Lab 2: Mechanisms of Evolution

Loading Data Into R

We will use a similar procedure as the last lab to load R data.

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

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

Next, we will load in our data file. Today's data is found at: https://docs.google.com/spreadsheets/d/1t63u8_5onAk4PNK1bATVSGVxwOcMzvPLuIInP1UgBzE

```
#assign website address for the data to a variable
```

```
#Load the data from google sheets
```

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`.

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

1. 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, calculate the class averages for each snail type at each generation time. 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`.

```
#First load the dplyr library to provide a method of filtering the data
library(dplyr)
```

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

This code filters for the white snails in experiment 1. The `==` checks for equality (a `=` is the same as `<-` and assigns a value to a variable).

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")
```

- Remember when you point R to words rather than numbers ("white" vs 1), put the word in quotes.

Calculate the average (mean) number of snails for these rows

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

```
## [1] 25
```

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!

2. Automatically splitting data into groups

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, where 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`

```
## Source: local data frame [6 x 4]
## Groups: exp, generation [3]
##
##     exp generation snailcolor     mean
##   <int>      <int>      <chr>    <dbl>
## 1     1          0        red 25.00000
## 2     1          0      white 25.00000
## 3     1          1        red 19.33333
## 4     1          1      white 30.66667
## 5     1          2        red 10.33333
## 6     1          2      white 39.66667
```

3.Graphing the OysterCatcher Data

Just like last time, we are going to use ggplot to graph the data.

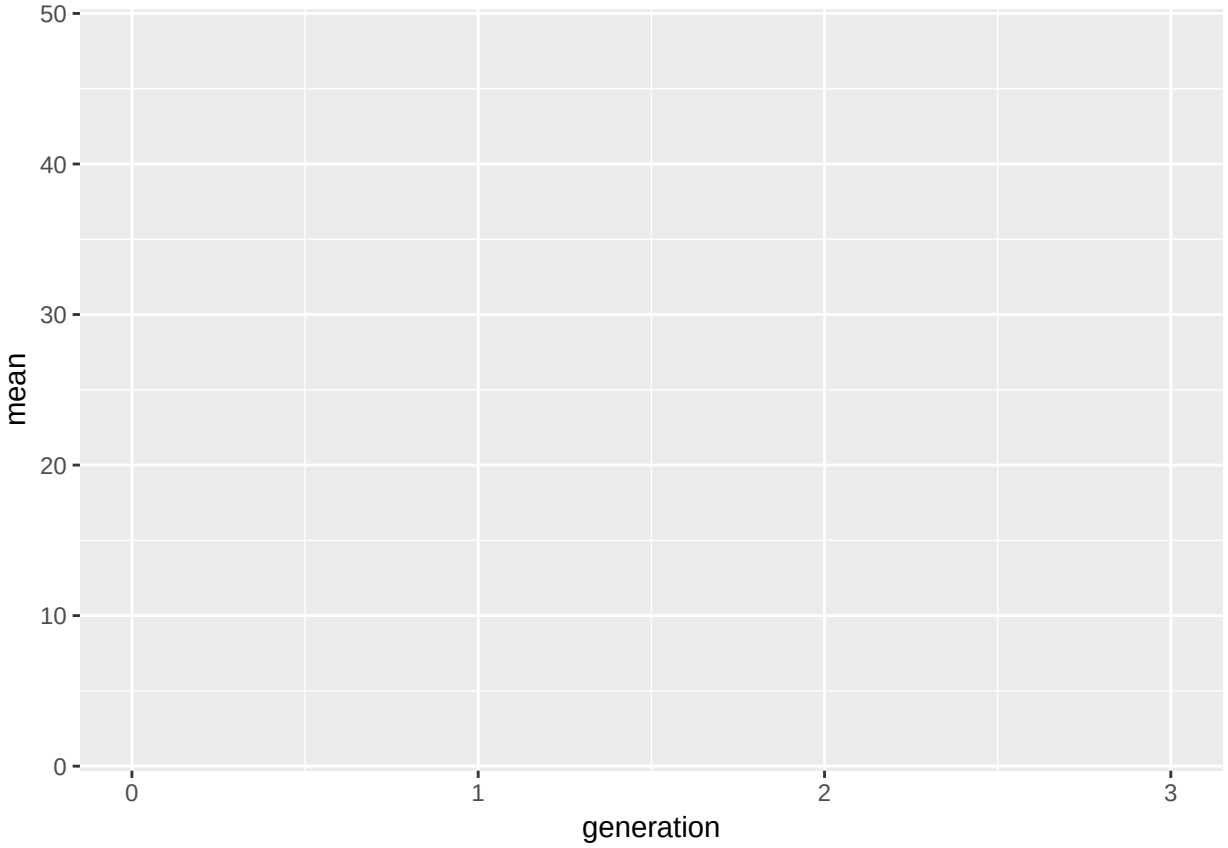
- filter your new dataset with averages for just experiment 1 (otherwise, you would graph both the oystercatcher and the driftlog data, making a very confusing graph to look at)
- assign the filtered data to a new variable

