Additional File 1: Cross-Correlation Function Tutorial

Here we provide an example applying the CCF and sliding CCF to a consumed tag dataset. The dataset used in this example specifies depth records as negative values. This example was originally evaluated in R version 2.15.3 [1]. Before trying this example, you'll need to save the following two files (obtained from Additional Files 2 and 3) on your computer:

Additional File 2. Time-series depth and temperature records from X-Tag 65821
Additional File 3. Sliding CCF function R code

You will need to be able to identify the directory in which the documents are saved. Below, all blue text indicates R code. Simply copy and paste the blue text directly into the R console or into an R script.

Step 1: Clear the R workspace.

```
### (1) CLEAR WORKSPACE

      rm(list = ls())
```

Step 2: Load the sliding.ccf function into your R workspace. To complete this step, you must specify the directory on your computer containing the "Sliding_CCF_Function.r" file by changing the highlighted text to reflect the location of the script on your computer.

```
### (2) LOAD SLIDING CCF FUNCTION

      source("C:\\Test\\Sliding_CCF_Function.r")
```

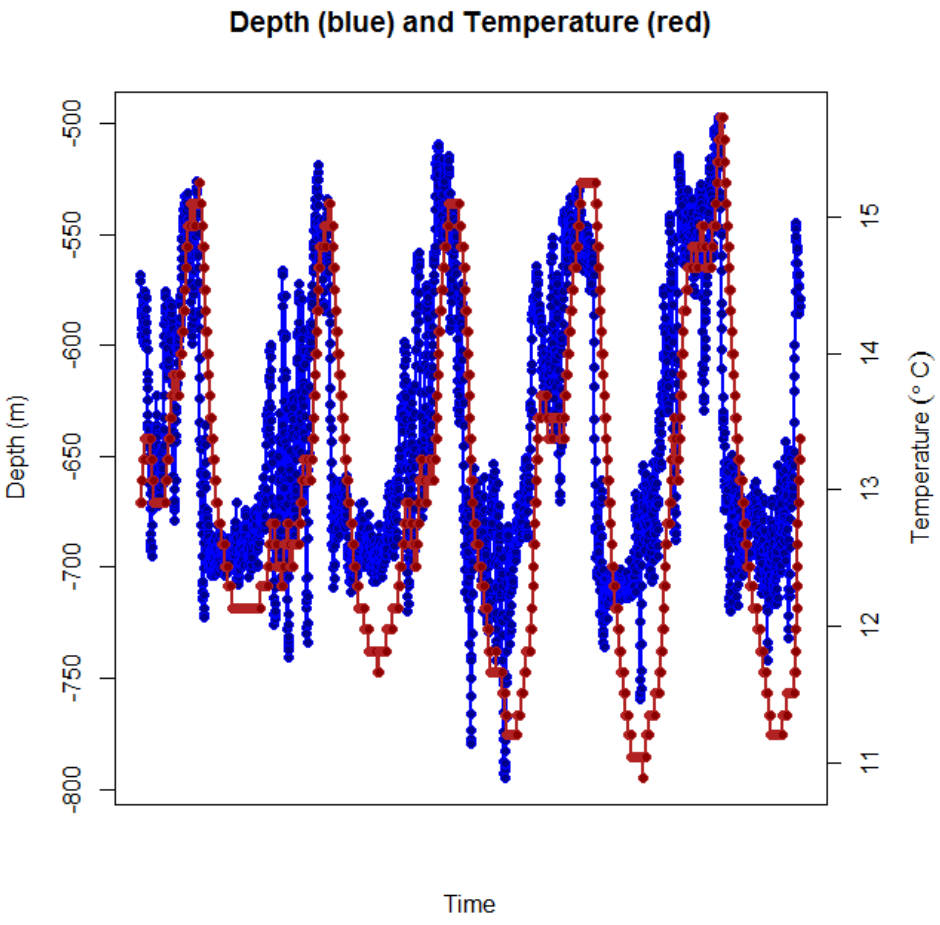
Step 3: Load the depth and temperature data file into your R workspace. First, you must specify the directory on your computer containing the "65821_ExampleData.txt" file by changing the highlighted text to reflect the location of the data on your computer. Then you will be able to read the data directly into R.

```
### (3) LOAD DATA

      ### SPECIFY FILE WITH DIRECTORY FOR ANALYSIS
      dataFile = "C:\\Test\\65821_ExampleData.txt"

      ### IMPORT FILE
      myData <- read.table(dataFile,header=T,sep="\t")
      head(myData) # VIEW THE FIRST FEW ROWS OF DATA
```

