Package 'UnitEvents'

January 12, 2018

Type Package

Title Unitary Events Method with Delayed Coincidence Count, with Gaussian Approximation (MTGAUE) or Permutation Method for Statistical Tests in Neuroscience

Version 0.0.2

Date 2018-01-12 10:53:50 +0100

Author Melisande Albert <melisande.albert@math.univ-toulouse.fr>,
Chahrazade Bahbah,
Yann Bouret <yann.bouret@gmail.com>,
Imen Chaarana <imen.chr@gmail.com>,
Julien Chevallier <julien.chevallier@u-cergy.fr>,
Magalie Fromont <magalie.fromont@univ-rennes1.fr>,
Franck Grammont <grammont@unice.fr>,
H. Issarane,
Thomas Laloe <laloe@unice.fr>,
I. Mouline,
Patricia Reynaud-Bouret <reynaudb@unice.fr>,
Amel Rouis <rouis.a@chu-nice.fr>,
Christine Tuleau-Malot <malot@unice.fr>,
Khalil Zahri

Maintainer Gilles Scarella <gilles.scarella@unice.fr>

Description Unitary Events Method with Delayed Coincidence Count, with Gaussian Approximation (MTGAUE) or Permutation Method. Performs statistical independence tests between two neurons based on coincidence counts. Using C++ neuro-stat code.

License GPL (>=2)

URL https://github.com/ybouret/neuro-stat,
http://math.unice.fr/~malot/liste-MTGAUE.html,
https://sourcesup.renater.fr/docman/?group_id=3267

Imports methods

VignetteBuilder knitr

Suggests knitr

R topics documented:

`BH` *This function performs Benjamini-Hochberg procedure.*

Description

This function allows to identify the time windows detected by the MTGAUE or permutation procedure.

Usage

```
BH(alpha, p, plot)
```

Arguments

`alpha`	double; Array of alpha values or maximum values for the false positive. Could be a vector (in case of several tests)
`p`	double; Array of p-values. Needs to be ordered.
`plot`	bool; To plot the result or not. Default value is FALSE.

Value

`k`	The index obtained by Benjamini-Hochberg procedure. Could be an array

Examples

```
p <- runif(10)
alpha <- 0.05
BH(alpha, p, FALSE)
p <- c(0.22, 0.25, 0.41, 0.11, 0.00932, 0.17, 0.0049, 0.01641,
     0.166, 0.17)
alpha <- c(0.075, 0.1, 0.025, 0.05)
#should return 3 3 0 2 values ( [7, 5, 8] , [7,5])
resu <- BH(alpha, p, plot=TRUE)
```

---

compute_pvalues *Computation of the p-values associated to the GAUE symmetric test*

Description

This function computes the p-values associated to the GAUE symmetric test and the numerator of this test.

Usage

```
compute_pvalues(C, ntrials, a, b, delay, test)
```

Arguments

C	matrix with 3 rows. It corresponds to the output of the function wink_coincmat
ntrials	int; the number of trials of the datasets used to compute the average coincidence count with delay
a	lower bound of the time window
b	upper bound of the time window
delay	double; value used in coincidence detection
test	string; To define the test; only possible values are "all", "symmetric", "upper" and "lower". Default value is "all" (upper values appear in red, while lower ones appear in blue)

Value

The output of this function is a list with two elements, the first one is the p-value associated to the GAUE symmetric test and the second one is the numerator of the statistic test.

Examples

```
a <- 0 # lowest time
b <- 2 # highest time
ntrials <- 1
delay <- 0.02
C <- matrix(ncol=1,c(0,1,0.5))
test <- "symmetric"
compute_pvalues(C, ntrials, a, b, delay, test)
```

compute_time_windows

To create time windows spanning the whole interval of observation

Description

To create time windows of the same duration spanning the whole interval of observation

Usage

```
compute_time_windows(a, b, duration, spacing)
```

Arguments

a	Beginning of recordings
b	End of recordings
duration	Duration of a time window. Identical for every time window.
spacing	Value to define a shift between time windows. If spacing=duration, there is no overlap.

Value

Returns a list containing time windows.

