**Module 7/8 Lab Activity: Statistical inference in SLR & Multivariate correlations**

**PSY 652 Research Methods**

**Oct 23, 2019**

**Activity #1: Inferences in SLR**

This activity is designed to help you understand sampling from a population and inferences we can make towards the population. We simulate a population and pull random samples. No datasets are necessary for this activity.

1. Create a new R notebook from your project file and name it “Inference\_Exploration”
2. Create a new R chunk with a first level header: “Load Libraries”
   1. load the tidyverse, psych & broom packages in this R chunk
3. Create a first level header: “Consider lbslost in the hypothetical population”
4. Create a second level header: “Simulate the population”
   1. Create an R chunk. Copy and paste the following code (yes, really!) and click run.

**# generate some data**

**set.seed(8642) #<- set a seed for consistent results**

**p\_lbslost <- rnorm(n=50000, m=3.00, sd=2.00) # generate a population of 50,000 men**

**my\_pop <- data.frame(p\_lbslost) #<- set it into a data frame**

**meanpop <- mean(my\_pop$p\_lbslost) # save the average lbslost in the population -- our population mean**

*This code creates a dataset with 50,000 observations with a M = ~3.00 and a SD = ~2.00 and places it into a dataframe called my\_pop. The obtained mean is now in an object called “meanpop”. This is the theoretical population distribution.*

***lbslost*** *= pounds lost after one month on the weight loss program*

1. Write a Second level header: “Describe my Data”
   1. Use your preferred method to describe the my\_pop dataset
2. Create a second level header: “Create a histogram of the population”
   1. Create an R chunk: Create a histogram of the variable *p\_lbslost* from the my\_pop dataframe. Set the binwidth to 0.25. Title your histogram “Pounds lost following a weight loss program” Title your x axis “pounds lost".
3. Create a second level header: “Randomly select 3 samples of size 100 from the population”
   1. Create an R chunk: Pull 3 random samples of 100 (with replacement) from the my\_pop dataframe and save them each in a separate object.

Hint: To pull a random sample, use the following code sample:

*object\_name* <- sample\_n(*dataframe\_name*, *sample\_size,* replace=TRUE)

Do this 3 times, once for each random sample.

1. Create a second level header: “Calculate the mean in each random sample”
   1. Create an R chunk. Calculate the mean of *p\_lbslost* in each random sample you pulled (do this 3 times, once for each random sample).
   2. In the white space, write a sentence describing how the means vary across random subsamples of the original population. How much do these means differ from the original population mean?
2. Create a first level header: “Inference exploration for SLR”
3. Create a second level header: “Simulate a population based on SLR results”
   1. Create an R chunk. Copy and paste the following code (yes, really!) and click run.

**set.seed(83587)**

**caldef <- rnorm (mean = 10.80, sd = 5.15, n = 100000)**

**b0 <- -.12**

**b1 <- .29**

**sigma <- 2.1**

**e <- rnorm(mean = 0, sd = sigma, n = 100000)**

**lbslost <- b0 + b1\*caldef + e**

**my\_regpop <- data.frame(caldef, lbslost)**

*This code creates a dataset with 100,000 foobservations and 2 variables: caldef & lbslost to an object called my\_regpop. We are setting true parameter intercepts and slopes. This is the theoretical population distribution.*

***lbslost*** *= pounds lost after one month on the weight loss program*

***caldef*** *= caloric deficit over the course of the program (expressed in 1000 calories). A caloric deficit is a state in which you are burning more calories than you eat.*

1. Create a second level header: “Visualize correlation and regression line for simulated population”
   1. Using whichever method you prefer, plot the correlation of *lbslost* (y axis) and *caldef* (x axis). Add the best fit linear line.

*Hint:* We did this in module 6.

1. Create a second level header: “Fit a Simple Linear Regression (SLR) Model of the population”
   1. In a new R-chunk, create a linear model in which caldef predicts lbslost. Save this model to an object called “mod1”
   2. Display a summary of the model results
2. Create a second level header: “Randomly select 3 samples of size 100 from the population”
   1. Create an R chunk: Pull 3 random samples of 100 (with replacement) from the my\_regpop dataframe and save them each in a separate object.

Hint: To pull a random sample, use the following code sample:

*object\_name* <- sample\_n(*dataframe\_name*, *sample\_size,* replace=TRUE)

1. Create a second level header: “Fit an SLR to each random sample”
   1. Create an R chunk. create a linear model in which caldef predicts lbslost for each of the random samples (do this 3 times, once for each random sample).
   2. Display a summary of each of the model results.
2. In the white space, write a sentence describing how the models vary across random subsamples of the original population. How different are the intercepts and slopes? Are the R^2 the same? How about the standard errors? Why would these results be the same/different between the samples and the actual population.