Step 4: Visualize data. It is always good practice to view data prior to analysis. In these next lines of code, we overlay the depth and temperature data.

```
### PLOT DATA - OVERLAY DEPTH AND TEMPERATUER PROFILE

      x11();par(mar = c(5,5,5,5))
      plot(myData$Depth,type="l",col = "blue", ylab="Depth
            (m)",xaxt="n",xlab="Time",lwd = 2)
      points(myData$Depth, pch = 21, col = "blue", bg = "darkblue")
      par(new = TRUE)
      plot(myData$Temp, col = "firebrick",type="l",ylab="",
            yaxt="n",xaxt="n",xlab="",lwd = 2)
      points(myData$Temp, pch = 21, col = "firebrick", bg = "darkred")
      axis(side = 4)
      mtext(side = 4, line = 3, expression(Temperature~(degree ~C)))
      title("Depth (blue) and Temperature (red)")
```

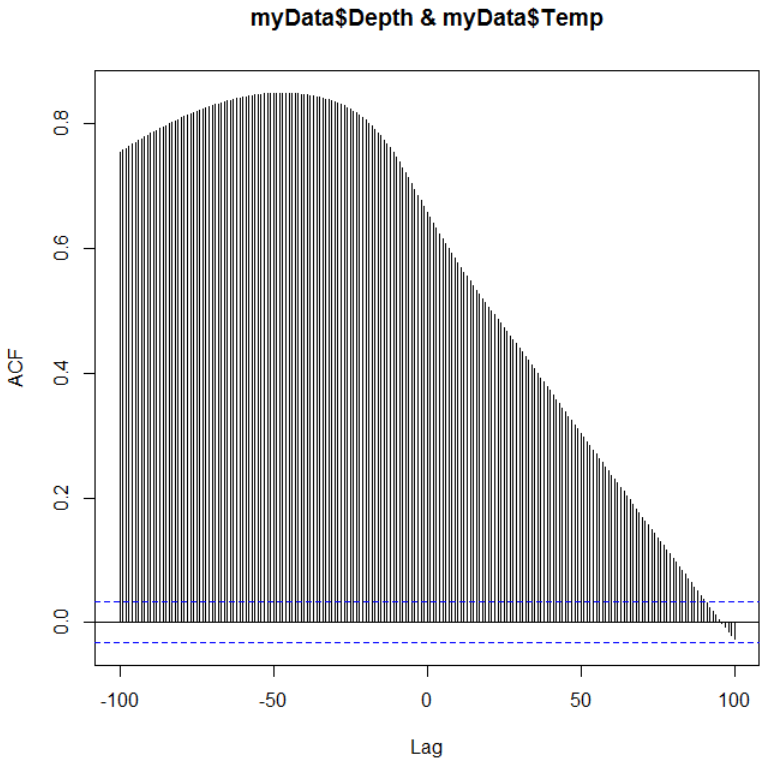
This plot of overlaid depth and temperature data allows you to visually assess the data for any potential lags. Here, we notice a potential lag, such that the depth (blue) is temporally leading the temperature (red) profile. In other words, the red profile appears shifted to the right.



Step 5: Apply cross-correlation function. Here we use the "ccf" function available in R.

```
### (5) RUN CCF ON DEPTH AND TEMPERATURE DATA

      myCCF = ccf(x = myData$Depth, y = myData$Temp, lag.max = 100, type =
            "correlation", plot = TRUE)

      ### IDENTIFY THE MAXIMUM CORRELATION AND CORRESPONDING LAG VALUE
      maxIdx = which.max(abs(myCCF$acf)) # INDEX OF MAXIMUM CORRELATION VALUE
      myCCF$acf[maxIdx] # MAXIMUM (ABSOLUTE-VALUE) CORRELATION
      myCCF$lag[maxIdx] # LAG ASSOCIATED WITH MAXIMUM CORRELATION
```