- A: Matrix of time windows with two rows. Each column is a time window. The number of columns of A is equal to the number of time windows.
- L: Time step or spacing between the starts of two consecutive time windows. Equal to input argument spacing.
- len: length or number of time windows

Examples

```
# duration of each time window is 0.01, no overlapping
TW1 <- compute_time_windows(a=0, b=1, duration=0.01, spacing=0.01)
print(TW1)

# with overlapping
TW2 <- compute_time_windows(a=0, b=1, duration=0.1, spacing=0.05)
print(TW2)

# limit case 1
TW3 <- compute_time_windows(a=0, b=2, duration=2, spacing=0.5)
print(TW3)

# # limit case 2
TW4 <- compute_time_windows(a=0, b=1, duration=2, spacing=0.1)
print(TW4)

# # limit case 3
TW5 <- compute_time_windows(a=0, b=1, duration=0.1, spacing=0.2)
print(TW5)
```

counting_spikes *To compute the number of spikes for each trial.*

Description

To compute the number of spikes for each trial. It modifies the input matrix by adding an additional first column equal to the number of spikes for each trial.

Usage

```
counting_spikes(M)
```

Arguments

M	matrix; input matrix containing the spike times of a neuron without spike count.

Value

The output is the modified input matrix with an additional first column containing the number of spikes for each trial.

Examples

```
M <- matrix(c(0.5, 0.51, 1.1, 1.09, 0, 1.12), nrow=2)
M <- counting_spikes(M)
print(M)
```

DNeur *To create DataNeur from input neuron files.*

Description

To create DataNeur from input neuron files. A DataNeur is a matrix of size nneurons*ntrials-by-(nmax+1) containing spike times, where nneurons is the number of neurons, ntrials is the number of trials and nmax is the maximum number of spikes for every neuron and trial. For a missing value, 0 is used to fill the matrix. The first column of the matrix is the number of spikes for each trial.

Usage

```
DNeur(linput=list("Neur1_c13.txt","Neur2_c13.txt"), DName="",
listOptions=list())
```

Arguments

linput	list; Either a list of two character vectors corresponding to the two file names containing the spike times of the two neurons or a list of two matrices containing the spike times of the two neurons
DName	string; Name of the DataNeur (used in plot captions). Default value is NULL.
listOptions	list; List of options to define the DataNeur. Names of the list are

- removeCols Either a function or an array of column indices to be removed from the raw matrix. In case of a function, it should apply to any row of the matrix.
- removeRows Either a function or an array of row indices to be removed from the raw matrix. In case of a function, it should apply to any column of the matrix.
- scaling Numeric; a scalar value to rescale the matrix. The result will be M*scaling if M is the initial matrix.
- shift Numeric; a scalar value to shift the matrix. The result will be M+shift if M is the initial matrix.

Value

The output is a matrix containing neuron spike times, of size nneurons*ntrials-by-(nmax+1), where nneurons is the number of neurons, ntrials is the number of trials and nmax is the maximum number of spikes for every neuron and trial. For a missing value, 0 is used to fill the matrix. The first column of the matrix is the number of spikes for each trial. Warning: each neuron should have the same number of trials!

See Also

DNeurFile.matrix, DNeurMatrix.matrix, counting_spikes, use_options

Examples

```
D1 <- DNeur(list('UnitEvents/data/Neur1_c13.txt',
          'UnitEvents/data/Neur2_c13.txt'))
if (is.matrix(D1)) print(D1[c(1:5,200:204),1:10])
M1 <- matrix(c(0.5, 0.51, 1.1, 1.09), nrow=2)
M2 <- M1 + 0.02
M2 <- cbind(M2, c(0, 1.24))
D2 <- DNeur(list(M1, M2))
if (is.matrix(D2))  print(D2)
```

DNeurFile.matrix *To create a DataNeur (a description of an experiment) from an input neuron file using DataNeur version as "matrix" (DataNeur is a matrix)*

Description

To create a DataNeur (a description of an experiment) from an input neuron file. For a missing value, 0 is used to fill matrices.

Usage

```
DNeurFile.matrix(neurfile, listOptions=list())
```

Arguments

neurfile	string; Character vector as a file name containing the spike times of a neuron
listOptions	list; List of options to define the DataNeur. May contain a scaling factor or a function to remove columns. Default value is the empty list.

Value

The output is a matrix containing spike times, of size ntrials-by-(nmax+1), where ntrials is the number of trials and nmax is the maximum number of spikes over the trials. For a missing value, 0 is used to fill the matrix. The first column of the matrix is the number of spikes for each trial.

