A peer-reviewed version of this preprint was published in PeerJ on 10 December 2019.

View the peer-reviewed version (peerj.com/articles/7907), which is the preferred citable publication unless you specifically need to cite this preprint.

Tweedell AJ, Tenan MS. 2019. motoRneuron: an open-source R toolbox for time-domain motor unit analyses. PeerJ 7:e7907 https://doi.org/10.7717/peerj.7907

motoRneuron: an open-source R toolbox for time-domain motor unit analyses

Andrew J Tweedell [Corresp., 1], **Matthew S Tenan** [2]

[1] Human Research and Engineering Directorate, United States Army Research Laboratory, Aberdeen Proving Ground, Maryland, United States of America

[2] Defense Health Agency, Falls Church, Virginia, United States of America

Corresponding Author: Andrew J Tweedell
Email address: andrew.j.tweedell.civ@mail.mil

Motor unit synchronization is the tendency of motor neurons and their associated muscle fibers to discharge near-simultaneously. It has been theorized as a control mechanism for force generation by common excitatory inputs to these motor neurons. Magnitude of synchronization is calculated from peaks in cross-correlation histograms between motor unit discharge trains. However, there are many different methods for detecting these peaks and even more indices for calculating synchronization from them. Methodology is typically laboratory-specific and requires expensive software, like Matlab or LabView. This lack of standardization makes it difficult to draw definitive conclusions about motor unit synchronization. To combat this, we have developed a freely available, open-source toolbox, “motoRneuron”, for the R programming language. This toolbox contains functions for calculating time domain synchronization using different methods found in the literature. Our objective is to detail the program’s functionality and provide a clear use-case for implementation. The programs primary function “mu_synch” automatically performs the cross-correlation analysis based on user input. Automated peak detection methods such as the cumulative sum method and the z-score method, as well as subjective, visual analysis are available. Users can also define other parameters like the number of recurrence intervals to be used and histogram bin size. The function outputs six common synchronization indices, the common input strength (CIS), k’, k’-1, E, S, and Synch Index. This toolbox allows for better standardization of techniques and for more comprehensive data mining in the motor control community.

motoRneuron: an open-source R toolbox for time-domain motor unit analyses

Andrew J. Tweedell[1], Matthew S. Tenan[2]

[1]Human Research and Engineering Directorate, United States Army Research Laboratory, Aberdeen Proving Ground, Maryland, USA
[2]Defense Health Agency, Falls Church, Virginia, USA

Corresponding Author:
Andrew Tweedell
459 Mulberry Point Road, Aberdeen Proving Ground, MD, 21009, USA
Email address: andrew.j.tweedell.civ@mail.mil

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.27578v1 | CC0 Open Access | rec: 11 Mar 2019, publ: 11 Mar 2019

Abstract

Motor unit synchronization is the tendency of motor neurons and their associated muscle fibers to discharge near-simultaneously. It has been theorized as a control mechanism for force generation by common excitatory inputs to these motor neurons. Magnitude of synchronization is calculated from peaks in cross-correlation histograms between motor unit discharge trains. However, there are many different methods for detecting these peaks and even more indices for calculating synchronization from them. Methodology is typically laboratory-specific and requires expensive software, like Matlab or LabView. This lack of standardization makes it difficult to draw definitive conclusions about motor unit synchronization. To combat this, we have developed a freely available, open-source toolbox, "motoRneuron", for the R programming language. This toolbox contains functions for calculating time domain synchronization using different methods found in the literature. Our objective is to detail the program's functionality and provide a clear use-case for implementation. The programs primary function "mu_synch" automatically performs the cross-correlation analysis based on user input. Automated peak detection methods such as the cumulative sum method and the z-score method, as well as subjective, visual analysis are available. Users can also define other parameters like the number of recurrence intervals to be used and histogram bin size. The function outputs six common synchronization indices, the common input strength (CIS), k', k'-1, E, S, and Synch Index. This toolbox allows for better standardization of techniques and for more comprehensive data mining in the motor control community.

