Lab 4: Z-Test, T-Test, and Chi-Square

In this lab, you'll analyze data using a chi-square test, z-test, and t-test. This is the first lab with real analysis, so you'll go through the process of hypothesis building, data management, figure generation, statistical analysis, and finally interpretation of results. Here are the new functions for today:

```
shapiro.test()
z.test()
t.test()
table()
sum()
chisq.test()
```

There are 15 question checkpoints in **bold**. Answer them directly in your R Markdown file, outside of the chunks.

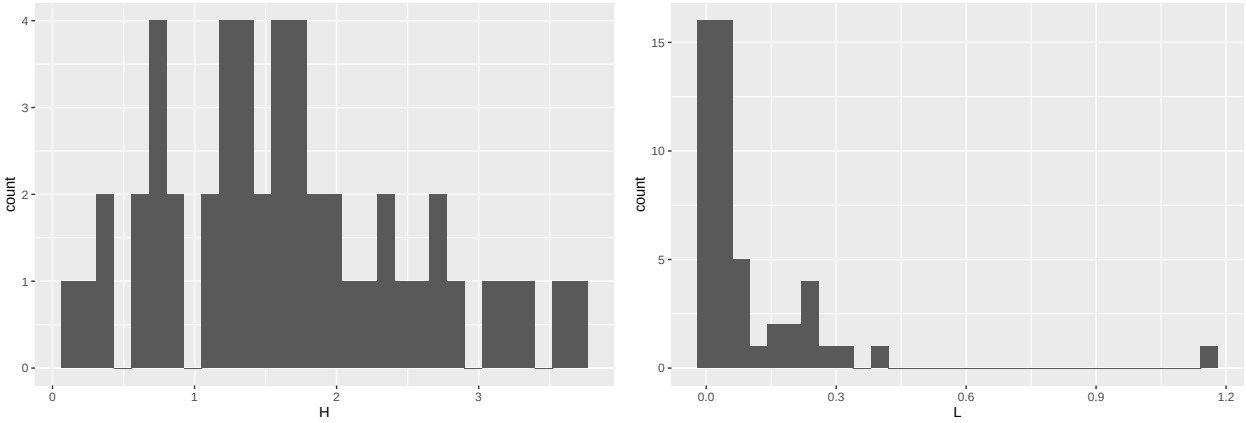
Section 1: Set-up

1. Open a new R Markdown file.
2. Today we'll be using two packages: ggplot2, which we're familiar with, and BSDA, which stands for Basic Statistics and Data Analysis. BSDA contains the z-test function we'll be using, so we'll need to install that first. Copy and paste this into your console and run it: `install.packages("BSDA")`
3. Now, load in both packages for later use so you don't forget.

```
library("BSDA")
library("ggplot2")
```

Section 2: z-test

4. Download the dataset `fly_hindleg_RSPM.csv` from the course website and read it into R using `read.csv()`, assigning it to a variable. These data come from the hindleg RSPM deposition data for fruit flies, approximated from figure 4e.
5. Let's conduct a z-test to compare the data from one of the sites to a mean of zero. There are two sites you can choose from, but we want data that is normally distributed in order to conduct a z-test. Sometimes you can get away with data that is not very normal, but it really depends on each case and in this case let's just go with the safer option!
    - To determine which site has more normal data, create a histogram for each site. You might want to review lab 2 to see how to do this, but remember that the base function is `ggplot()`. The x column in `aes()` will depend on which variable you're graphing, so both histograms should have a different x column.



6. You should have an idea of which data are more normally distributed, but neither are very clearly normal. There are different tests you can run to get a quantifiable measure of normality, so use one of those called the Shapiro-Wilk test to confirm your conclusion from the histograms. The function is `shapiro.test()` and the input should be the column you want to check for from the appropriate data frame. You'll need to use one of the column-selection methods from lab 1 to select the column. Run a separate test for each of the two columns. For this test, the null hypothesis is that the data are normal, so a high p-value suggests normal distribution. **+Q1. According to both the histograms and Shapiro-Wilk tests, which of the two sites have more normal data? +Q2. You should run a z-test on the site with more normal data. What is the alternative and complementary null hypothesis (please label each as null or complementary)? For our one-sample z-test, we're going to compare the mean of our sample of interest to zero.**

```
##
##  Shapiro-Wilk normality test
##
## data:  high
## W = 0.96689, p-value = 0.1725
```

```
##
##  Shapiro-Wilk normality test
##
## data:  low
## W = 0.53379, p-value = 2.277e-11
```

