Assignment 2 Instructions

Linda Forrester, Rachel Schwartz. Markdown: Eren Ada

BIO104 Lab A2: Mechanisms of Evolution

Goals for Lab 2 Be able to...

- Draw line graphs of evolutionary mechanisms in action.
- Create code that you will understand in the future (i.e. by including comments)
- Calculate mean and standard deviation (using dplyr in R)
- Plot line graphs (using ggplot)
- Compare two datasets statistically (using a t-test in R)

0. Before starting with R, sketch your expected results Drawing line graphs to represent evolutionary change.

Line graphs are useful to represent change in a variable over time. The pace of evolution is not determined by absolute time, but by generations. That why our x-axis for evolutionary change graphs will be 'generation.' We want to compare the population size between red and white snails over generations to illustrate mechanisms of evolution.

A. Graph null hypothesis (Ho): Hardy-Weinberg equilibrium.

Start with a theoretical population of 100 snails. Assume equal frequency of white and red snails, so draw a graph starting with 50 white and 50 red snails. Draw a line graph showing what the frequency of each snail color will likely be over 5 generations **if no evolution is occurring.** This is our null hypothesis, which we call the Hardy-Weinberg equilibrium.

B. Graph evolutionary mechanism in action (HA)

Start again with a population of 100 snails, again half white and half red snails at generation 0. This time, **there is a predator** (a bird called an Oystercatcher) **that prefers to eat red snails to white snails.** What mechanism of evolution is this an example of?

Draw a line graph, showing your predictions for the frequency of each snail color over 5 generations if this mechanism of evolution were in action every generation. Write a 1-2 sentence description of how a mechanism of evolution is acting on color frequencies in the snail population.

1. Get started – new R project Make a new R project in a new directory and open a new R script.

Begin by loading all the libraries you will need. In this case, we will be using ggplot2 to make our graphs. We are also entering data into Google Sheets, so we will need gsheet to pull that data into R.

- Copy the comment lines into your script
- Enter the necessary code based on the previous lab and your R script from Lab A1. **If there are no code instructions then look at your Lab A1 materials**

```
# MyFullName, myTAname = learning linegraph & T-test in RStudio
# Load the packages ggplot2, gsheet, and dplyr using the library command
```

SAVE yourwork. Save your code by clicking `File>Save`

Name your file: Lab2LineGraphTtest_YourName.R

Remember to put ".R" at the end! Your work will save to the server that we are using for this lab. You will be able to access your work on this URI server from other computers.

2. Get your data into R Use the same procedure as lab 1 to load data into R. Go to the Lab Sakai site. Scroll down and select the link for your lab section.

```
# Get data into R. Store the Google sheets link as the variable url
# Load the data stored in the variable url
# using the gsheet2tbl function and storing it as the variable snail_data
```

```
library(gsheet)

#save url as variable
snail_url <- 'https://docs.google.com/spreadsheets/d/1OT3t5E6qbZqKUrW4PzpaagOaZRRBzK1NsfGh6Zwbiew/edit#g

#use gsheet2tbl function to store the data as variable
snail_data <- gsheet2tbl(snail_url)
```

Once you have your variables, check to make sure the data imported correctly. Click on the variables listed in the R Environment (top right) and compare them to the Google Sheet data. Another way to look at the first few lines of your data is using the head function:

```
head(snail_data)
```

```
## # A tibble: 6 x 5
##   group   exp generation snailcolor snails
##   <dbl> <dbl>      <dbl> <chr>       <dbl>
## 1     1     1          0 red            25
## 2     1     1          0 white          25
## 3     1     1          1 red            22
## 4     1     1          1 white          28
## 5     1     1          2 red            18
## 6     1     1          2 white          32
```

These are the column names of your raw data. Use these to analyze and plot the data.

- `exp` is the experiment number
- `group` is your group number
- `generation` is the number of generations from the beginning (0,1,2,3)
- `snailcolor` is white or red
- `snails` is the number of snails

You can also see the column names (i.e. the variables) by using the colnames function.

```
colnames(snail_data)
```

```
## [1] "group"      "exp"        "generation" "snailcolor" "snails"
```

3. Find the averages and standard deviations of the snail populations over time We want to compare the population size (`snails`) between red and white snails (`snailcolor`), over generations (`generation`), and experiments (`exp`). To show these differences, we need to graph the mean of the snail populations over time. We do this by calculating the mean for each snail type at each generation for each experiment. To do this, we need to group the data into appropriate categories. Since we are looking for the mean snail population at each generation for each color for both experiments, we will group our data based on experiment / generation / color combination.

