experiment. Why do you think this is?

Your y-axis should start at 0. Also, to make a good comparison between experiments the y-axes of the two plots should be the same. To change your y-axis **add a layer `xlim(0, 50)`** (make sure to use the appropriate numbers for your data).

Note that these graphs do not show the variance in the data. We will show you how to add error bars in future labs.

6. Compare number of snails in different groups

Now that we have produced graphs that allow us to visualize our data, we are interested in knowing if the number of snails of each color differs. To compare the number of each color, you will run a *t*-test. A *t*-test is a statistical analysis that compares the means of two groups to see if they are statistically different from one another. To run a *t*-test in R, we will go back to the original data set.

First we filter our white and red snails from generation 3 in both the oystercatcher and driftlog experiments.

```
# Subset snail data by generation, experiment and snail color
# Red1 contains just data from exp 1 red snails
Red1 <- filter(snail_data, exp==1 & generation==3 & snailcolor=="red")

# White1 contains just data from exp 1 white snails

# Red2 contains just data from exp 2 red snails

# White2 contains just data from exp 2 white snails
```

Now that we have separate variables for each subset of data, we can run our *t*-tests! Input the two groups you want to compare, and then select the type of *t*-test to use. For this class, we will be using the two-sample *t*-test because we have two samples (a one sample test compares your data to some expected value).

What is your null hypothesis for the Oystercatcher data?

```r
# My null hypothesis is that red and white snail numbers are the same at the third generation of experi

# Oystercatcher_ttest
t.test(Red1$snails, White1$snails)
```

```
##
##  Welch Two Sample t-test
##
## data:  Red1$snails and White1$snails
## t = -56.338, df = 4, p-value = 5.943e-07
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -48.26696 -43.73304
## sample estimates:
## mean of x mean of y
##         2        48
```

What is your null hypothesis for the Driftlog data?

```r
# My null hypothesis is:

# Driftlog_ttest
t.test(Red2$snails, White2$snails)
```

```
##
##  Welch Two Sample t-test
##
## data:  Red2$snails and White2$snails
## t = 0, df = 4, p-value = 1
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -8.173633  8.173633
## sample estimates:
## mean of x mean of y
##        25        25
```

The output from `t.test` gives you a lot of information. For this class, we are going to focus on the p-value. A p-value is the probability of getting data as extreme or more extreme than the observed data given that the null hypothesis is true.

Imagine you are willing to accept a 5% probability that you reject the null hypothesis if it is really true. That's like saying for 20 experiments where the null is true, one of them will probably appear as if the null is false ($1/20 = 5\%$). Because you have some variance in your data you need to allow for some probability of being wrong. If you're having trouble with the concept of variance think about flipping a coin. Most of the time your coin flips will not be 50/50 heads/tails. But you still don't reject it being a fair coin. If your p-value is less than 0.05 you will reject the null hypothesis. If it is greater than 0.05 you cannot reject the null hypothesis.

For each experiment

- can you reject the null hypothesis?
- give a one sentence conclusion for each experiment