7. Now I'll take this opportunity now to teach you how to better understand the documentation behind functions. Either type `?z.test` into your console or search `z.test` in the help tab of the bottom right panel and you should be able to find the documentation for the function `z.test()`. Documentation is a good place to start when using a new function, but authors of functions vary in how well they write their documentation so it's not always as helpful as we'd like. Here's a rundown of common sections you'll find useful:

    - Description: a description of what the function does
    - Usage: the basic format, or *syntax*, telling you how to use the function. Within the parentheses are *arguments*, each separated by commas. Anything in the Usage section with an equals sign (`=`) has a default value, also specified in the Usage section, so you don't have to use that argument when you're using the function and it'll just use the default. If there's no equals sign, you do have to specifically use that argument. So for `z.test()`, the only argument required by R is `x`, which is the input.

  - (Unfortunately this function has some more requirements that aren't listed in the documentation, so you may find that it asks you to fill in other things too; but this is a different error message than if you didn't use an argument that doesn't have a default value)
- Arguments: a list of the arguments you can use for the function, and their descriptions. When you declare arguments in the parentheses of a function, they can go in any order as long as you specify what it is with the argument's ID (left side of the Arguments table) and an equals sign (=), as is shown in the Usage section. If you don't specify what the argument is in this manner, then they have to go in the order in which they appear in the Usage section.
- Details: further information on how the function works.
- Value: the output of the function
- Examples: examples showing ways that the function can be used

8. With that mini-lesson out of the way, let's finally conduct our z-test. Assume an alpha of 0.05. For this function, let's be extra careful and move through the different arguments slowly. This is from the documentation: `z.test(x, y = NULL, alternative = "two.sided", mu = 0, sigma.x = NULL, sigma.y = NULL, conf.level = 0.95)`
   - x: This is the sample you want to compare to zero. It should be the vector, which you can obtain by selecting the appropriate column using one of the methods that yields a vector.
   - y: The default here is NULL, and we'll leave it as-is because we're only doing a one-sample test. This argument is only if you want to use a second sample.
   - alternative: This is going to depend on what your alternative hypothesis was earlier (you'll lose points if it doesn't match). If you said something about means not equaling each other, you want `two.sided`. If you said that your sample mean would be higher than the overall population, you want `greater`. If you said that your sample mean would be lower than the overall population, you want `less`. The default here is `two.sided` so you can leave it out completely if that's the one you want.
   - mu: This is the mean that the z-test will compare the sample against. In this case, it should be 0, which you can achieve by either specifying `mu=0` or leaving it out completely since 0 is the default (as seen in the documentation).
   - sigma.x: This is the population standard deviation that we're comparing our sample to. Since the sample size is >30, assume that the sample standard deviation is a close-enough estimate. Set this as the standard deviation of the column you chose above. Remember that the function for standard deviation, as you learned in the first lab, is `sd()`.
   - sigma.y: The default here is NULL, and we'll leave it as-is because we're only doing a one-sample test. This argument is only if you want to use a second sample.
   - conf.level: This is the confidence level, as a number between 0 and 1, which will dictate the confidence interval. We'll leave it as-is.
9. Run your z-test and take a look at the output. `z` is the overall z-statistic using the standard error z-statistic formula, `p-value` is the p-value as usual, `alternative hypothesis` is what you tested against (you'll want to make sure this is the one you wanted), `mean of x` is the mean of the sample you input. For now, you can ignore everything else.
   - **Q3. Based on the alpha of 0.05 and the p-value from your result, do you reject the null hypothesis? Explain why or why not.**
   - **Q4. There's a fruit fly exposed to the highly-polluted site has a hindleg RSPM deposition of 1%. What percentage of fruit flies exposed to the same site have a lower hindleg RSPM deposition? Determine this using the z-score, which you'll need to calculate yourself for this individual fruit fly.**

```
##
##  One-sample z-Test
##
## data:  appropriate.column.here
## z = 13.385, p-value < 2.2e-16
```

```
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
##  1.428306 1.918361
## sample estimates:
## mean of x
##  1.673333
```

Section 3: t-test

10. Download and import the dataset `fly_interbeat.csv` to a variable, which represent the standard deviations of interbeat intervals of fruit flies, approximated from figure 4f.