We will use the `group_by` function from the `dplyr` package to create a new variable where each subset of data is labeled. Tell `group_by` the name of the data you want to divide into subsets, followed by the columns you want to include.

```
#load the dplyr package. If not installed, install it by typing install.package("dplyr")

library(dplyr)

# Group the snail_data variable by exp, generation, and snailcolor
# using the group_by function

grouped_snail_data <- group_by(snail_data, exp, generation, snailcolor)
```

Calculate the mean for each group, and put this into a new variable snail_data_means using the summarise function. 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 using the summarise function
# by giving the name of the data you want to summarize (`grouped_snail_data`)
# and that you want the mean value of the snail counts (`snails`)


snail_data_means <- summarise(grouped_snail_data,mean=mean(snails))
```

Click on the snail_data_means variable in the in the R environment (top right).

What are your mean values for the data? Check the values make sense before continuing.

3b. Calculate the standard deviation of the means

- The standard deviation is the average amount the individual data points differ from the overall mean. For example, if we saw red snail population totals of 3, 5, 5 and 6, we would have a low standard deviation since all the data is close to the mean. However, if we had red snail total of 0, 2, 12 and 16 the standard deviation would be high since all of the data is far from the mean.
- Standard deviation is calculated in R using the function sd.

```
# Remake your table of means so it includes standard deviation
# and store the new table as the variable snail_data_means_sd
snail_data_means_sd <- summarise(grouped_snail_data,mean = mean(snails), stdev = sd(snails))
```

(All the above code goes on ONE line in R. It starts a new line on this printed copy only.)

Click on the snail_data_means_sd variable in the R environment.

What are your standard deviation values for the data? Include your mean and SD values in your report.

4. Plot your data from Experiment 1

4a. Subset your data from Experiment 1 We will use `ggplot` to graph the mean and standard deviation for Experiment 1. (Remember, `ggplot2` is a package you load and `ggplot` is the function you use for graphing).

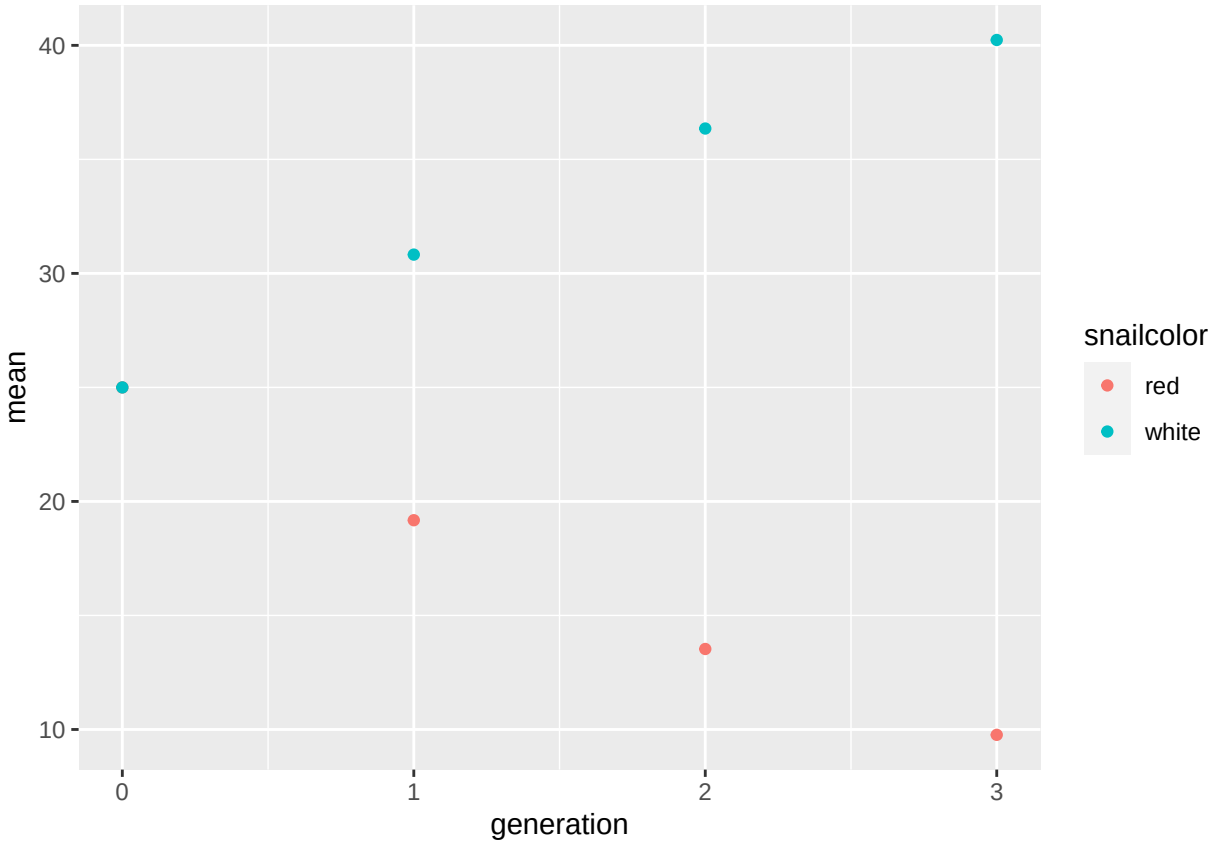
- First, filter your data variable that contains the mean and standard deviation (snail_data_means_sd) for just Experiment 1 (otherwise, you would graph both the data from all experiments, making a very confusing graph to look at).
- Assign the filtered data to a new variable named snail_data_means_sd_exp1.