```
# get just averages for exp 1
```

First create the base layer of your plot

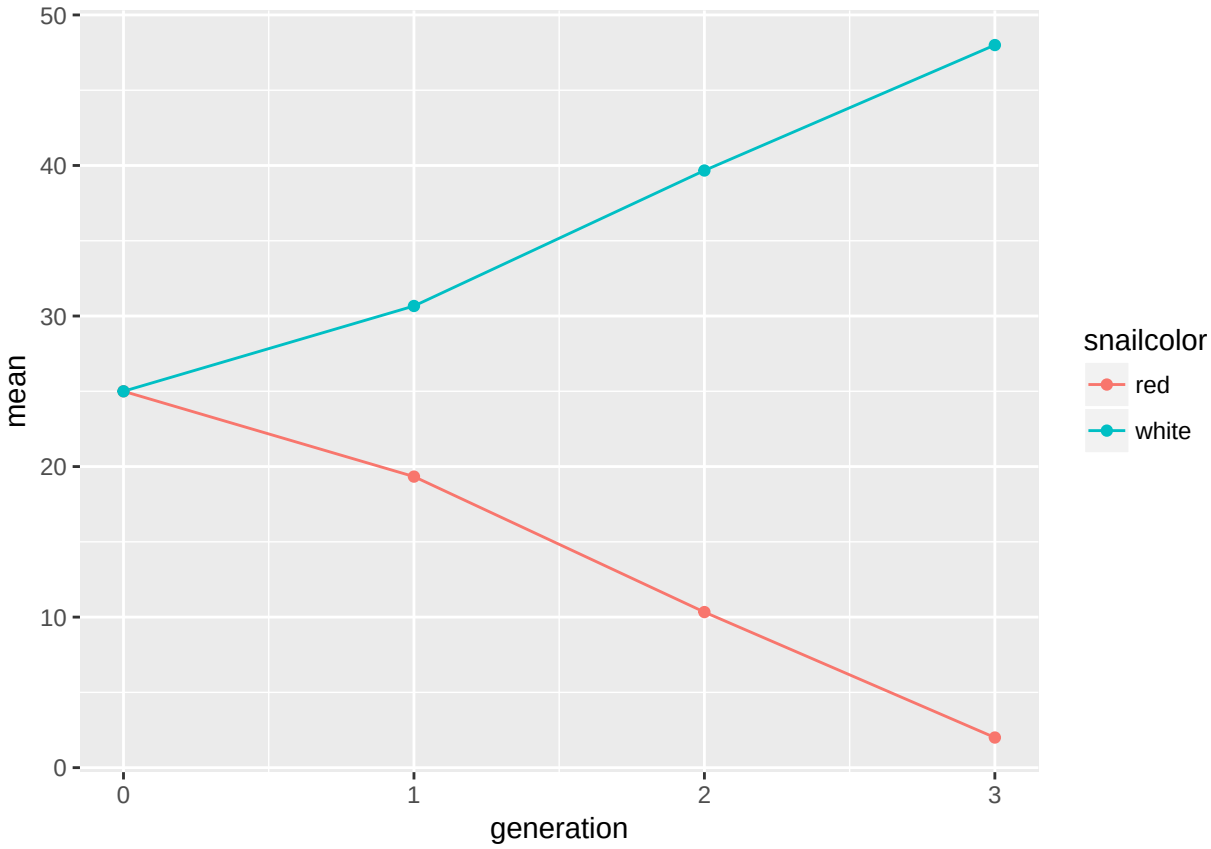
- use your new filtered dataset

```
#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. This gives your two lines different colors to easily differentiate between them.