See Also

use_options, counting_spikes, DNeur

Examples

```
rm1standlast7col <- function(v) { return(v[-c(1,(length(v)-6):length(v))])}
scaling <- 1/10000
D <- DNeurFile.matrix('UnitEvents/data/Neur1_c13.txt',
list('removeCols'=rm1standlast7col, 'scaling'=c(scaling)))
```

DNeurMatrix.matrix *To create a DataNeur (a description of an experiment) using an input matrix.*

Description

To create a DataNeur (a description of an experiment) using an input matrix. A DataNeur contains a matrix of size ntrials-by-nmax containing spike times. nmax is the maximum number of spikes over the trials. For a missing value, 0 is used to fill the matrix.

Usage

```
DNeurMatrix.matrix(M, listOptions=list())
```

Arguments

M	matrix; input matrix containing the spike times of a neuron
listOptions	list; List of options to define the DataNeur. May contain a scaling factor or a function to remove columns. Default value is the empty list.

Value

The output is a matrix containing spike times, of size ntrials-by-(nmax+1), where ntrials is the number of trials and nmax is the maximum number of spikes over the trials. For a missing value, 0 is used to fill the matrix. The first column of the matrix is the number of spikes for each trial.

See Also

use_options, counting_spikes

Examples

```
M1 <- matrix(c(0.5, 0.51, 1.1, 1.09), nrow=2)
M1 <- cbind(M1, c(0, 1.24))
D1 <- DNeurMatrix.matrix(M1)
```

draw_windows — *To draw time windows. To highlight detected windows after test.*

Description

To draw time windows. Called after Benjamin-Hochberg procedure. Fill and border colors and density can be modified (see `polygon` function). Default values are "yellow" for fill color, "black" for border and no density.

Usage

```
draw_windows(A, xrange, yrange, col="yellow", density="",
             border="black")
```

Arguments

`A`	double; Time window matrix (with two rows). For example it is the matrix of detected windows after Benjamini-Hochberg procedure
`xrange`	Double; Numeric vector of length two to define the x-coordinate range in the plot.
`yrange`	Double; Numeric vector of length two to define the y-coordinate range in the plot
`col`	string; Fill color in the call to polygon function. Default value is "yellow".
`density`	numeric; Density in the call to polygon function. No density by default.
`border`	string; Border color in the call to polygon function. Default value is "black".

Examples

```
par(mfrow=c(1,2))
s = seq(0.001, 0.9, 0.1)
A = matrix(ncol=length(s), nrow=2)
A[1,] = s
A[2,] = s+0.1
xrange = c(0,max(A))
yrange = c(-25,0)
draw_windows(A, xrange, yrange, density=c(0), col="black")
draw_windows(A, xrange, yrange, col="lightgreen", border=NA,
   density=c(11))
```

findUE — *To find Unitary Events in detected time windows with a given delay*

Description

To find Unitary Events in detected time windows with a given delay

Usage

```
findUE(A, DataNeur, delay)
```

Arguments

A	Matrix of detected time windows. Contains two rows. It should contain detected time windows after Benjamini-Hochberg procedure.
DataNeur	A DataNeur e.g. the output of DNeur function. Matrix of spike times, of size nneurons*ntrials-by-(nmax+1), where nneurons is the number of neurons, ntrials is the number of trials and nmax is the maximum number of spikes for every neuron and trial. For a missing value, 0 is used to fill the matrix. The first column of the matrix is the number of spikes for each trial.
delay	double; delay is the value for delay used in coincidence detection.

Value

The output is a list with 2 elements

- N1: Matrix with three rows; the first row contains detected spike times of the first neuron, the second one is for the trial index and the last one is the type of detection (-1 for a negative one, 1 for a positive one)
- N2: Matrix with three rows; the first row contains detected spike times of the second neuron, the second one is for the trial index and the last one is the type of detection (-1 for a negative one, 1 for a positive one)