Introduction

Motor unit synchronization 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 (Farmer et al. 1997; Semmler 2002). It is often interpreted as an indicator of functional connectivity between motor neurons through common excitatory post-synaptic potentials (Sears & Stagg 1976). Typically, cross-correlation analyses are employed, whereby the discharge times of one motor unit are correlated against those of another concurrently active motor unit (*Fig*. 1) and a histogram is created based on these recurrence intervals. Peaks in the histogram represent a higher probability of a discharge from the response motor unit around that latency of the reference motor unit discharge (seen in *Fig*. 2B). Various indices are calculated from these peaks and their magnitude indicates the level of synchronization (for review see (Farmer et al. 1997; Semmler 2002; Farina & Negro 2015)). This time-domain synchronization appears to be a critical factor in force modulation. For example, synchronous activation of muscle fibers produces longer and greater twitch forces than if they were activated asynchronously (Merton 1954). In practice, this phenomenon is evidenced in strength-trained individuals, who display higher motor unit synchrony than untrained individuals do (Semmler & Nordstrom 1998; Fling et al. 2009). Although beneficial for producing high forces, synchronization has been shown to be detrimental to force steadiness (Yao et al. 2000). Thus, understanding motor unit synchronization seems to be important for modeling neuromuscular performance.

Over the last few decades, it has become much easier and cheaper to collect motor unit action potentials with either intramuscular or decomposed surface electromyography. Researchers have gone from examining synchronization in 2-3 motor units to 15+ in a single contraction (Schmied & Descarreaux 2010; Defreitas et al. 2014). Unfortunately, while data collection technology has improved and multiplied, so have the options for synchronization analysis. Reconciling results from different types of analyses remains difficult. Concerning the cross-correlation analysis, there are numerous ways in which to determine the size and location of peaks present in histograms. Methodology is largely laboratory specific, with some groups using automated methods like the z-score method or the cumulative sum method. Before automated methods were developed, subjective, visual analysis was used. Within these methods, parameters such as the number of orders of recurrence intervals used and histogram bin size are likely to vary as well. Additionally, there are a number of indices available to characterize synchronization magnitude. Common input strength (CIS) and k' ("k prime") are most often reported; however, the Synch Index (SI) and others are available. The lack of standardization in respect to motor unit synchronization hinders are ability to make definitive conclusions. Therefore, we have developed the open-source toolbox "motoRneuron" in the statistical programming language R (henceforth referred to as R) for the calculation of time-domain synchronization using various peak determination methods. This toolbox provides a list of functions to calculate recurrence intervals, create and plot cross-correlation histograms, and ultimately, calculate synchronization indices.

R has quickly risen in popularity recently because of its very active user/developer base that rapidly iterates to improve the functionality of the language. Typically, programs that perform synchronization analyses are handed down through laboratories using paid software like Matlab and LabView. Meanwhile, R and our toolbox are freely available. Source code for all R

functions are available to the user. MotoRneuron was created as a free, open-source platform with which users can perform all necessary functions to calculate synchronization, or alter to suit their unique needs. With the numerous ways to calculate synchronization, this toolbox allows for better standardization of techniques and for more comprehensive data mining in the motor control community. The objective is to detail the functionality of the motoRneuron toolbox for investigating motor unit time-domain synchronization.

Functionality and Application

R Programming Environment

MotoRneuron was developed in RStudio (v 1.1.453) using R (v 3.5.0) (R Development Core Team 2015). It was configured on a Windows 10 computer (Enterprise V. 1703. Intel® Core ™ 2.80 GHz, x64-based processor). The toolbox is under the GNU General Public License version 3. This paper will discuss general methods used and applications for the motoRneuron toolbox. For readers unfamiliar with the R language, sample motor unit data and R scripts with instructions are provided in the Supplementary Material. In addition, packages will always include help files for specific details about their functions. It is highly recommended to download R and RStudio in order to follow along with the sample scripts provided. Briefly, R uses command-line scripting to perform functions on data within the working environment. Common data formats for R are vectors, matrices, lists, and data frames, which can be imported into the working environment from any number of formats, including text or csv files. MotoRneuron leverages many functions not included in base R which are automatically incorporated by downloading the following add-on packages from Github or the Comprehensive R Archive Network (CRAN): 'dplyr', 'ggplot2', 'dygraphs', 'magrittr', and 'tseries' (Milton-Bache & Wickham 2014; Trapletti & Hornik 2018; Vanderkam et al. 2018; Wickham et al. 2018a; Wickham et al. 2018b).