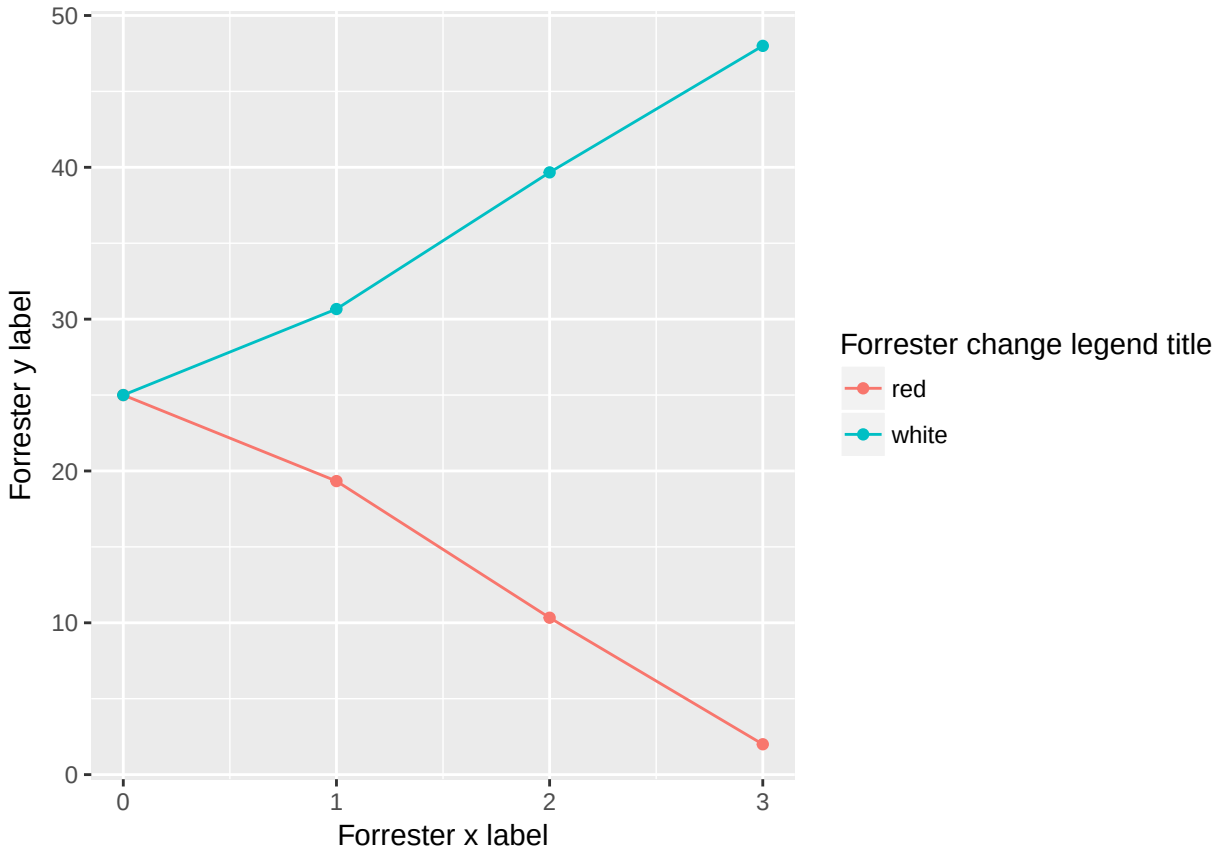
Second add the line and points to your plot



