Week 4 – Battery Data Confidence Interval Exercise

Instructions: This exercise provides an opportunity to work with data and develop a confidence interval. The provided data set represents battery life in mobile phones for two different types of batteries: NiCad and Li-Ion. The reported times are in minutes starting with power-on, leaving the phone with the screen dark, and waiting until the phone powers itself off because of low battery. The research question is whether the Li-Ion batteries (labeled as type 2 in the data set) offer a better battery life than the NiCads under these conditions.

1. Download the provided CSV file (bBatterydata.csv) and read in the data using **read_csv() (from the readr package)** or the Import Data dialog in R-Studio. Examine the data with str() and summary() and provide an overview of what you find in a comment.

2. Use graphical diagnostics to explore the data and report what you see in comments. Among other options, make sure to use a grouped boxplot to examine the distributions of battery life for each type of battery. Note that boxplot() will take formula language as its first argument, like this: Time ~ Battery

3. Calculate the mean and standard deviation of both NiCad and Li-Ion. Add a comment indicating which mean is higher. Comment on the standard deviations for each subsample. Power user trick: You can use tapply() to apply a procedure like mean or sd to groups: tapply(X=battData$Time,INDEX=battData$Battery,FUN=mean)

4. Use the t.test() command to create a confidence interval for the mean difference between these two vectors of data. Report the numeric values of the upper and lower bounds of the confidence interval.

5. In a brief comment, interpret the 95% confidence interval with respect to the research question. Explain what a confidence interval is and what, if anything, you know about the position of the population mean difference between the two types of battery. Hint: Do NOT say that there is a 95% chance that the population mean difference lies within this confidence interval.

6. Imagine you are now reporting these results back to the engineers who are working on these batteries. Engineers generally have no difficulty with quantitative concepts, but they are not statisticians, so they need thoughtful guidance on making sense of the results. Explain in a comment what caveats, cautions, or warnings accompany the confidence interval you just documented.

Next, with the help of some R&D funding, we have obtained a complete population of battery data. Using these data, we can obtain not one sample, not two samples, but 100 (or more) samples of battery data. In this next exercise, write some R-code to draw 100 samples of n=172 batteries (which will be about half NiCad and half Li-Ion). Then calculate a confidence interval for each one using t.test(). Then plot the results and count how many of your confidence intervals actually contain the mean difference.

7. Read in the data from the provided file, "cBattPop.csv." Store the resulting data set in a data frame called "cBattPop". Use summary(cBattPop) to examine the variables.

8. Calculate and report the **true population mean difference** between NiCad batteries and Li-Ion batteries. Save the result for later use in a new variable called popMeanDiff. Try this power user hint to get the information you need:
tapply(X=cBattPop$Time,INDEX=cBattPop$Battery,FUN=mean)

9. In order to use replicate()to draw samples and calculate confidence intervals, we will need to build a small custom function. Here's one that does the job, or write your own:

```
replBattCI <- function()
{
  mySamp <- cBattPop[sample(1:100000,size=172, replace=TRUE), ]
  nicad <- mySamp$Time[mySamp$Battery==1]
  liion <- mySamp$Time[mySamp$Battery==2]
  return(t.test(nicad,liion)$conf.int)
}
```

   Remember that you must run this code to define the function and make it ready to call.

10. Use replicate() to call your function 100 times and store the resulting list of confidence intervals. This line of code will do that:

```
confIntList <- t(replicate(100,replBattCI()))
```

   Note that the t() function is a matrix transposition that flips the rows and columns of the result to simplify plotting. Examine the results. Which of the columns is the lower bound of the confidence interval and which is the upper bound?

11. Plot the lower bounds of the 100 CIs using plot.ts(). That will treat the 100 lower bounds as a time series, which makes a pleasing line. Here's a line of code that does the job:

```
plot.ts(confIntList[,1],col="red", ylim=c(-40,-5))
```

   Note that we have set the limits on the Y-axis to range from -5 down to -40. This makes sure that there is room for the upper bound as well. You may be able to adjust these limits to make the graph clearer.

12. Now add the upper bounds of the 100 Cis using the lines() command. This command simply adds a new line to an existing plot:

```
lines(confIntList[,2], col="green")
```

13. Finally, add a horizontal line that shows the position of the true population mean difference. If you did Question #2 correctly, this should work:

```
abline(h=popMeanDiff)
```

14. Examine the graph to see how many times either the red or the green line crosses the black horizontal line. Report this result in a comment. Are these crossings bad? What if you drew one of those samples in your own work, that is, where the upper or lower bound crossed the black line? Would you have any way of knowing that this had occurred?