11. Since the t-test is parametric like the z-test, we should check for normality. Run the Shapiro-Wilk test on each of the three columns of the interbeat data.

```
##
##  Shapiro-Wilk normality test
##
## data:  unexposed
## W = 0.96597, p-value = 0.6686
```

```
##
##  Shapiro-Wilk normality test
##
## data:  low
## W = 0.90526, p-value = 0.05181
```

```
##
##  Shapiro-Wilk normality test
##
## data:  high
## W = 0.92156, p-value = 0.1062
```

12. Choose *only two* sites to test against each other. Assume an alpha of 0.05.

    - **Q5. What are the complementary null and alternative hypotheses for your planned test? Please label which hypothesis is which, and include the two sites that you're testing as part of the hypotheses.**

13. Check the documentation for the function `t.test()` to understand how to use it. It's pretty similar to `z.test()`. Since this is a two-sample test, you do need to specify both x and y. x and y should each be one of the columns you've chosen to test; remember to select them using one of the vector-yielding methods from lab 1. The only other argument you need to specify is `var.equal`. Set this to `var.equal = TRUE` so that R uses the Student's T-test. The output below may be different from yours depending on which columns you chose.

```
##
##  Two Sample t-test
##
## data:  first.column.here and second.column.here
## t = -1.7669, df = 38, p-value = 0.08527
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
```

```
##  -67.44625   4.58040
## sample estimates:
## mean of x mean of y
##  109.9695  141.4024
```

14. After running your test, take a look at the output. `t` is the t-statistic, `df` is the degrees of freedom, `p-value` is the p-value, and so on.

    - **Q6. You'll notice that it says "Two Sample t-test". Why was this test two-sample and not one-sample or paired?**
    - **Q7. Based on the p-value and our alpha of 0.05, would you reject the null hypothesis?**
    - **Q8. Since this is data from the paper we've been reading, you can refer back to the paper to see if their conclusions matched yours. Which part of the paper would tell you whether there was a significant difference between the SD interbeat intervals of unexposed, low-exposure, or high-exposure fruit flies?**

Section 4: Chi-square

15. For this last test, the chi-square test, you'll need two files. Download both `antennae_DEGs.csv` and `expected_frequencies.csv`. The former details the differentially expressed genes founds in the antennae of *A. dorsata*, taken from the supplemental material. You'll notice that each gene has an associated function, which is what we'll investigate today. The latter file includes simplified estimates of the number of genes with matching functions in *A. dorsata*'s genome (found on the Hymenoptera Genome Database), on which to model the expected frequencies. If there's no bias toward a certain function's differential expression, then the observed frequencies should match these expected frequencies.

    - Import `antennae_DEGs.csv` to a variable as usual.
    - The second file, `expected_frequencies.csv`, needs to be imported in such a way that the first column actually act as row names rather than its own column. Use the code below to do this, noting the argument, `row.names = 1`.

```
expected <- read.csv("expected_frequencies.csv", row.names = 1)
```

16. Examine both datasets first. The data frame of expected values gives frequencies, which is good, but you'll notice that the actual observed data doesn't have any frequencies even though we need those for chi-squared tests. To remedy this, use a function called `table()`. Run this with the `Functions` column of the observed data as the input, and assign the result to a variable called `observed.frequencies`. This will make a table of how many times each unique entry in the input column appears, which we can use as the observed frequencies in our chi-squared test.

17. You'll notice that the expected values include a category called "Other" which isn't included in your new observed frequency table. This is because these are functions that appear in the whole genome, but didn't appear at all in the list of differentially expressed genes. The chi-square function will want matching entries for both the observed and expected frequencies, so you need to add an "Other" entry to the observed frequencies. Copy and paste the following into the chunk to set this new entry to 0 since it doesn't appear at all in the observed data.

```
observed.frequencies["Other"] <- 0
```

18. Before moving on to chi-square, let's first make a preliminary figure: a bar graph might give us some insight into what our data looks like. If you take a look at `observed.frequencies` and the expected

frequencies, you'll see that the expected frequencies are much higher since they're based on the whole genome even though the observed frequencies are only a subset. Since we can't compare these 1 to 1, you'll need to normalized both of these to appear as proportions instead. Then you'll be able to compare them in a bar graph.

- You can achieved this by dividing each frequency value by the total of all the frequencies for that category. So, each observed frequency would be divided by the total observed frequencies. Each expected frequency would be divided by the total expected frequencies.
- Use the following code to normalize the observed frequencies. `observed.frequencies` is a vector of length 7, while `sum(observed.frequencies)` is only of length 1, so the sum will repeat over and over until it has divided each entry in `sum(observed.frequencies)`.
- Write your own code to do the same for the expected frequencies, but remember that you'll have to select the correct column for both the numerator and the denominator. Assign this to the variable name `expected.frequencies`. If this is done correctly, all of the expected proportions will add up to 1.