To access motoRneuron through R and all the functions, sample data, and help files wherein, the following functions are called in the console of RStudio. "*install.packages*" will automatically download the package from CRAN. "*library*" will attach the packages items to your working environment for use.

```
> install.packages("motoRneuron")
> library(motoRneuron)
```

Package Implementation

In general, there are three steps involved in calculating time-domain synchronization. First, the cross-correlation histogram is created. Second, the size and location of the peak is determined. Last, synchronization indices are calculated based on the size of the peak. The primary function of motoRneuron is *mu_synch*, which completes all three steps based on the user's inputs. The function's syntax and five formal arguments are:

mu_synch(motor_unit_1, motor_unit_2, method, order, binwidth, plot)

Motor_unit_1 and *motor_unit_2* arguments are vectors of the discharge times of two motor unit action potential trains. Included in motoRneuron is a real world data set that will be used to reduce and analyze motor unit synchronization. The data set was selected from a previous study, with informed consent and in accordance with the United States Army Research Laboratory Institutional Review Board (approval number ARL 16-099), using fine wire electromyography from the flexor digitorum superficialis during a 30-second isometric finger flexion task. The data format is a data frame time series of two concurrently active motor units, named motor_unit_1 and motor_unit_2. Here we provide the code to read in the data and reduce into the constituent motor units discharge times for further use in the package.

```
> Sample_data <- motoRneuron::motor_unit_data
> motor_unit_1 <- as.vector(subset(Sample_data, select = Time, motor_unit_1
== 1))
> motor_unit_2 <- as.vector(subset(Sample_data, select = Time, motor_unit_2
== 1))
```

Below is sample output from R for the two motor unit discharge vectors showing the first six time points by calling the function *head*. For example, the first three discharge times for *motor_unit_1* are at 0.035, 0.115, and 0.183 seconds, while *motor_unit_2* discharged at 0.1, 0.205, and 0.298 seconds.

```
> 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 ...
```

The motor unit with fewer discharges is called the reference unit, while the other is referred to as the event unit. This distinction is made automatically within the function and is output as a part of the motor unit characteristics. *Method* indicates which method(s) of cross-correlation peak determination is to be used, while *order* and *binwidth* specifies how many orders of recurrences intervals to calculate and the size of the bins for the histogram, respectively. Some researchers argue that only first order intervals should be used for analysis, as the presence of harmonics within the cross-correlation may cause non-physiological peaks to appear in the long latency portions of the histogram (De Luca et al. 1993). Therefore, the default argument of *order* is set at 1, indicating only first order intervals are to be used; however, the function is flexible enough to handle any order input by the user. Additionally, the *binwidth* argument is set at a default of 0.001 sec or 1 ms. This allows for appropriate resolution in short-term synchronization

measurements. What is returned from this function is a list of individual motor unit characteristic data along with a list of all synchronization indices (detailed below). Characteristics for each motor unit included are the number of discharges, the mean interspike interval (ISI), all ISI's, and the intervals for each specified recurrence order. The *plot* argument takes a TRUE or FALSE to indicate whether the resulting histogram will be displayed or not.

Peak Determination Methods Available

The three methods employed in this toolbox reflect the three broad classes of cross-correlation histogram peak determination in the current motor unit synchronization literature. The *methods* argument allows the user to choose how a peak in the histogram will be determined. More specifically, this code automatically computes the boundary bins of the peak based on certain criteria.

The visual method - The bins of the histogram are progressively summed across from -100 to +100 ms and subsequently divided by the baseline mean count (considered as <= -60 and >= 60 ms) to produce a "normalized" cumulative sum graph. Large increases or decreases in bin counts are seen as large deflections in the cumulative sum graph. The user is asked to identify the peak as the beginning and end of this large deflection (Nordstrom et al. 1992). An example of this plot is shown in *Fig*. 2A, with boundaries chosen by a user highlighted. *Visual_mu_synch* is the function syntax to call this method separately.

