Instructions: In the previous exercise, you created a 95% confidence interval around the mean difference of a sample of n=86 NiCad batteries and a sample of n=86 Li-Ion batteries. Usually we only have access to samples and no matter how large or small they may be, they are our only view of what may be happening in the population.

But 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, 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.

1. 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.
2. 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)
3. In order to use replicate()to draw samples and calculate confidence intervals, we will need to build a small custom function. Cut and paste this function into R and add comments to each line of code explaining what it does:  
     
   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.

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. 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.
3. 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")
4. 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)
5. 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?
6. If time permits, modify replBattCI() so that it only samples 60 elements from the population instead of 172. Rerun exercises 4 through 7. Comment on what, if anything, has changed about your graph.
7. Power user move: Write a few lines of code that automatically counts and reports the number of line crossings in confIntList.
8. Share your code and comments