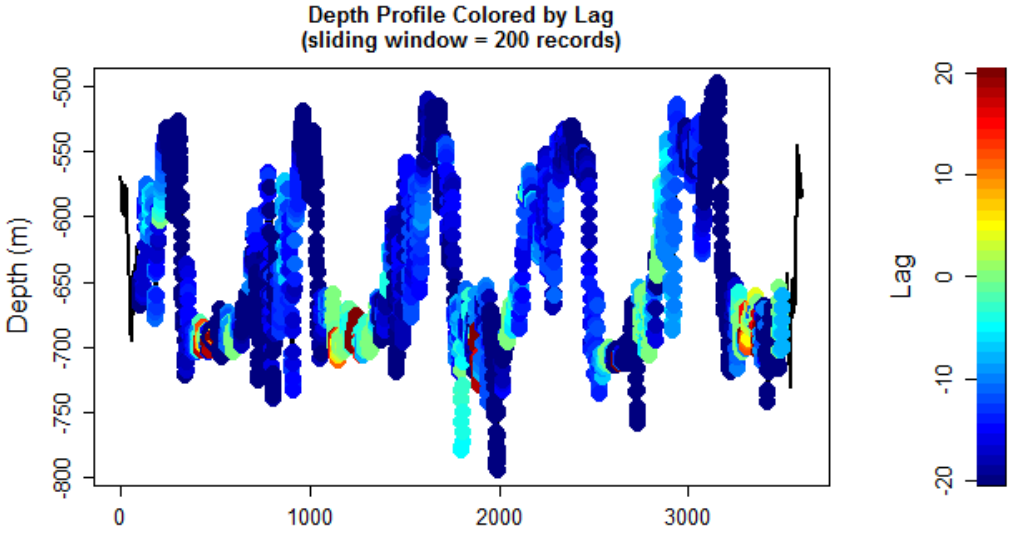
This code should produce a plot (below) and print numerical results to your R console. Specifically, you should see that the maximum absolute-value correlation is 0.849 corresponding to a lag of −48. This result matches the figure output (below). In order to see the documentation for the R "ccf" function, simply type `?ccf` into your R console.

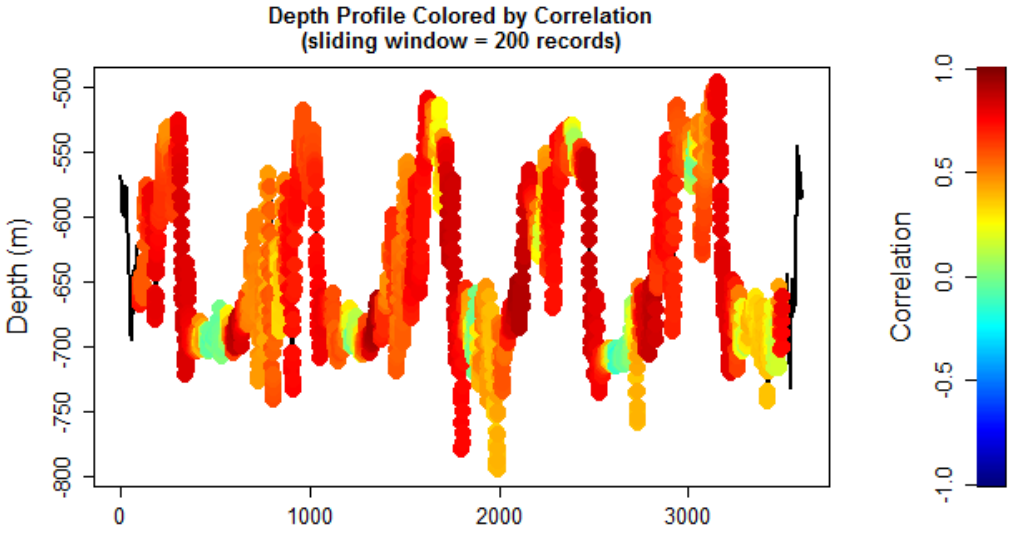


Step 6: Apply the sliding cross-correlation function. Here we run the sliding CCF R function, which you already loaded into R in step 2.

```
### (6) RUN SLIDING CCF ON DEPTH AND TEMPERATURE DATA

      mySlidingCCF = sliding.ccf(ts.depth = myData$Depth, ts.temp =
            myData$Temp, win.width = 200, lag.max = 20, plot.results= TRUE)
```

Notice in the output (below) that the first 100 records and last 100 records do not have an associated correlation and lag (represented by solid black line). Since, the window width has been specified to 200 records (win.width = 200), the function starts on record 100 and stops within 100 records of the end. Therefore, the edges of the time-series do not acquire any CCF lag or correlation values.





Reference

1. R Core Team. R: a language and environment for statistical computing. Version 2.15.3. R Foundation for Statistical Computing, Vienna; 2013.