The cumulative sum (cumsum) method – The bins of the histogram are progressively summed across from -100 to +100 ms to produce a cumulative sum. Peak boundaries are determined as the bins associated with 10 and 90% of the range (maximum - minimum) of this cumulative sum. The peak is considered significant if it's mean bin count exceeds the sum of the mean and 1.96 times the standard deviation of the baseline bins (considered as <= -60 and >= 60 ms). If no significant peak is detected, a default peak is used as ±5 ms (Keen et al. 2012). *Cumsum_mu_synch* is the function syntax to call this method separately.

The z-score method - First, a random uniform distribution is used to create a cross-correlation based on parameters from the experimental data. This new, "shuffled" histogram depicts the correlation of two motor unit trains that are completely independent (i.e. flat). It is used to calculate a significance threshold (Equation A) to compare with the experimental data.

$$A)\ Significance\ Threshold = \ mean\ bin\ count\ + (1.96 * standard\ deviation\ of\ bin\ count)$$

Any bins in the experimental histogram within ±10 ms of 0 that crosses this threshold are considered to be significantly greater than expected due to chance and subsequently used for analysis. If no peak is detected, synchronization indices of 0 are returned. Because the z-score method tests each bin individually, peak bins are not necessarily adjacent (Defreitas et al. 2014). *Zscore_mu_synch* is the function syntax to call this method separately.

The following R scripts calls the *mu_synch* function to perform all three methods for first order recurrence intervals with a bin size of 1 ms. Each individual method can also be called separately with their respective functions.

```
> mu_synch(motor_unit_1, motor_unit_2, method = c("Visual", "Zscore",
"Cumsum"), order = 1, binwidth = 0.001, plot = FALSE)
```

Recurrence_intervals and *bin* are support functions used within the synchronization functions to compute the recurrence intervals and discretize the data for the histogram, but they can also be called separately for individual use. A *plot_bins* function is also available that will display the associated histogram in the Plot window of RStudio (*Fig*. 3D). This is useful for visually checking data for abnormalities prior to calculating synchronization. The code below creates an R list named '*first_order_intervals*' that contains the motor unit characteristic data along with the first order recurrence intervals.

```
> first_order_intervals <- recurrence_intervals(motor_unit_1, motor_unit_2,
order = 1)
```

To access just the intervals, we need to index them using the '$' operator. Below, we use the *head* function again just to view the first six elements of the first order intervals.

```
> head(first_order_intervals$`1`)
```

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

Now these intervals are input to the *bin* function, along with the user-defined bin width, to discretize the intervals into bins for the detection peaks. A data frame '*binned_data'* is created with the code using a bin width of 1 ms. The resulting data frame contains a column depicting the bin, or the amount of time in second before (negative) or after (positive) the reference motor unit discharge, and the frequency of occurrence at that interval. This data frame can be put directly into the *plot_bins* function to display the histogram (such as in *Fig*. 3D).

```
> binned_data <- bin(first_order_intervals$`1`, binwidth = 0.001)
> head(binned_data)
```

```
##  Bin Freq
```

```
1 -0.101    1
```

```
2 -0.100    0
```

```
3 -0.099    0
```

```
4 -0.098    0
5 -0.097    0
6 -0.096    0
> plot_bins(binned_data)
```

Synchronization Indices

Once the boundaries of the peak are established, the synchronization indices are calculated. Six indices that are commonly found throughout the literature are automatically returned (Nordstrom et al. 1992; De Luca et al. 1993; Kamen & Roy 2000). The peak of the histogram can be considered two different regions; the region of counts that are expected due to chance and the region containing "extra" counts more than what is expected due to chance (*Fig*. 2B). These extra counts are typically the number of counts in the peak bins over a certain threshold. In the Z-score method, this is the significance threshold calculated from the shuffled histogram (Defreitas et al. 2014). In the Visual method, the threshold is the baseline mean bin count (Nordstrom et al. 1992). The total counts in peak consists of the summation of the regions. Synchronization indices attempt to quantify the relationship between these different regions. The larger the magnitude of the indices, the higher the chances that the motor units are firing in synchronization. The CIS index is commonly used because it allows for normalization with respect to trial duration. Nordstrom *et al.* developed the CIS because most other indices available at that time were influenced by discharge rate (Nordstrom et al. 1992). The equations used to calculate the various indices are below.