```
#Filter data to include only means for Experiment 1 (exp==1).
# Refer to Lab A1, Section 5c for the commands.
```

4b. Create the base layer of your plot and add points

- Use your new filtered dataset (snail_data_means_sd_exp1)
- Add color to the aes function. The color variable separates the data based on snailcolor and then plots these separately on the same graph (in different colors!).

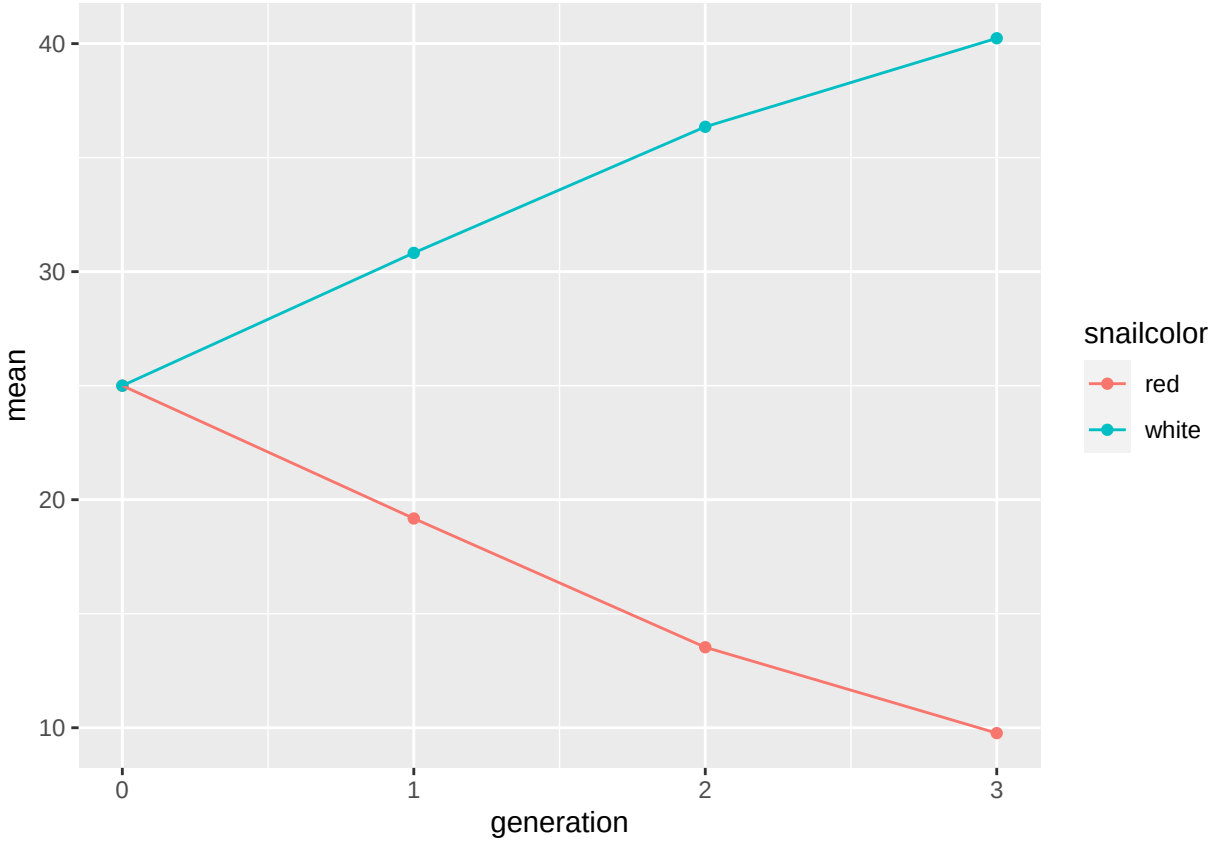
```
# Make your plot for the snail_data_means_sd_exp1 data and add points

