motoRneuron

Andrew Tweedell

September 26, 2018

# Using motoRneuron for motor unit synchronization

The purpose of this vignette is get you started using the motoRneuron package for analyzing time-domain motor unit synchronization, even if you’ve never used R before! So we will show you how to install the toolbox through CRAN, access the functions, help files, and sample motor unit data. Next, we will provide code to get typical time series data into a workable format for our functions. Then we will go step by step through some analyses with this sample data to demonstrate its applicability.

## Time-domain synchronization

It’s assumed that if you’re reading this, you have at least a grasp of what motor unit synchronization means and what you hope to get out of this package. But breifly, time-domain synchronization, in the neuromuscular context we are presenting it, is the tendency of separate motor units (i.e. motor neurons and their associated muscle fibers) to discharge near-simultaneously (within 1 - 5 ms of each other) more often than would be expected by chance. It is often interpreted as an indicator of functional connectivity between motor neurons through common excitatory post-synaptic potentials. This is quanitified by a cross-correlation analyses, which is essentially the core of this package. A peak in the resulting cross-correlation histogram represents a higher probability of discharges at that latency. Indices that quantify the magnitude of the peak are the primary outputs for the package

### Functions

1. recurrence\_intervals() correlates the discharge times of one motor unit against those of another with motor unit.
2. bin() creates cross-correlation histogram from the recurrence intervals.
3. plot\_bins() plots the cross-correlation histogram with ggplot2.
4. mu\_synch() is an all-in-one function that automatically performs the cross-correlation process from 1) and 2) then assesses the histogram for peaks using 3 optional methods (listed in 4)). 6 commonly used synchronization indices are returned:
   * Common Input Strength (CIS)
   * k’
   * k’-1
   * E
   * S
   * Synch Index (SI)
5. 3 methods of peak determination available in mu\_synch() and as individual functions.
   1. Visual method - visual\_mu\_synch()
   2. Cumulative Sum (Cumsum) method - cumsum\_mu\_synch()
   3. Z-score method - zscore\_mu\_synch()

## Package Implementation

Install the package off the Comprehensive R Archive Network and attach it to your workspace with the following functions.

install.packages("motoRneuron")

library(motoRneuron)

View the package or any of the function help files by using ‘?’.

?motoRneuron  
?mu\_synch

## Data: motor\_unit\_data

Let’s load the sample motor unit data frame from the package into the working environment and look at it before we play around with it. The data is 30 seconds of 2 concurrently active motor units discharging at semi-regular intervals. It was sampled at 1000 Hz.

motor\_unit\_data <- motoRneuron::motor\_unit\_data  
head(motor\_unit\_data)

## Time motor\_unit\_1 motor\_unit\_2  
## 1 0.000 0 0  
## 2 0.001 0 0  
## 3 0.002 0 0  
## 4 0.003 0 0  
## 5 0.004 0 0  
## 6 0.005 0 0

You’ll see our first column is Time in seconds. Makes sense. The other columns indicate what motor units are active with 1’s indicating a discharge at that time point. But the synchronization functions say they want the actual discharge times as inputs… so how do I get that? We’ll help you out. We included this step because this is a commonly used format for time series data like motor unit discharges. So to get the data in a usable form for our functions, we will have to subset out only the time points the motor unit has a 1.

motor\_unit\_1 <- data.frame(Time = motor\_unit\_data[["Time"]], MotorUnit1 = motor\_unit\_data[[2]]) %>%   
 subset(. , MotorUnit1 ==1)   
motor\_unit\_1 <- as.vector(motor\_unit\_1$Time)  
motor\_unit\_2 <- data.frame(Time = motor\_unit\_data[["Time"]], MotorUnit2 = motor\_unit\_data[[3]]) %>%  
 subset(. , MotorUnit2 ==1)  
motor\_unit\_2 <- as.vector(motor\_unit\_2$Time)

You’ll see the individual motor unit data is now in vector format containing only the time points of discharge.

head(motor\_unit\_1)

## [1] 0.035 0.115 0.183 0.250 0.306 0.377

head(motor\_unit\_2)

## [1] 0.100 0.205 0.298 0.377 0.471 0.577

So we’ve got our data in a format that are motoRneuron functions will take.

## Calculating recurrence\_intervals()

recurrence\_intervals() calculates the recurrence intervals based on how many *recurrence interval orders* is specified (see image below). Specifically they are calculated as the timing difference between a discharge event in one (reference) motor unit and the nearest forward and nearest backward discharge events in another (event) motor unit. The motor unit used as the reference unit is the unit with *fewer* discharges. Order refers to how many discharges before and after we are calculating. Default settings are for first order recurrence intervals only as previous research has shown false peaks are likely with higher orders.

knitr::include\_graphics("C:/Users/andrew.tweedell.ctr/Desktop/Synch Working Folder/R Package Creation/motoRneuron/vignettes/Recurrence\_Intervals\_Diagram.png")

Caption The function takes the vector arguments motor\_unit\_1 and motor\_unit\_2, along with a numeric indicating the order. The code below calculates the first and second forward and backward discharges.