$$CIS = \frac{extra\ counts\ in\ peak}{duration\ of\ trial\ (s)}$$

$$k' = \frac{total\ counts\ in\ peak}{expected\ counts\ in\ peak}$$

$$k' - 1 = \frac{extra\ counts\ in\ peak}{expected\ counts\ in\ peak}$$

$$E = \frac{extra\ counts\ in\ peak}{number\ of\ discharges\ from\ reference\ motor\ unit}$$

$$S = \frac{extra\ counts\ in\ peak}{total\ number\ of\ discharges\ from\ both\ motor\ units}$$

$$SI = \frac{extra\ counts\ in\ peak}{\frac{total\ counts}{2}}$$

Along with the synchronization indices listed above, also reported is the peak duration and peak center. These refer to the width of the peak and the bin location of the center of the peak, respectively, to help characterize the latency of synchronization. Below is some example R

output of the synchronization results of data obtained from the Cumulative Sum technique for motor unit synchronization. Here we see the CIS between the example motor units is 2.16, which indicates 2.16 synchronous discharges per second. The peak duration was 10 ms centered at bin 0.

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

An advantage to R, as alluded to before, is its robust statistical computing. Using the R environment allows for direct access to many statistical packages. The “stats” package comes included in base R so many statistical tests are immediately available for testing synchronization metrics. This eliminates the need for transforming and importing data into 3rd party statistical software, such as SAS and SPSS. Simple tests such as t-tests and ANOVAS are common, while more complex, multi-level models are available.

Bugs or errors in software are common in open-source scripted codes like R. As this is the first stable version of the motoRneuron package, it is possible that users will notice performance issues or errors stemming from R version fragmentation or other sources. Users are urged to

email any errors or issues found in motoRneuron to the package maintainer (Andrew Tweedell andrew.j.tweedell.civ@mail.mil). Errors that can be fixed will be updated in new versions of the package as they are found. As such, it is important to update the package continually to guarantee efficient performance.

Discussion

MotoRneuron is a free package containing a list of functions capable of performing many different cross-correlation analyses for calculating many time-domain synchronization metrics for use in the motor control field. This free, all-inclusive software package enables researchers to easily examine the many options for calculating and reporting synchronization indices. Additionally, new data can be quickly reconciled with results from previous studies for better physiologic interpretation. MotoRneuron is written in the R programming language, which provides an open-source platform to perform data and statistical analysis on motor unit data. MotoRneuron's simplistic function syntax and detailed output allow for easy comprehension. The package also allows for the visualization of these analyses through R's powerful and flexibility graphics capabilities. In the future, the package will be expanded to include frequency-domain characterization as well.

Citation

Researchers using motoRneuron in a published paper should cite this article and indicate the used version of the package.

Acknowledgements

The authors would like to acknowledge Courtney Haynes for her work as technical reviewer for the manuscript. The authors would also like to acknowledge the entire R community for providing a free platform for the creation and distribution of this package to the greater scientific community.

References

De Luca CJ, Roy AM, and Erim Z. 1993. Synchronization of motor-unit firings in several human muscles. *Journal of Neurophysiology* 70:2010-2023.

Defreitas JM, Beck TW, Ye X, and Stock MS. 2014. Synchronization of low- and high-threshold motor units. *Muscle Nerve* 49:575-583. 10.1002/mus.23978

Farina D, and Negro F. 2015. Common synaptic input to motor neurons, motor unit synchronization, and force control. *Exercise and Sport Sciences Reviews* 43:23-33. 10.1249/JES.0000000000000032

