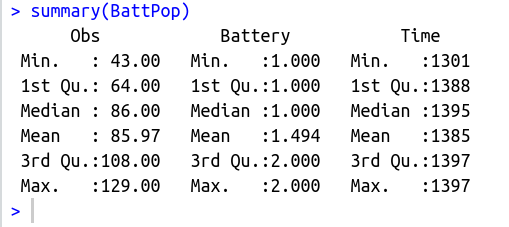
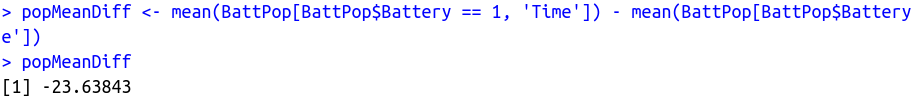
# IST772 Week 4 Breakout – Build and Evaluate 100 Confidence Intervals

Today, with the help of some intensive research lab support, 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 exercise, you will draw 100 samples of n=200 batteries (which will be about half Battery 1-NiCad and half Battery 2-Li-Ion). Then calculate a confidence interval for each one using t.test(). You will plot the results and count how many of your confidence intervals contain the mean difference.

1. Read in the data from the provided file, “BattPop.csv.” Store the resulting data set in a data frame called “BattPop”. **Use summary() to examine the variables.**



1. **Calculate the true population mean difference** between NiCad batteries (labeled 1) and Li-Ion batteries (labeled 2). **Save the result for later use in a new variable called popMeanDiff**.



1. In order to use replicate()to draw samples and calculate confidence intervals, we will need to build a small custom function. **Copy and paste this function into R and add comments to each line of code explaining what it does.**  
     
   replBattCI <- function()

{

mySamp <- BattPop[sample(1:100000,size=200, replace=TRUE), ]

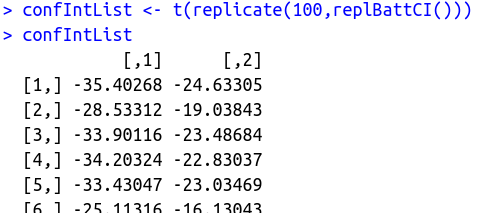
nicad <- mySamp$Time[mySamp$Battery==1]

liion <- mySamp$Time[mySamp$Battery==2]

return(t.test(nicad,liion)$conf.int)

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

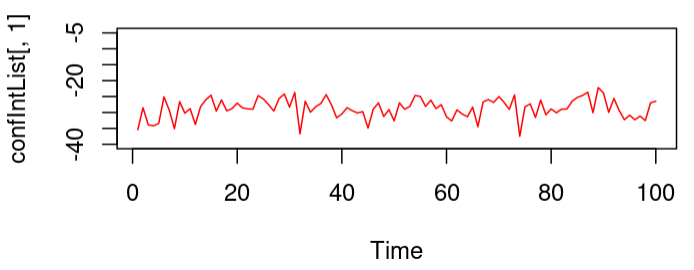
1. Use replicate() to call your function 100 times and store the resulting list of confidence intervals. This line of code will do that:  
     
   confIntList <- t(replicate(100,replBattCI()))  
     
   Note that the t() function is a matrix transposition that flips the rows and columns of the result to simplify plotting. Examine the results. **Which of the columns is the lower bound of the confidence interval and which is the upper bound?**

****

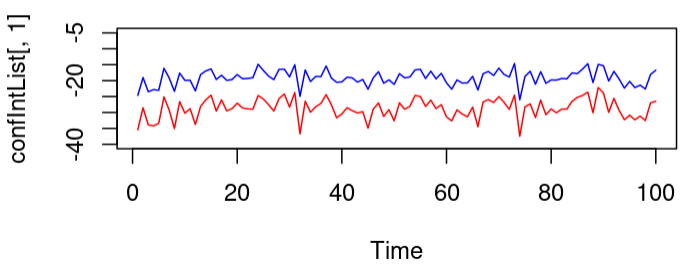
[,2] is the upper bound and [,1] is the lower bound since the values in [,2] are greater than [,1]

1. Plot the graph of the results using the code below. **Include a copy of the graph.**
2. 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))

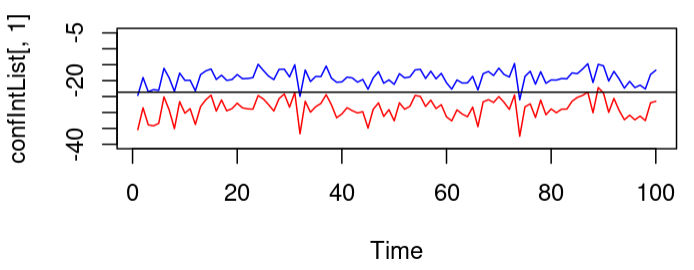
Note that we have set the limits on the Y-axis to range from -40 down to -5. This makes sure that there is room for the upper bound as well. You may be able to adjust these limits to make the graph clearer.



1. 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="blue")



1. Finally, add a horizontal line that shows the position of the true population mean difference. If you did Question #2 correctly, this will work:  
   abline(h=popMeanDiff)



1. Examine the graph to see how many times either the red or the blue line crosses the black horizontal line. **Report this result in a comment and answer the following questions.**
2. What do these crossings represent?

The lines cross the black horizontal line 4 times in my chart. It means that in this 4 times the samples mean fall outside the 95% of confidence interval, so they are likely not a sample from the true population.

1. Are these crossings bad?

The crossings are likely to be bad, because they might not represent the population, or they can also be outliers.

1. Would you have any way of knowing that this had occurred if you drew one of those samples in your own work, that is, where the upper or lower bound crossed the black line?

Yes, but it would be very difficult to check the means of the samples one by one. In huge datasets, it would be impossible, and in real life, there might also not be any population mean represented at all.