library(ggplot2)

ggplot(snail_data_means_sd_exp1, aes(x=generation, y=mean, color=snailcolor))+
geom_point()
```



4c. Add a line to your plot to connect the points

- The function to add a line is `geom_line()` and it is added to the ggplot command just like you added `geom_point()`.

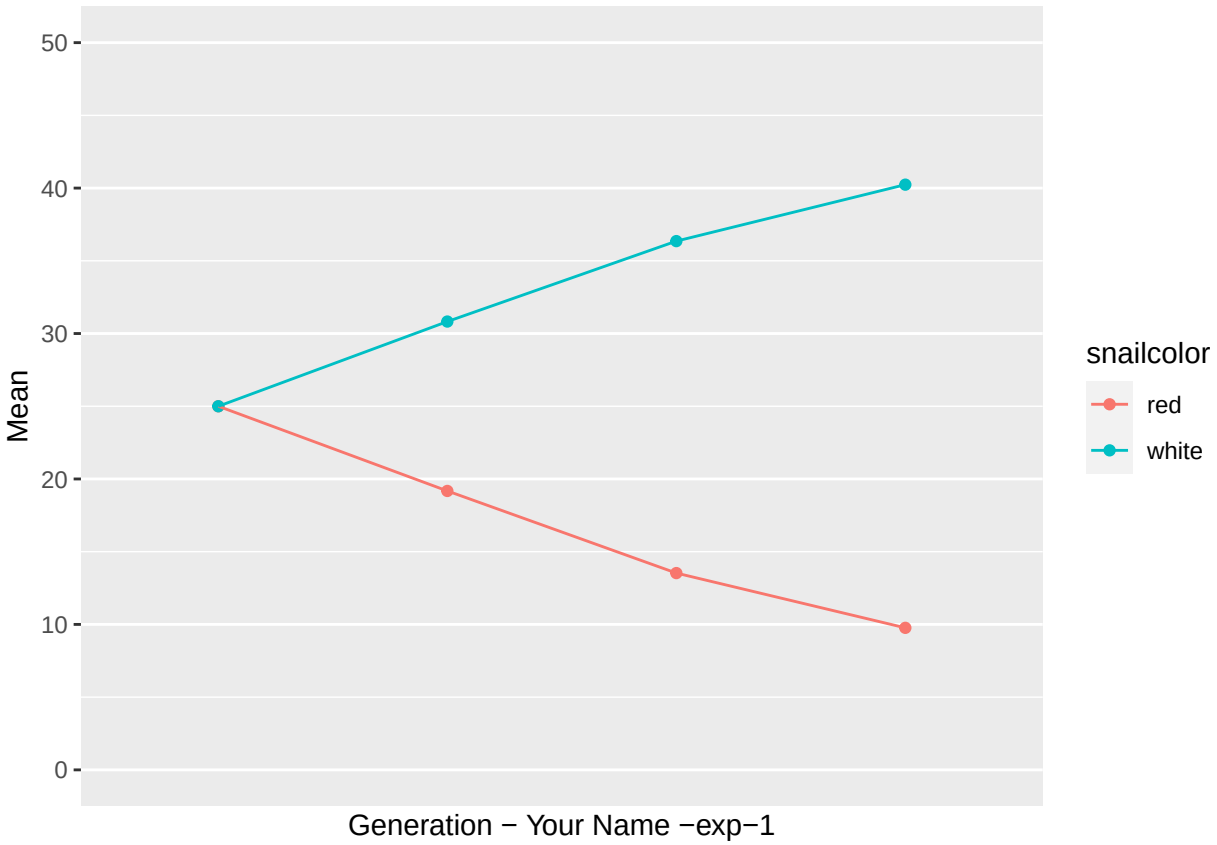
```
# Add a line to the plot