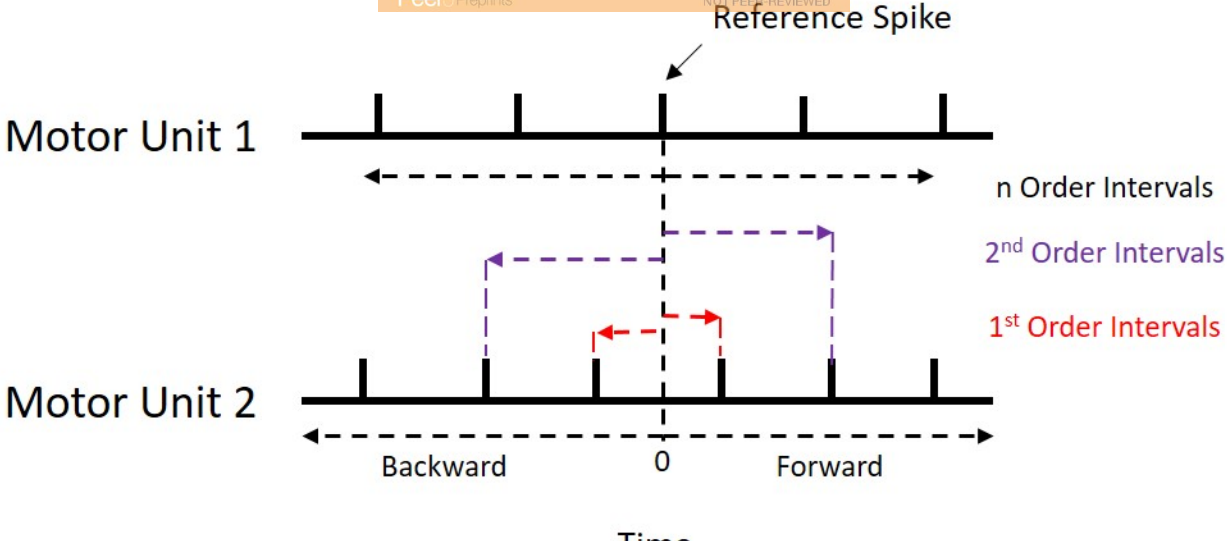
Farmer SF, Halliday DM, Conway BA, Stephens JA, and Rosenberg JR. 1997. A review of recent applications of cross-correlation methodologies to human motor unit recording. *Journal of Neuroscience Methods* 74:175-187.

344 Fling BW, Christie A, and Kamen G. 2009. Motor unit synchronization in FDI and biceps
345 brachii muscles of strength-trained males. *Journal of Electromyography and Kinesiology*
346 19:800-809. 10.1016/j.jelekin.2008.06.003
347 Kamen G, and Roy A. 2000. Motor unit synchronization in young and elderly adults. *European*
348 *journal of applied physiology* 81:403-410. 10.1007/s004210050061
349 Keen DA, Chou LW, Nordstrom MA, and Fuglevand AJ. 2012. Short-term synchrony in diverse
350 motor nuclei presumed to receive different extents of direct cortical input. *Journal of*
351 *Neurophysiology* 108:3264-3275. 10.1152/jn.01154.2011
352 Merton PA. 1954. Interaction between muscle fibres in a twitch. *Journal of Physiology* 124:311-
353 324.
354 Milton-Bache S, and Wickham H. 2014. magrittr: A Forward-Pipe Operator for R. 1.5 ed.
355 Nordstrom MA, Fuglevand AJ, and Enoka RM. 1992. Estimating the strength of common input
356 to human motoneurons from the cross-correlogram. *Journal of Physiology* 453:547-574.
357 R Development Core Team. 2015. R: A language and environment for statistical computing.
358 Vienna, Austria.: R Foundation for Statistical Computing. .
359 Schmied A, and Descarreaux M. 2010. Influence of contraction strength on single motor unit
360 synchronous activity. *Clinical Neurophysiology* 121:1624-1632.
361 10.1016/j.clinph.2010.02.165
362 Sears TA, and Stagg D. 1976. Short-term synchronization of intercostal motoneurone activity.
363 *Journal of Physiology* 263:357-381.
364 Semmler JG. 2002. Motor unit synchronization and neuromuscular performance. *Exercise and*
365 *Sport Sciences Reviews* 30:8-14.
366 Semmler JG, and Nordstrom MA. 1998. Motor unit discharge and force tremor in skill- and
367 strength-trained individuals. *Experimental Brain Research* 119:27-38.
368 Trapletti A, and Hornik K. 2018. tseries: Time Series Analysis and Computational Finance. 0.10-
369 45. ed.
370 Vanderkam D, Shevtsov P, Allaire J, Owen J, Gromer D, Thieurmel B, and Laukhuf K. 2018.
371 dygraphs: Interface to 'Dygraphs' Interactive Time Series Charting Library. 1.1.1.6 ed.
372 Wickham H, Chang W, Henry L, Lin Pedersen T, Takahashi K, Wilke C, and Woo K. 2018a.
373 ggplot2: Create Elegant Data Visualisations Using the Grammar of Graphics. 3.0.0 ed.
374 Wickham H, Francois R, Henry L, and Muller K. 2018b. dplyr: A Grammar of Data
375 Manipulation. 0.7.6 ed.
376 Yao W, Fuglevand RJ, and Enoka RM. 2000. Motor-unit synchronization increases EMG
377 amplitude and decreases force steadiness of simulated contractions. *Journal of*
378 *Neurophysiology* 83:441-452. 10.1152/jn.2000.83.1.441
379