Third add labels to your plot and change axis titles

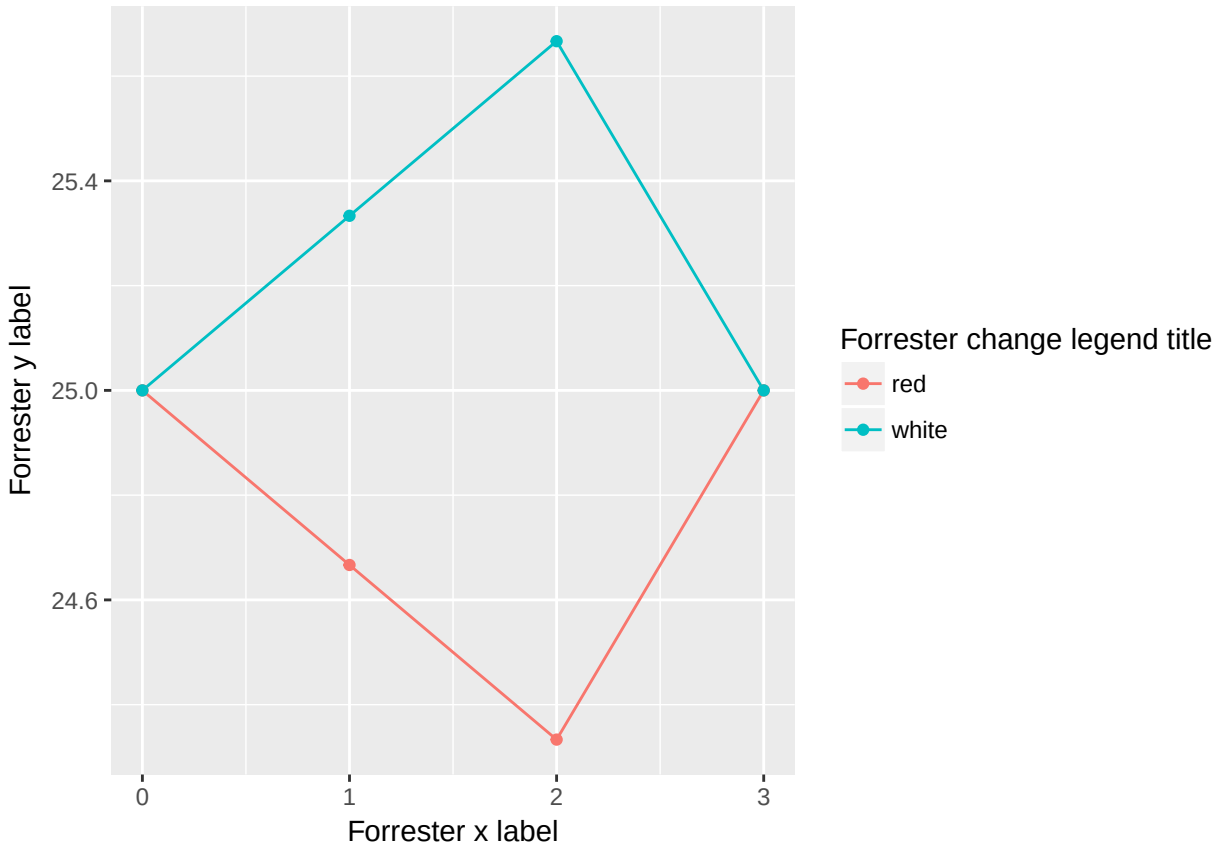
Just like last class, you'll want to clean up your graph and make it look professional.

- to label the axes make a new layer using `labs(x="", y="", color="")`.



4. Graphing The Driftlog Experiment

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



Remember to check 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?

5. Performing a T-test in R

Now that we have produced graphs that allow us to visualize our data, we are interested in knowing if the data between the two experiments is statistically different by the third generation. To do this, you'll 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 will have to 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 <- filter(snail_data, exp==1 & generation ==3 & snailcolor=="red")
White1 <- filter(snail_data, exp==1 & generation ==3 & snailcolor=="white")
Red2 <- filter(snail_data, exp==2 & generation ==3 & snailcolor=="red")
White2 <- filter(snail_data, exp==2 & generation ==3 & snailcolor=="white")
```

Now that we have all our variables separated out, we can run our t-tests! To do this, you 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. Make sure that you include the 'var.equal=True' or you will be running the wrong test!

```
#Oystercatcher_Ttest
t.test(Red1$snails, White1$snails, var.equal=TRUE)
```

```
##
##  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
```

```
#Driftlog_Ttest
t.test(Red2$snails, White2$snails, var.equal=TRUE)
```

```
##
##  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 the `t.test` will give 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.

Another way to explain this is that when a p-value is greater than 0.05, this suggests that your data SUPPORTS your null hypothesis, which is that the two groups are not different from each other. When the p-value is less than 0.05, this suggests that your data REJECTS your null hypothesis, indicating that the two groups are different from each other. What results did you get from the Oystercatcher data? How about the drift log data?