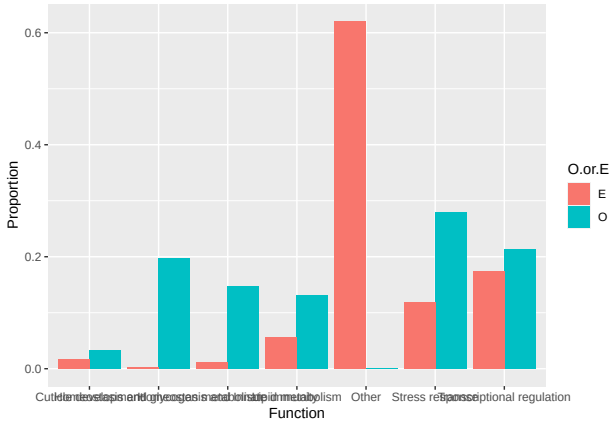
```
observed.normalized <- observed.frequencies/sum(observed.frequencies)
```

19. Remember, ggplot needs a data frame in order to work. The code below, using the `data.frame()` function, will combine everything in a way that ggplot understands. Simply copy and paste it into your code.

```
chi.bar.data <- data.frame( "Function"=rep(rownames(expected), 2),
                            "O.or.E"=c(rep("O",7),rep("E",7)),
                            "Proportion"=c(observed.normalized,expected.normalized))
```

20. Finally, put together the graph! The data frame should be `chi.bar.data`, the x-axis should be `Function`, and the y-axis should be `Proportion`. Within `aes()`, add `fill=O.or.E` to give the observed and expected bars differentiating colors.

- In `geom_col()`, add `position="dodge"` to separate the bars for each function.
- **Q9. What is one thing in this figure that you think is interesting?**



21. Take a look at the figure and choose either "Stress response" or "Transcriptional regulation" to test in the chi-square test. The other functions have lower frequencies so they won't work as well. Assign the name of your chosen function to the variable name, `specific.function`. Remember to put the function name in quotes and type it exactly as it's named in the data. (Example: `specific.function <- "Your chosen function"`)

- **Q10. You'll be testing the observed vs expected frequencies of your chosen function. What are the complementary null and alternative hypotheses for this test? Please label each hypothesis and include which function you chose.**
- **Q11. Is this a goodness-of-fit test or a test of independence?**

22. Since you're only testing one function, the frequencies you want will be the frequency of the function itself and the frequency of every other possible function. Paste and run the following code into your project; note that it depends on the last step so it won't work correctly if you didn't assign the variable.

    - **Q12. Take a close look at this code and the objects it returns. What exactly is this code doing?**

```
specific.observed <- c( observed.frequencies[specific.function],
                   sum(observed.frequencies) - observed.frequencies[specific.function] )
specific.expected <- c( expected[specific.function,],
                   sum(expected) - expected[specific.function,] )
```

23. Now we can get to the actual chi-square test. First, assume an alpha of 0.05. Use the code below to run the test, but keep in mind that the output may be different depending on your chosen function. Notice that we include `correct=FALSE` because this function includes something called a Yates correction by default. We haven't learned about that and it can be finicky sometimes anyway, so we'll just use the basic chi-square without correction.

    - **Q13. Look up the documentation for `chisq.test()`. What do you think that `rescale.p = TRUE` does?**

```
chisq.test( specific.observed, p=specific.expected, correct=FALSE, rescale.p = TRUE)
```

```
##
##  Chi-squared test for given probabilities
##
## data:  specific.observed
## X-squared = 14.891, df = 1, p-value = 0.0001139
```

24. You've completed your first chi-square test in R! Now it's time to interpret it. `X-squared` is the chi-square value, `df` is the degrees of freedom, and `p-value` is our old friend, the p-value.

    - **Q14. How would you have calculated the degrees of freedom?**
    - **Q15. Based on the alpha of 0.05 and the p-value from your result, do you reject the null hypothesis? Explain why or why not.**

25. Feel free to play around some more with the data. When you're done, make sure you've answered Q1-Q15 outside of the chunks in your R Markdown file, then knit your file to PDF or HTML and submit it.

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

Walsh, A.T., Triant, D.A., Le Tourneau, J.J. et. al. **Hymenoptera Genome Database: new genomes and annotation datasets for improved go enrichment and orthologue analyses.** Nucleic Acids Research, Jan 2022, 50 (D1O) D1032–D1039; DOI: 10.1093/nar/gkab1018

Fouks, B., Brand, P., Nguyen, H.N., et. al. **The genomic basis of evolutionary differentiation among honey bees.** Genome Res., May 2021, 4:gr.272310.120. DOI: 10.1101/gr.272310.120. Epub ahead of print. PMID: 33947700.