Figure 1(on next page)

Recurrence Intervals Diagram

Schematic representation of the recurrence intervals between two concurrently active motor units. Each discharge from one motor unit is used as a reference point to determine forward and backward latencies to the discharges of the second motor unit. The first order intervals are the latencies to the *first* forward and backward discharges (noted in red). The second order are the *second* forward and backward discharges (noted in purple).

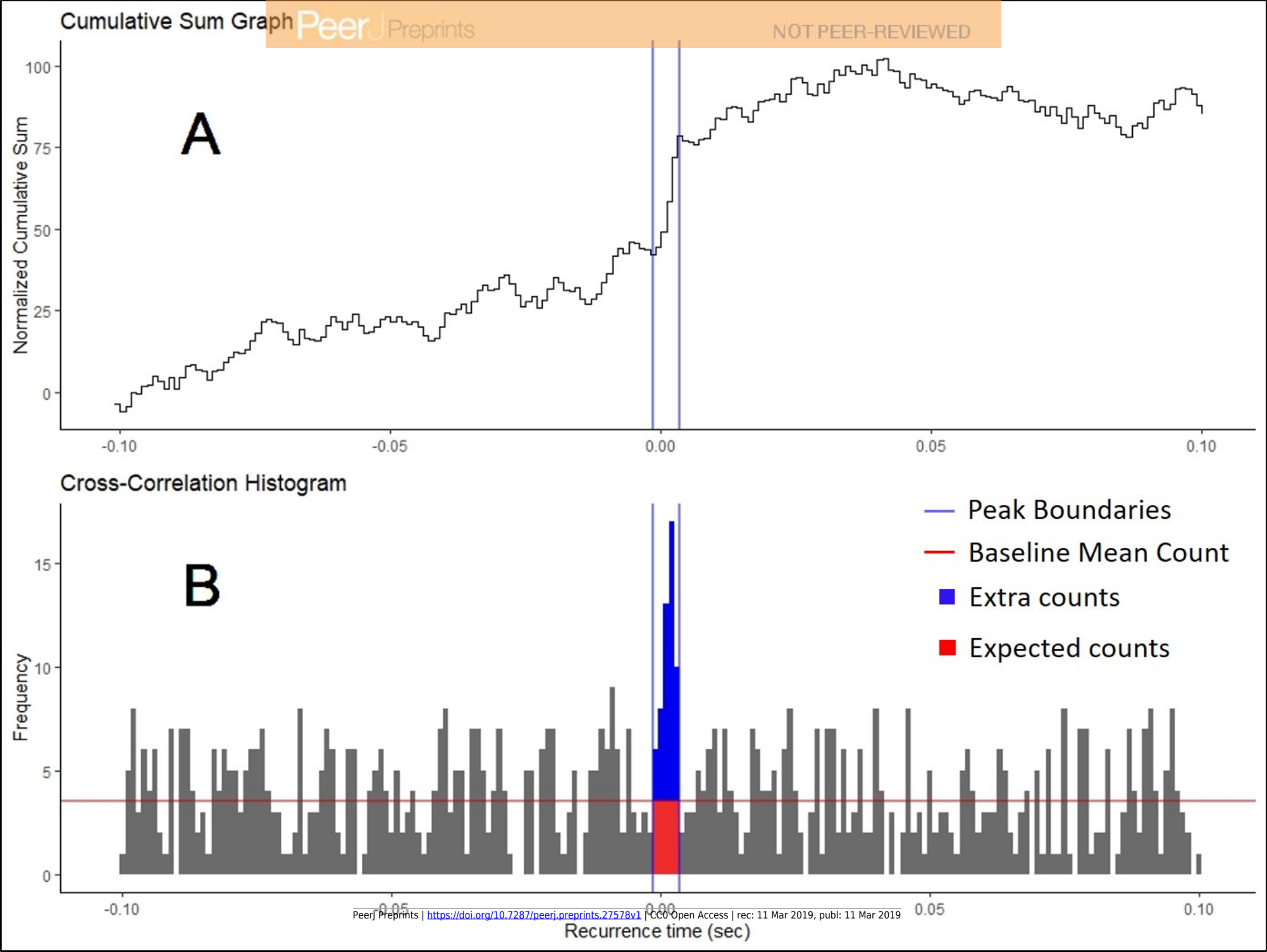


PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.27578v1 | CC0 Open Access | rec: 11 Mar 2019, publ: 11 Mar 2019

Figure 2(on next page)

Cumulative Sum and Histogram

(A) Example of cumulative sum graph of bin counts. (B) Cross-correlation histogram rendered in RStudio using the “ggplot” package. Peak boundaries (blue lines) were determined by visual analysis of the cumulative sum graph, where a peak is seen as a large deflection near time 0. Synchronization indices are calculated based on the relationship between the counts of the histogram expected due to chance (in red) and the counts that are in excess of what is expected (in blue). In most cases, this threshold for determination (red line) is the baseline mean count of the histogram.



PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.27578v1 | CC0 Open Access | rec: 11 Mar 2019, publ: 11 Mar 2019

Figure 3(on next page)

RStudio Graphical User Interface

R integrated development environment RStudio's graphical user interface. The interface is made up of four panels: (A) R script panel, (B) Console, (C) Global Environment, (D) Plot panel depicting an example cross-correlation histogram.

RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

Go to file/function | Addins | Project: (None)

Untitled1*

Source on Save | Run | Source

```
1  This is where you write your own code!
```

A

1:39 (Top Level) R Script

Console Terminal

```
> This is where you execute your functions!
```

B

Environment History Connections

Import Dataset | List

Global Environment

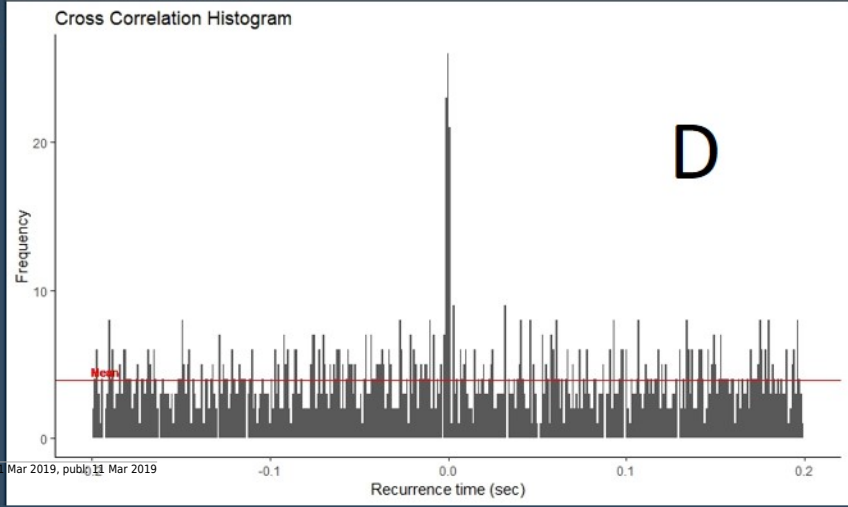
Values

```
motor_unit_1    num [1:443] 0.035 0.115 0.183 0.25 0.306 0.377 0.455 0.512 0.577 0.6…
motor_unit_2    num [1:307] 0.1 0.205 0.298 0.377 0.471 0.577 0.664 0.739 0.843 0.94…
```

C

Files Plots Packages Help Viewer

Zoom | Export



PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.27578v1 | CC0 Open Access | rec: 11 Mar 2019, published: 11 Mar 2019