Examples

```
TW = compute_time_windows(a=1e-3, b=2.1, duration=1e-1, spacing=0.05)
rm1standlast7col <- function(v) { return(v[-c(1,(length(v)-6):length(v))])}
scaling <- 1/10000
neur1file = "UnitEvents/data/Neur1_c13.txt"
neur2file = "UnitEvents/data/Neur2_c13.txt"
DataNeur = DNeur(list(neur1file, neur2file),
 listOptions=list('removeCols'=rm1standlast7col, 'scaling'=c(scaling)))
delay = 0.02
A = matrix(nrow = 3, ncol=ncol(TW$A)) #every time window from A
                                      # is considered
A[1:2,] = TW$A
T = findUE(A, DataNeur, delay)
```

---

minmaxtimes — *Function to compute the minimum and maximum time values of a given experiment for a neuron.*

Description

Function to compute the minimum and maximum time values of a given experiment for a neuron. It is especially useful for plotting or defining the windows to test on.

Usage

```
minmaxtimes(M)
```

Arguments

M	matrix; A DataNeur matrix e.g. the output of DNeur function - Contains spike times and their count per trial

Value

numeric; Minimum and maximum time values for a given neuron

Examples

```
## Not run:
# creation of spikes
lambda <- 50 #firing rate
a <- 0; b <- 2 # start and end recording times
ntrials <- 10
DN = spikes.inject(lambda, a, b, ntrials)
print(paste('Minimum and maximum time values are', as.character(minmaxtimes(DN))))

## End(Not run)
```

`MulTestsBH` *This function performs multiple statistical tests to detect synchronization, either using Gaussian Approximation (MTGAUE) or permutation method. It runs the Benjamini-Hochberg procedure.*

Description

This function performs multiple statistical tests to detect synchronization (or dependence), either using Gaussian Approximation (MTGAUE) or permutation method.

Usage

```
MulTestsBH(DataNeur, A, delay, level, Rtest="all",
   iperm=FALSE, B=10000, num_threads=1, statistical_value='H',
   neurostatpath = "")
```

Arguments

`DataNeur`	A DataNeur e.g. the output of `DNeur` function. Matrix of spike times, of size nneurons*ntrials-by-(nmax+1), where nneurons is the number of neurons, ntrials is the number of trials and nmax is the maximum number of spikes for every neuron and trial. For a missing value, 0 is used to fill the matrix. The first column of the matrix is the number of spikes for each trial.
`A`	A two-row matrix containing time windows. Each column is a time window. The number of columns of A is equal to the number of time windows.
`delay`	double; delay is the value used in the coincidence count.
`level`	double; level is the alpha value used in Benjamin-Hochberg procedure.
`Rtest`	string; Rtest allows to define the type of the test. Only possible values are "all", "symmetric", "upper", "lower". Default value is "all" which means that upper and lower tests are performed and discernible.
`iperm`	Logical to select Permutation method or not. Default is FALSE: MTGAUE method, which performs more rapidly, is chosen but can be dubious if no stationarity. Permutation method takes more time but is more robust.
`B`	integer; size of Monte Carlo sample for Permutation method. Default is 10000.

`num_threads` integer; Number of threads for neuro-stat computation. Default is 1.

`statistical_value`
: string; Only possible values are "T" or "H". Default is "H". Corresponds to a centered computation or not.

`neurostatpath`
: string; It allows to define the path of neuro-stat code if necessary. It could be a relative or an absolute path. Default value is empty.

Value

A three-row matrix containing the detected time windows after Benjamini-Hochberg procedure. The first two rows contain time values of the detected windows and the last row contains the detection type: -1 for a negative, one meaning that the coincidence count is smaller than the expected one for the time window, 1 for a positive detection, meaning that the coincidence count is bigger than the expected one for the time window.

`Neur1_c13` *Spike times for the 1st neuron of pair 13*

Description

Matrix of spike times. Times are in tenths of milliseconds. First and last seven columns contain the signal times.

Usage

```
data(Neur1_c13)
```

Format

matrix

Source

unknown

`Neur1_c40` *Spike times for the 1st neuron of pair 40*

Description

Matrix of spike times. Times are in tenths of milliseconds. First and last seven columns contain the signal times.

Usage

```
data(Neur1_c40)
```

Format

matrix

Source

unknown

`Neur2_c13` *Spike times for the 2nd neuron of pair 13*

Description

Matrix of spike times. Times are in tenths of milliseconds. First and last seven columns contain the signal times.

Usage

```
data(Neur2_c13)
```

Format

matrix

Source

unknown

`Neur2_c40` *Spike times for the 2nd neuron of pair 40*

Description

Matrix of spike times. Times are in tenths of milliseconds. First and last seven columns contain the signal times.