ggplot(snail_data_means_sd_exp1, aes(x=generation, y = mean, color = snailcolor)) +
  geom_point() + geom_line()
```



Note: `geom_line()` and `geom_point()` are followed by empty parentheses because they are functions. Functions must be able to accept arguments (e.g. the range of the data you would like to show). These arguments need to go in the parentheses associated with the function.

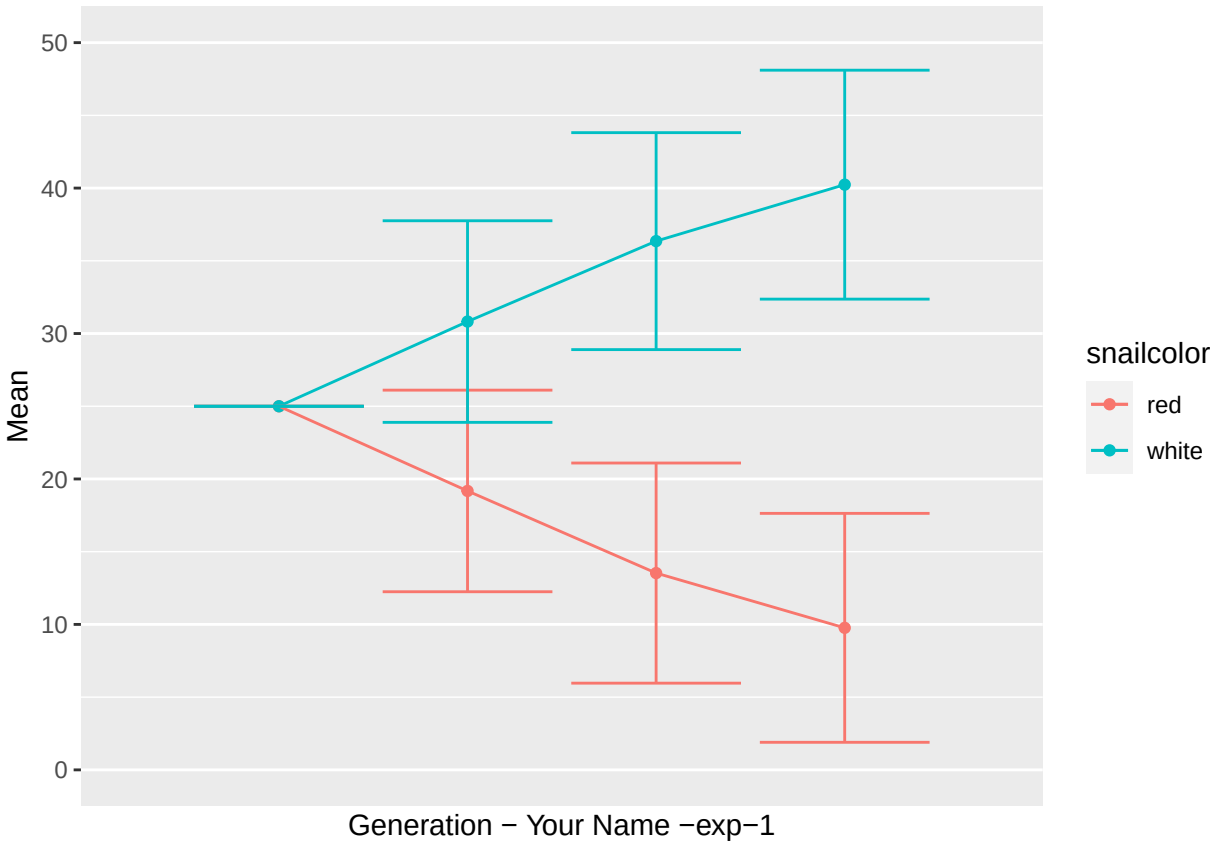
4d. Add labels to your plot and change axis titles

- Label the axes by adding the functions `scale_x_continuous` and `scale_y_continuous` after your code like you did in Lab 1, Section 5c. Use `name=" "` to name the x and y axes. Label the legend using `+labs(color="Snail Color")`.
- To make sure your `y-axis` starts at 0 and stops at 50, add "`limits = c(0,50)`" in the `scale_y_continuous` function. Note, the maximum for your y-axis may not be 50.



4e. Add standard deviation bars to your plot These graphs currently do not show the variance in your class's data like your boxplots in Lab A1. To show variation of our data in our line graphs, we add standard deviation bars by adding a layer using `geom_errorbar()`.

- geom_errorbar() draws an error bar that has an upper and lower value. In this case, the upper value is the mean + the standard deviation and the lower value is the mean-the standard deviation.
- Add + geom_errorbar(aes(ymin=mean+stdev, ymax=mean-stdev)) to your ggplot command.