**Activity 2: Practice Multiplying Matrices in R**

This activity is designed to give you some hands-on practice with basic skills in matrix algebra. This example doesn’t go into a multivariate correlation example, but the answer key for this week’s lab explains how these steps could be applied to this analysis.

We suggest using the following resources for this activity:

1. Module 8 lecture slides and the R notebook that accompanies the Murphy, *In Press* reading.
2. This “Math is Fun” webpage provides a nice overview of what you’re actually *doing* when you multiply matrices: <https://www.mathsisfun.com/algebra/matrix-multiplying.html>
3. Quick-R: provides a dictionary of commands in the base R package that are used for matrix algebra in R: <https://www.statmethods.net/advstats/matrix.html>

Relevant syntax:

%\*% is the command for multiplying two matrices

matrix() is the function for creating a matrix

t() is the function for transposing matrices

1. Create a new R notebook titled “Matrix Algebra Practice”
2. Write a first level header: “Create matrices”
3. Create an R chunk with a second level header: “Create a 3x3 matrix”
4. Use the matrix() function to create a 3x3 matrix of the following values. Save this to an object called “cor\_matrix”

|  |  |  |  |
| --- | --- | --- | --- |
|  | Y1 | Y2 | Y3 |
| X1 | 1.00 | 0.75 | 0.50 |
| X2 | 0.85 | 1.00 | 0.30 |
| X3 | 0.60 | 0.35 | 1.00 |

Hint: *object\_name* <- matrix(c(*value1, value2, value3, etc*), ncol=*#\_of\_columns*,

byrow=T)

1. Confirm that this worked by either asking for R to show the matrix in R output or by clicking on the object in your global environment.
2. Create a new R chunk with a second level header: “Create a 1x3 matrix”
3. Use the matrix() function to create a 1x3 matrix of the following values. Save this to an object called “weights”

|  |  |  |  |
| --- | --- | --- | --- |
|  | Y1 | Y2 | Y3 |
| X1 | 1.0 | 1.0 | 1.0 |

1. Confirm that this worked by either asking for R to show the matrix in R output or clicking on the object in your global environment.
2. Create a new R chunk with a second level header: “Transpose the 1x3 matrix”
3. Use the t() function to transpose the 3x1 matrix you just created. Assign this to a new object called “weights\_t”

*Hint*: *object\_name* <- t(*matrix\_name*)

1. Confirm that this worked by either asking for R to show the matrix in R output or clicking on the object in your global environment (this matrix should now be 3x1).
2. Create a first level header: “Multiply matrices”
3. Create a new R chunk with a second level header: “Multiply the 3x3 and 1x3 matrices”
4. Multiply the first two matrices you created (“cor\_matrix” and “weights”) and save this to a new object called “wR\_matrix” (*Hint*: object\_name <- matrix1 %\*% matrix2)

*\*\*\* Note: Order matters in matrix multiplication. In this case, list the “weights” matrix before the “cor\_matrix” matrix in your code\*\*\*\**

1. Confirm that this worked by either asking for R to show the matrix in R output or clicking on the object in your global environment (this matrix should now be 1x3 with the following values: 2.45, 2.1, 1.8).
2. Create a new R chunk with the second level header: “Multiply this new matrix with the transposed 3x1 matrix”
3. Multiply the matrix you just created (“wR\_matrix”) by the 1x3 matrix you created in step 5 (“weights\_t”) and save this to a new object called wR\_wprime\_matrix.

*\*\*\* Note: in this case, list the “wR\_matrix” matrix before the “weights\_t” matrix in your code\*\*\*\**

1. Confirm that this worked by either asking for R to show the matrix in R output or clicking on the object in your global environment (this matrix should now be one value: 6.35. This is the simple sum of all of the elements in the matrix).

**Activity #3: Multivariate Correlation Conceptual Practice**

Answer the following questions in the white space at the bottom of your R notebook for activity number 2. Write 1-2 sentences for each. Use the Module 8 lecture slides and Murphy, *In Press* reading for reference.

1. What are the pros and cons of conducting multivariate correlations that include multiple outcome variables? Why would you select this method over models that include univariate outcomes?
2. Why would you assign different weights to predictor and outcome variables in multivariate correlations? How might you adjust your weighting if some predictors are highly correlated?
3. Thinking about your own research interests, when would multivariate correlation with variable weighting be a good fit for your research? Give a brief example of a research study you might use this for.

Turn in your R notebooks for Activity 1 and Activities 2-3 to the Module 7 and 8 lab assignments, respectively, on Canvas.