Lab 3: Hypotheses and significance testing

In this lab, you'll analyze data and interpret the resulting P-value. You'll learn how to manage data, including: extracting columns, naming columns, and extracting rows based on their contents. You'll also learn about three data types in R and logical operators. These are the new functions used:

```
data.frame()
colnames()
c()
```

There are 9 question checkpoints in **bold**. Please answer them directly in your R Markdown file, outside of the chunks.

Section 1: Importing data

1. Open a new R Markdown file.

2. Today we'll be using data from paper 2 of the course. Specifically, this is floral visitation data approximated from figure 1d, pictured below. A bee is counted when it enters a flower from the inflorescence, so as to pollinate it.

    - **Q1. We're going to test whether the sites of low and moderate pollution are significantly different in either direction. In other words, a positive result will mean that either the L site has significantly more visits than the M site OR that the M site has significantly more visits than the L site. A negative result will mean that there is no significant difference. In plain words without mathematical symbols or terms (no less than, greater than, equals to, etc.), what are the *complementary* null and alternative hypotheses for this test? Please label which is the null and which is the alternative.**
    - **Q2. Translate both hypotheses above to be written with inequality symbols. You can write out what goes on each side of the inequality, but use the symbols in between them. Again, please label the null and alternative.**

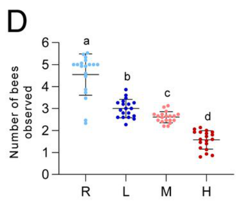

Figure 1: Number of bees observed, by site

3. Download the dataset from the course website (floral_visitation.csv) and import it. I'll name mine `all.data.df` and refer to it as such for the rest of the lab.

Three data types

Data in R comes in a variety of different types, but here are four important ones to understand. Data types are important because many functions will only take certain types as input, and different types will interact differently with one another.

- *Vectors:* Vectors are "one-dimensional". They contain one or more values in a specific order, so if you want to access a specific value, you can call it by the position that it's in. Values in a vector can also be named, so they can be called by their name if applicable.
- *Data Frames:* Data frames are like tables, with rows and columns. They're "two-dimensional", in that each column can be thought of as its own vector. Rows and columns also have a specific order and can be named, so you can access a specific row, column, or cell by calling the position by number or name.
- *Lists:* Lists are "three-dimensional". They contain a list of any combination of data types. You can even have a list inside of a list.

Section 2: Extracting and naming columns

4. The `View()` function is best for data frames and lists. For vectors and small data frames, you can simply type the variable name in the console and it will print. Since this is a larger data frame, take a look at the data in your new variable using either `View(all.data.df)` or by clicking on it in the top right panel. You should see 5 different columns, but today we only need two: `pollution` and `bees`. Let's make a new data frame that contains only these columns.
    - **Q3. Which column corresponds to the independent variable? Which column corresponds to the dependent variable?**
    - **Q4. Thinking back to the experimental design, what are 3 of the controlled variables? These may not be listed in the dataset, but you can go back to the paper or lectures if you need help.**

5. First, identify how to extract each individual column from `all.data.df`. Remember the four ways to access a column that we learned in the first lab, listed below. The first three methods will return the column as a vector, while the last method will return it as a data frame.
    - `all.data.df[ , COLUMN]` , where COLUMN is either a number or the name in quotes
    - `all.data.df$COLUMN` , where COLUMN can only be the column name, without quotes
    - `all.data.df[[COLUMN]]` , where COLUMN is either a number or the name in quotes
    - `all.data.df[COLUMN]` , where COLUMN is either a number or the name in quotes

6. The function `data.frame()` will create a data frame, which is the data type that will appear like a table to you; the function will need variables that will each act as column(s) in the overall data frame. In the parentheses, list each column using your preferred method, with commas in between each.

7. Assign the result of this function to a new variable, e.g. `clean.df <- data.frame( COLUMN1, COLUMN2 )` but with the contents of the `data.frame()` function filled in appropriately.

8. Take a look at your new variable to make sure it has the correct two columns, and only those columns.
    - **Q5. What default column names did R give to your data frame?**
    - Sometimes, the default column names from `data.frame()` can get pretty long. To rename columns and make them easier to work with, we need the `colnames()` function. Using `colnames()` on its own will return the column names of the input variable, but assigning values to `colnames()` can be used to change the column names in the same way we assign variables.

- Although we have two columns, we can only assign one input to `colnames()` , so we use the `c()` function to combine them. *Just make sure that the new names are in the correct order!*

9. Below is the final code for naming the columns, assuming your data frame from step 7 is called `clean.df`. Add this to your script if your columns don't already have these names.

```
colnames(clean.df) <- c("pollution", "bees")
```

Section 3: Extracting specific rows

10. If you look at your data frame, you should see that observations from the different locations are all mixed together, but we're only interested in testing the lowly- and moderately-polluted sites. Therefore, we'll need to separate out the appropriate rows based on the `pollution` column.
11. To do this, we need to use something called a logical operator, which is used to make comparison. Here's a list of the logical operators in R:

```
x == y # x is equal to y
x < y # x is less than y
x > y # x is greater than y
x <= y # x is less than or equal to y
x >= y # x is greater than or equal to y
x != y # x is not equal to y
!x # Not x
x & y # x and y
x | y # x or y
```

- With the exception of `!` (not), logical operators go between two terms to compare them, and return either `TRUE` or `FALSE`. For instance, `4 == 5` would return `FALSE`, but `4 == 4` would return `TRUE`.
- You can combine operators. For instance, `(4 == 5) | (4 == 4)` would return `TRUE` because at least one of the statements in the parentheses are true.