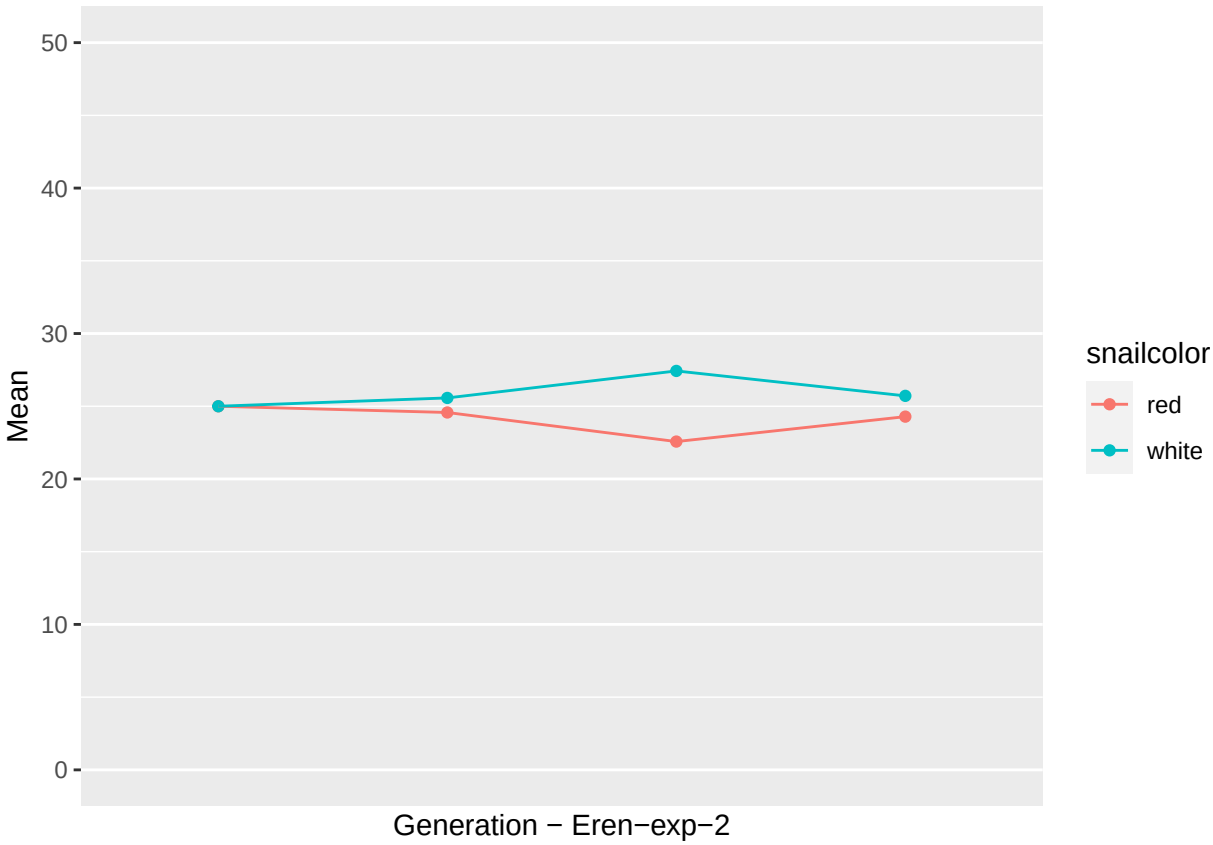


Examine your standard deviation bars visually. Do they overlap or are they far apart? Does this indicate that your means are different or similar to each other? Make sure to consider this as you write your lab report.

Save this plot as Lab2_Experiment1 so that you can turn it in with your lab report.

5. Plot your data from Experiment 2

- Repeat everything you did for Experiment 1 for Experiment 2.
- Make sure your axis and legend titles are correct, and that error bars are included.



Save this plot as Lab2_Experiment2 so that you can turn it in with your lab report.

6. Compare number of snails in different groups using a statistical test Now that we have examined our data visually, we are interested in knowing if the red and white snail populations are statistically different at the end of the experiment (generation3). To compare red and white snail populations at generation 3, you will run a t-test.

To run a t-test in R, we will use our original data set (snail_data) since the t-test takes into account the means of the data it is comparing, the sample sizes of the data it is comparing and the variation of the data it is comparing.

First, we filter our white and red snails from Generation 3 in both Experiments 1 and 2. Use the command from Red1 for Red2, White1, and White2.

Make sure you include quotes around your variable names ("red" or "white").