recur <- recurrence\_intervals(motor\_unit\_1, motor\_unit\_2, order = 2)  
names(recur)

## [1] "Reference\_Unit" "Number\_of\_Reference\_Discharges"  
## [3] "Reference\_ISI" "Mean\_Reference\_ISI"   
## [5] "Event\_Unit" "Number\_of\_Event\_Discharges"   
## [7] "Event\_ISI" "Mean\_Event\_ISI"   
## [9] "Duration" "1"   
## [11] "2"

As you can see, various descriptors of the motor units are returned as a part of a list . At the end are vectors containing all the recurrence intervals. For recur, this is “1” and “2”. For example, the mean inter-spike interval (ISI) for the reference motor unit and the number of times the event motor unit discharged are returned.

recur$'Reference\_Unit'

## [1] "motor\_unit\_2"

recur$Mean\_Reference\_ISI

## [1] 0.098

recur$Number\_of\_Event\_Discharges

## [1] 443

head(recur$'1')

## [1] -0.065 0.015 -0.022 0.045 -0.048 0.008

For demonstration purposes we will only be using the first order intervals.

## Discretize recurrence intervals with bin()

Only 2 arguments are needed for bin(); a vector containing the recurrence intervals and a numeric indicating the bin size or width. Using unlist() will take our intervals from before out a list format and into a vector format.

recur <- unlist(recur$`1`)  
binned\_data <- bin(recur, binwidth = 0.001)

A simple data frame is returned. Bin tells us how far before or after the discharge of our reference motor unit that the other motor unit discharged. Bin “0.000” represents the two motor units firing at the exact same time. Freq is the frequency of that interval happening in the trial. Below we parse out from the data frame only the bins representing +/- 0.015 s around the reference motor unit discharge.

subset(binned\_data, Bin >= -0.015 & Bin <= 0.015)

## Bin Freq  
## 87 -0.015 4  
## 88 -0.014 7  
## 89 -0.013 12  
## 90 -0.012 5  
## 91 -0.011 3  
## 92 -0.010 4  
## 93 -0.009 5  
## 94 -0.008 12  
## 95 -0.007 14  
## 96 -0.006 5  
## 97 -0.005 10  
## 98 -0.004 10  
## 99 -0.003 2  
## 100 -0.002 13  
## 101 -0.001 3  
## 102 0.000 21  
## 103 0.001 13  
## 104 0.002 1  
## 105 0.003 5  
## 106 0.004 4  
## 107 0.005 6  
## 108 0.006 4  
## 109 0.007 5  
## 110 0.008 7  
## 111 0.009 4  
## 112 0.010 4  
## 113 0.011 0  
## 114 0.012 9  
## 115 0.013 2  
## 116 0.014 4  
## 117 0.015 4

## Visualization of histogram with plot\_bins()

plot\_bins() leverages ggplot2 to graph the binned recurrence intervals as a histogram. Mean bin size or average frequency of the bins is indicated as a red line on the graph.

plot\_bins(binned\_data)

 From the graph, we can see that there are higher frequencies the nearer to “0.000” we get. This may indicate that there is a more than likely chance that these two motor units are functionally connected somehow. If there is a peak in this histogram, then our synchronization functions will determine where they are and calculate the various indices from it.

## Synchronization functions

mu\_synch() is the all-in-one function that calls both recurrence\_intervals() and bin() to perform the initial analysis. It then uses the user-chosen methods for determining if and where there is a peak. More specifically, it uses these methods to find the upper and lower boundaries of a peak if there is one. “Visual”, “Cumsum”, and “Zscore” methods are described below and are available for peak determination. In addition, each method can be called individually using their own functions.

synch\_data <- mu\_synch(motor\_unit\_1, motor\_unit\_2, method = c("Visual", "Cumsum", "Zscore"), order = 1, binwidth = 0.001, plot = F)

* Motor unit vectors, “order”, and “binwidth” are the same arguments used for recurrence\_intervals() and bin().
* “method” is a vector of character strings representing which methods you want to call for peak boundary determination. Default is just “Visual”. mu\_synch() calls each specified method by their individual functions (described later).
* “plot” is a TRUE/FALSE logical on whether to output the histogram from plot\_bins(). Default is FALSE

A list is returned including the motor unit characteristic data and the 6 synchronization indices (described below) for each of the methods chosen.

names(synch\_data)