12. You can also compare variables. If both variables are vectors, R will compare the first position of each variable against each other, then the second position of each variable, then the third, and so on. If one vector only contains a single term and the other contains multiple, R will compare the single-term vector against each term in the other variable.
    - Therefore, in order to check which rows were observed at the low-pollution site, we'll compare the `pollution` column to the term `"low"`, as in any one of these three lines:

```
clean.df[ , "pollution" ] == "low"
clean.df[["pollution"]] == "low"
clean.df$pollution == "low"
```

13. Try running one of the lines above and see what you get as a result. Remember, R is comparing `"low"` to each term in the column and returning the result in the same order as the terms, so each result will correspond to a row in your dataset.

14. So now that we have a vector of results telling us which rows come from a low-pollution site, we need to use it to make a new data frame with only those rows. We'll do this by extracting rows as we already know how to do (`data.frame.name[row, ]`), but we'll feed in our new TRUE/FALSE logical vector as a guide to extract all TRUE rows. We'll also assign this to a new variable name, `low.df`. Putting everything together looks like this:

```r
low.df <- clean.df[ clean.df$pollution == "low", ]
```

15. Using variables, we can make this a little easier to understand, as seen below. Use whichever you feel more comfortable with (above or below) and add it to your code.

```r
pollution.lvl <- clean.df$pollution
low.pollution <- pollution.lvl == "low"
low.df <- clean.df[ low.pollution, ]
```

16. Take a look at `low.df` and make sure it only contains rows with `low` pollution.

17. Now, we're planning to compare low and moderate pollution, so we still need a data frame containing only rows with `moderate` pollution. Do the same procedure as above, but for the rows from the moderately-polluted site. Remember to assign the result to a new variable name; don't overwrite any of the existing variables!

18. We need to do one final step before analysis. The functions we'll be using only take vectors (not data frames) as inputs, so select for only the `bees` column of both the low and moderate data frames you created above. Assign the results to `low.vector` and `moderate.vector` respectively. Confirm that each variable contains only a vector before moving on.

    - *Important!* If the variable is showing up under the "Data" section of the Environment in the top right, it's not a vector. It should be showing up under "Values". If you're not getting a vector, try using a different method of selecting the column. (In particular, the `all.data.df[COLUMN]` method will NOT give you a vector)

Section 4: P-values

19. Now we'll be using a t-test to obtain a P-value. We'll go into t-tests more in-depth in a later lecture, but for now what you need to know is that they test for significant differences in the means of two groups. (We're testing for either positive or negative differences; it doesn't matter which group mean is greater if it's significantly different enough.)

20. Paste the following code into your script and run it to generate a P-value and the difference in means between the two sites' samples. (Note: If you get NA for an output, your variables may not have been vectors)

    - **Q6. Copy and paste the following sentence into your answers, then use the two values from your output (the mean and P-value) to fill in the two blanks (not necessarily in the same order): "If the null hypothesis is actually true, then there is a ________ chance that a repetition of this experiment would yield a result of ________ or something more extreme."**
    - **Q7. Based on an alpha (significance threshold) of 0.05 and the resulting P-value, should you reject the null hypothesis? Why or why not?**

```
# run t-test for a p-value and save results to variables
t.result <- t.test(low.vector, moderate.vector)
p.value <- t.result[["p.value"]]
mean.difference <- mean(low.vector) - mean(moderate.vector)

# output results
print(paste("P-value:", p.value))
print(paste("Difference (low - moderate):", mean.difference))
```

21. To wrap up, assume an alpha (significance threshold) of 0.05 and consider the following scenarios:
    - **Q8. Imagine that bees from both the lowly- and moderately-polluted sites behave in the same way. If you got a P-value of 0.1, would it lead to an error? If so, would it be a type I or type II error? Explain your reasoning.**
    - **Q9. Imagine that bees from the lowly-polluted site are less inclined to pollinate *T. stans* than those from the moderately-polluted site. If you got a P-value of 0.1, would it lead to an error? If so, would it be a type I or type II error? Explain your reasoning.**
22. And you're done! Make sure you've answered Q1-Q9 outside of the chunks in your R Markdown file, then knit your file to PDF or HTML and submit it. Make sure you're submitting a knitted PDF or HTML report or you'll lose points.

Reference

Thimmegowda, G.G., Mullen, S., Sottilare, K. et. al. **A field-based quantitative analysis of sublethal effects of air pollution on pollinators.** Proceedings of the National Academy of Sciences, Aug 2020, 117 (34) 20653-20661; DOI: 10.1073/pnas.2009074117