```
# Subset snail data by generation, experiment and snail color and store it in the Red1 variable.
# Red1 contains just data from experiment 1's, generation 3, red snails

Red1 <- filter(snail_data, exp==1 & generation==3 & snailcolor=="red")

# White1 contains just data on experiment 1's, generation 3, white snails # (write the
command for White1 here)

# Red2 contains just data from experiment 2's, generation 3 red snails

# White2 contains just data from exp 2's, generation 3, 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).

Red1, Red2, White1, and White2 contain much more data than just number of snails, like `generation`, `exp`, `group`, etc. However, we are only interested in comparing the number of snails. To extract the number of snails in Generation 3 from the Red1, Red2, White1, and White2 variables, we use the dollar sign `($)`. The dollar sign extracts items from a variable based on their names. For example: Red1$snails extracts the number of red snails in Generation 3 from Experiment 1.

What is your null hypothesis for the Experiment 1 data?

```
# My null hypothesis is:
# red and white snail populations are the same at the third generation of experiment
# Oystercatcher_ttest
t.test(Red1$snails, White1$snails)
```

```
##
##  Welch Two Sample t-test
##
## data:  Red1$snails and White1$snails
## t = -11.288, df = 32, p-value = 1.081e-12
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -35.96925 -24.97193
## sample estimates:
## mean of x mean of y
##  9.764706 40.235294
```

What is your null hypothesis for the Experiment 2 data?

```
# My null hypothesis is:
# Driftlog_ttest
t.test(Red2$snails, White2$snails)
```

```
##
##  Welch Two Sample t-test
##
## data:  Red2$snails and White2$snails
## t = -0.47976, df = 26, p-value = 0.6354
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -7.549275  4.692132
## sample estimates:
## mean of x mean of y
##  24.28571  25.71429
```

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 different from 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 every 20 experiments where the null hypothesis 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 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.

Reporting independent samples t-test data To report results, include three main ideas:

- Test type and use.

Example: "An independent-samples t-test was conducted to compare red and white snail populations after three generations of predation from oyster catchers."

- Significant differences between conditions.

Example: "There was a significant (not a significant) difference in red (mean= ___) and white snail (mean= ___) populations after three generations of predation from oyster catchers; p = ___ ."

- Describe your results in simple words explaining your result.

Example: "These results suggest that snail color really does have an effect on oyster catcher predation.

Make sure to include these three sentences in your results for your oyster catcher and your drift log experiment.

Lab 2 Report Submission Follow rubric. Submit a hard copy of your lab report that includes the following:

1) Hand-drawn line graphs from pre-lab section 0a & 0b with a description of the mechanism of evolution.

2) Summary table for your section's data (made in excel/sheets/or R)

3) Line graph of Experiment 1 means from Step 4d

4) Line graph of Experiment 2 means from Step 5

5) Results of your t-tests for both experiments from Step 6

6) Explain why you rejected or failed to reject your null hypotheses based on your t-test results and explain the processes that influenced both experiments using the format above.

7) Your code. Code should be organized, include good comments, and include only code that works and does not produce errors.