## [1] "Data" "Visual Indices" "Zscore Indices" "Cumsum Indices"

synch\_data$`Visual Indices`

## $`CIS`  
## [1] 2.163168  
##   
## $kprime  
## [1] 3.79096  
##   
## $kminus1  
## [1] 2.79096  
##   
## $E  
## [1] 0.2110322  
##   
## $S  
## [1] 0.08638251  
##   
## $SI  
## [1] 0.2117218  
##   
## $Peak.duration  
## [1] 0.01  
##   
## $Peak.center  
## [1] 0

synch\_data$`Zscore Indices`

## $CIS  
## [1] 1.714933  
##   
## $kprime  
## [1] 4.284502  
##   
## $kminus1  
## [1] 3.284502  
##   
## $E  
## [1] 0.1673037  
##   
## $S  
## [1] 0.06848299  
##   
## $SI  
## [1] 0.1678505  
##   
## $Peak.duration  
## [1] NA  
##   
## $Peak.center  
## [1] NA

synch\_data$`Cumsum Indices`

## $CIS  
## [1] 2.163168  
##   
## $kprime  
## [1] 3.79096  
##   
## $kminus1  
## [1] 2.79096  
##   
## $E  
## [1] 0.2110322  
##   
## $S  
## [1] 0.08638251  
##   
## $SI  
## [1] 0.2117218  
##   
## $Peak.duration  
## [1] 0.01  
##   
## $Peak.center  
## [1] 0

### Synchronization Index Formulas

1. Common Input Strength (CIS) = extra counts in peak / duration of time both motor units are active (s)
2. k’ = total counts in peak / expected counts in peak
3. k’-1 = extra counts in peak / expected counts in peak
4. E = extra counts in peak / number of reference discharges
5. S = extra counts in peak / (number of reference discharges + number of event discharges))
6. Synch Index (SI) = (extra counts in peak / (total counts in peak / 2))

Peak location is also returned with the Synchronization Indices:

* Peak center - Describes the location of the middle of the peak and is calculated as the median between the upper and lower boundaries.
* Peak Duration - Describes the broadness of the peak and is calculated as the range for the upper and lower boundaries.

### Visual Determination

Visual determination of peaks in a cross-correlation histogram takes a more subjective approach than other automated detection algorithms. Investigators are usually asked where they see peaks, but as bin count or bin frequency can vary greatly, discerning areas of greater frequncy can be difficult for the human eye. To make it easier, the mean baseline bin frequency (considered the mean frequncy of bins >= -0.06 s and <= 0.06 s) is subtracted from each bin. Then the cumulative sum is taken from these normlized bins. This method allows for normally difficult to see increases in frequency (or peaks) to show up as deflections in the graph. The boundaries of these deflections are the boundaries of the peak. When called, this function automatically configures the normalized cumulative sum then leverages the dygraphs package to interactively display it for the user to view and determine peak boundaries.

knitr::include\_graphics('./Dygraph Cumulative Sum Graph.png')  
![Recurrence Intervals Diagram](Dygraph Cumulative Sum Graph.png)

In the graph above you see the x-axis is just like the bins of the cross-correlation histogram, representing the time difference between firings of the motor units. The y-axis notes the normalized cumulative sum values for each bin. The mouse can be used to drag a vertical cursor along the graph. This will display x and y values.

When the function is called, the user is prompted in the console to indicate whether they actually see a deflection in the graph near time “0.00” that may indicate a peak (y/n).

Is there a discernable change in slope or deflection near time 0 (y/n)?

* If no, default bins of +/- 0.005 s are chosen and fed into the calculations.
* If yes, the user is prompted one at a time to input the left and right boundaries (time points in seconds) of a peak seen as a dramatic slope increase in the cumulative sum graph around 0. These peak boundary bins are then fed into the calculations.

Enter the time (in sec) of the start of slope change (left or lower boundary of peak) of the graph:  
Enter the time (in sec) of the end of slope change (right or upper boundary of peak) of the graph:

The synchronization function for visual determination is below. A list similar to mu\_synch() is returned.

visual\_mu\_synch(motor\_unit\_1, motor\_unit\_2, order = 1, binwidth = 0.001, get\_data = T, plot = F)

### Cumsum Determination

The cumulative sum method is an automated method that doesn’t not require input by the user. Peak boundaries are determined as the bins associated with 10 and 90% of the range (maximum minus minimum) of the cumulative sum. The peak is considered significant if its mean bin count exceeds the sum of the mean and 1.96 \* standard deviation of the baseline bins (<= -0.06 s and >= 0.06 s). If no significant peak is detected, a default +/- 0.005 s peak is used.

cumsum\_mu\_synch(motor\_unit\_1, motor\_unit\_2, order = 1, binwidth = 0.001, get\_data = T, plot = F)

### Z-score Determination

Another automated method for peak determination, the z-score method relies on the assumption that if 2 motor units are independent (i.e. no synchronization) then the cross-correlation between them is random and the histogram between would appear flat. So first, a random uniform distribution is used to produce a flat histogram and calculate a mean + 1.96 \* standard deviation threshold for the experimental cross-correlation histogram. Any bins in the experimental histogram within +/- 0.01 s of 0.000 that crosses the threshold are considered to be significantly greater than expected due to chance and subsequently used for analysis. If no peaks are detected, synchronization indices of 0 are returned. *Because the z-score method tests each bin individually, peak bins are not necessarily adjacent.* Therefore, peak duration and peak center are returned as NA.

zscore\_mu\_synch(motor\_unit\_1, motor\_unit\_2, order = 1, binwidth = 0.001, get\_data = T, plot = F)