Usage

```
data(Neur2_c40)
```

Format

matrix

Source

unknown

`spikes.inject` *To create a DataNeur using given arguments.*

Description

To create a DataNeur using given arguments, such as firing rate, start and end recording times, number of trials. Injection function

Usage

```
spikes.inject(nneurons, lambda, a, b, ntrials)
```

Arguments

`nneurons`	integer; Number of neurons
`lambda`	double; Firing rate
`a`	double; Recording start time
`b`	double; Recording end time
`ntrials`	Number of trials. Should be the same for each neuron

Value

The output is a matrix containing the spikes times of a neuron, of size nneurons*ntrials-by-(nmax+1), where nneurons is the number of neurons, ntrials is the number of trials and nmax is the maximum number of spikes for every neuron and trial. For a missing value, 0 is used to fill the matrix. The first column of the matrix is the number of spikes for each trial.

See Also

`DNeur`, `spikes.inject.single`

Examples

```
D <- spikes.inject(2, 50, 0, 2, 100)
spikes.plot(D, v=1:2, 2, xrange=minmaxtimes(D))
dev.new()
D <- spikes.inject(1, 50, 0, 2, 100)
spikes.plot(D, v=1, 1, xrange=minmaxtimes(D))
```

`spikes.inject.single`
To create a DataNeur for only one neuron

Description

To create a DataNeur using given arguments, such as firing rate, start and end recording times, number of trials. Injection function

Usage

```
spikes.inject.single(lambda, a, b, ntrials)
```

Arguments

lambda	double; Firing rate
a	double; Recording start time
b	double; Recording end time
ntrials	Number of trials

Value

The output is a matrix containing the spikes times of a neuron, of size ntrials-by-(nmax+1), where ntrials is the number of trials and nmax is the maximum number of spikes for every trial. For a missing value, 0 is used to fill the matrix. The first column of the matrix is the number of spikes for each trial.

See Also

DNeur, spikes.inject

Examples

```
D <- spikes.inject.single(50, 0, 2, 100)
spikes.plot(D, v=1:1, 1, xrange=minmaxtimes(D))
```

---

spikes.plot *To plot neuron spikes*

Description

To plot neuron spikes. All the neurons or a part of them can be represented. Spikes of the first neuron are represented at the bottom in grey color, while spikes of the second neuron are represented at the top in green color.

Usage

```
spikes.plot(DataNeur, v=1:n, n, xrange, yrange, title='',
            col=c('gray','green3'))
```

Arguments

DataNeur	A DataNeur e.g. the output of DNeur function - Contains the matrices of spike times. May be a list or a matrix
v	numeric; A vector of indices corresponding to the neurons to plot. Should have a size less than or equal to n.
n	The number of neurons used in the DataNeur (as a matrix)
xrange	Double; Numeric vector of length two to define the x-coordinate range in the plot.

yrange	Double; Numeric vector of length two to define the y-coordinate range in the plot
title	string; Title of the figure. Default value is an empty string.
col	Character vector; Vector of length two to define colors for plotting spikes. Default value is c('gray', 'green3').

Examples

```
# creation of spikes
lambda <- 50 #firing rate
a <- 0; b <- 2 # start and end recording times
ntrials <- 10
D = spikes.inject(2, lambda, a, b, ntrials)
xrange <- minmaxtimes(D)
yrange <- c(0,2*ntrials+2)  # number of trials
title <- "Neuro"
spikes.plot(D, v=1:2, 2, xrange, yrange, title)
dev.new()
spikes.plot(D, v=1:2, 2, xrange, yrange, title, col=c('gray','black'))
```

---

spikes.plot.matrix *To plot neuron spikes (DataNeur is a matrix)*

Description

To plot neuron spikes (DataNeur is a matrix). All the neurons or a part of them can be represented. Spikes of the first neuron are represented at the bottom in grey color, while spikes of the second neuron are represented at the top in green color.

Usage

```
spikes.plot.matrix(DataNeur, v, n, xrange, yrange, title='',
                   col=c('gray','green3'))
```

Arguments

DataNeur	A DataNeur e.g. the output of DNeur function - Contains the matrices of spike times as a global matrix
v	numeric; A vector of indices corresponding to the neurons to plot. Should have a size less than or equal to n.
n	The number of neurons used in the DataNeur (as a matrix)
xrange	Double; Numeric vector of length two to define the x-coordinate range in the plot.
yrange	Double; Numeric vector of length two to define the y-coordinate range in the plot
title	string; Title of the figure. Default value is an empty string.
col	Character vector; Vector of length two to define colors for plotting spikes. Default value is c('gray', 'green3').

Examples

```
# creation of spikes
lambda <- 50 #firing rate
a <- 0; b <- 2 # start and end recording times
ntrials <- 10
D = spikes.inject(2, lambda, a, b, ntrials)
xrange <- minmaxtimes(D)
yrange <- c(0,2*ntrials+2)  # number of trials
title <- "Neuro"
spikes.plot.matrix(D, v=1:2, 2, xrange, yrange, title)
dev.new()
spikes.plot.matrix(D, v=1:2, 2, xrange, yrange, title,
        col=c('gray','black'))
```

---

spikes.plot.single *To plot spikes of a given neuron*

Description

To plot spikes of a given neuron. Spikes of the neuron are represented using a givencolor.

Usage

```
spikes.plot.single(DataNeur, i, n, xrange, yrange, col='gray')
```

Arguments

DataNeur	A DataNeur e.g. the output of DNeur function - Contains the matrices of spike times as a global matrix
i	integer; index of the neuron to be represented. Less than or equal to n.
n	integer; total number of neurons.
xrange	Double; Numeric vector of length two to define the x-coordinate range in the plot.
yrange	Double; Numeric vector of length two to define the y-coordinate range in the plot
col	Character; To define color for plotting spikes. Default value is 'gray'.

Examples

```
# creation of spikes
lambda <- 50 #firing rate
a <- 0; b <- 2 # start and end recording times
ntrials <- 10
D = spikes.inject(2, lambda, a, b, ntrials)
xrange <- minmaxtimes(D)
yrange <- c(0,2*ntrials+2)  # number of trials
spikes.plot.single(D, 1, 2, xrange, yrange)
dev.new()
spikes.plot.single(D, 2, 2, xrange, yrange, col=c('gray','black'))
```

UE.plot *Function for plotting unitary events (coincident spikes and detected windows)*

Description

Function for plotting coincident spikes and detected windows

Usage

```
UE.plot(A, UE, xrange, yrange, col)
```

Arguments

A	Time window matrix. It should contain detected time windows after Benjamini-Hochberg procedure.
UE	List of Unitary Events. Contains fields N1 (first neuron) and N2 (second neuron). UE$N1 is a 3-by-ntrials matrix.
xrange	Double; Numeric vector of length two to define the x-coordinate range in the plot.
yrange	Double; Numeric vector of length two to define the y-coordinate range in the plot
col	Character vector; Vector of length two to define colors for plotting positive and negative Unitary Events. If only one input color is given, the same color is used for both types of UEs. Default value is c('red', 'blue').

See Also

draw_windows

Examples

```
# time window
#TW$A is a time window matrix
TW <- compute_time_windows(0.001, 0.901, 0.1, 0.1)

# creation of spikes
lambda <- 50 #firing rate
a <- 0; b <- 2 # times
p = lambda*(b-a)
trials <- 10; N <- 20
ntops <- rpois(N,p)
M <- matrix(data=0, nrow=N, ncol=max(ntops)+1)
M[,1] <- ntops
for (i in 1:N) # for each trial
{
r <- ntops[i] # depending on ntops
if (r!=0)
{ x <- runif(r,a,b) # uniform law
M[i,2:(r+1)] <- sort(x) # sorting
}
}
```

```
F1 <- M[1:trials,]
F2 <- M[(trials+1):(2*trials),]
UE <-list(N1=F1, N2=F2)

xrange <- c(0,max(F1[,-1],F2[,-1]))    #maximum spike value
yrange <- c(-50,2*nrow(F1)+2)  # number of trials

t <- UE.plot(TW$A, UE, xrange, yrange)
```

`UEdemo` *Demo of UnitEvents package*

Description

Demo of UnitEvents package. Examples of MTGAUE method and permutation are run.

Usage

```
UEdemo()
```

Value

A list of four UEs corresponding to each example

Examples

```
## Not run:
library(UnitEvents)
UEs = UEdemo()

## End(Not run)
```

`UnitEvents` *Unitary Events Method with Delayed Coincidence Count, with Gaussian Approximation (MTGAUE) or Permutation Method.*

Description

Unitary Events method with delayed coincidence count, with Gaussian Approximation (MTGAUE) or permutation method. Currently works for two neurons.

Usage

```
UnitEvents(DataNeur, TW, delay=0.02, level=0.05, Rtest="all",
             iperm=FALSE, B=10000, num_threads=1, statistical_value="T",
             DName="Neuron", rasterPlot=TRUE, export=FALSE,
             neurostatpath="")
```

Arguments

DataNeur	A DataNeur e.g. the output of DNeur function. Matrix of spike times, of size nneurons*ntrials-by-(nmax+1), where nneurons is the number of neurons, ntrials is the number of trials and nmax is the maximum number of spikes for every neuron and trial. For a missing value, 0 is used to fill the matrix. The first column of the matrix is the number of spikes for each trial.
TW	List defining time windows, TW contains the following fields • A: Matrix of time windows with two rows. Each column is a time window. The number of columns of A is equal to the number of time windows. • L: Time step or spacing between the starts of two consecutive time windows. • len: length or number of time windows
delay	double; delay is the value used in coincidence count.
level	double; level is the alpha value used in Benjamin-Hochberg procedure.
Rtest	string; Rtest allows to define the type of the test. Only possible values are "all", "symmetric", "upper", "lower". Default value is "all" which means that upper and lower tests are performed and discernible.
iperm	Logical to select Permutation method or not. Default is FALSE: MTGAUE method, which performs more rapidly, is chosen but can be dubious if no stationarity. Permutation method takes more time but is more robust.
B	integer; size of Monte Carlo sample for Permutation method. Default is 10000.
num_threads	integer; Number of threads for neuro-stat computation. Default is 1.
statistical_value	string; Only possible values are "T" or "H". Default is "H". Corresponds to a centered computation or not.
DName	string; title of the plot. Default is "Neuron".
rasterPlot	bool; To display spikes in a window. Default is TRUE.
export	bool; To export the raster to png format. Default is FALSE.
neurostatpath	string; It allows to define the path of neuro-stat code if necessary. It could be a relative or an absolute path. Default value is empty which means that default path is used (for example '~/R/x86_64-pc-linux-gnu-library/3.2' on Linux)

Value

The output is a two-field list containing

- A: list of all detected time windows as a three-row matrix (the last row contains the detection sign)
- UE: list of unitary events as a two-row matrix with spike time and trial number for each unitary event

References

Multiple Tests based on a Gaussian Approximation of the Unitary Events method with delayed coincidence count. Tuleau-Malot C., Rouis A., Grammont F. and Reynaud-Bouret P., *Neural Computation*, **26**(7), pp1408–1454, 2014.

Surrogate Data Methods Based on a Shuffling of the Trials for Synchrony Detection: the Centering Issue. Albert M., Bouret Y., Fromont M., and Reynaud-Bouret P., *Neural Computation*, **28**(11), pp2352-–2392, 2016.

See Also

DNeur, MulTestsBH

Examples

```
rm1standlast7col <- function(v) { return(v[-c(1,(length(v)-6):length(v))])}
scaling <- 1/10000
DataNeur = DNeur(list("UnitEvents/data/Neur1_c13.txt",
"UnitEvents/data/Neur2_c13.txt"),
                listOptions=list('removeCols'=rm1standlast7col,'
                scaling'=c(scaling)))
res = UnitEvents(DataNeur = DataNeur,
TW = compute_time_windows(a=1e-3, b=2.1, 1e-1, 0.05))
```

---

use_options *To modify the DataNeur depending on options.*

Description

To modify the DataNeur depending on options.

Usage

```
use_options(M, listOptions=list())
```

Arguments

M	matrix; input matrix containing the spike times of a neuron without spike count.
listOptions	list; List of options to define the DataNeur. May contain a scaling factor or a function to remove columns. Default value is the empty list.

Value

The output is the modified input matrix.

Examples

```
scaling <- 10
M <- matrix(c(0.5, 0.51, 1.1, 1.09), nrow=2)
M <- use_options(M, list(scaling=scaling))
print(M)
```

Index