TwoPhoCoTo:

The best two-photon correlation toolbox in the world!
(and also the only one)

July 27th 2012
Version 1.2
By Jorrit Montijn

TwoPhoCoTo: Two-photon correlation toolbox
Documentation

Table of Contents

Preface

Congratulations! You see before you the documentation on TwoPhoCoTo: The best two-photon correlation toolbox in the world! (and also the only one).

1. Introduction

What does TwoPhoCoTo do?

Well, the TwoPhoCoTo can help you analyze correlations in neuronal activity that have been measured using two-photon imaging. Listed on the following pages are the functions you can use. You can perform the analysis on either the whole batch of cells, or you can perform the analysis for different cluster sizes.

Please note that this is work in progress, and changes are made from time-to-time. It is pretty certain that some functions will no longer work and need to be updated to conform to a new format. If this is the case, and you wish to use that function: please contact me.

TwoPhoCoTo: Two-photon correlation toolbox

Function Index

For more information on any function, just type “help functionName” at the Matlab prompt and you will receive additional information on the function. These tables are just to provide a short list of which functions are included in the package and to give an impression of what you should be able to do with them. The most important is *processActivityMatrix()*, which calculates pairwise correlations for all neurons in your recording. The high-level functions are *runAnalysis()*, *runClusterAnalysis()*, *runKMeansClustering()* and *runSlidingWindowAnalysis()*. If you want to really understand what it is that’s happening, check out *retrieveClusterCorr()* for the clustering analysis and *processActivityMatrix()* for the pairwise correlation calculations. They should be pretty well documented. Note that in order to do anything; you first need to have properly formatted raw data. The raw data should be formatted in .mat file where *ses.neuron(neuronIndex).dFoF* refers to the raw activity data. If you want to change the scripts to work with your data as well, you can always ask me. If I’m not too busy, I’ll try to help;-). In case you forgot my e-mail address, it’s jorritmontijn@gmail.com.

The group letters correspond to the following general categories:

Group Letter	**Function**
A	Awake/Anesth vs. Distance and interaction analysis
B	Complete cluster analysis of all possible clusters of all possible cluster sizes
C	K-means clustering comparing within/between cluster differences in correlation and distance
D	Sliding window analysis for time-course study of the input signal and correlation
S	Supplementary functions not belonging to group A-D
G	Functions required by multiple groups; including core functions

High-level functions (group masters)

Name	**Syntax**	**Purpose in short**	**Version Group**
runAnalysis	runAnalysis([boolDoTransforms=0])	Uses compareAwakeAnesth with spiking data and fluorescence data	**1.0 [A]**
runClusterAnalysis	runClusterAnalysis([boolDoTransforms=false])	Runs a cluster-based analysis with the *n* best cells for every cluster size	**1.0 [B]**
runKMeansClustering	runKMeansClustering(strDir,[boolUseSpikes=false],[plotVec=1:10],... [dblReqBinDuration=0.5], [vecEpochStart=10],... [dblEpochDuration=190],[intIncludeCells=0],[vecClusters=2]... [startAt=1],[stopAt=inf])	Uses k-means clustering to compare within/between cluster differences	**1.0 [C]**
runSlidingWindowAnalysis	runSlidingWindowAnalysis(strDir,[strOutput='slidingWindowData'],... [boolMakeGraph=false],[boolOneSessionPerMouse=false],... [intAnesthStart=1],[intAnesthStop=inf],[intAwakeStart=1]... [intAwakeStop=inf],[boolUseSpikes=false],[dblBinDur=0.5],... [intWindowBins=10],[intStartEpochAt=10],[intStopAtBin=260])	Uses sliding window analysis for correlation/time-course plots	**1.0 [D]**

High-level functions

Name	**Syntax**	**Purpose in short**	**Version [Group]**
compareAwakeAnesth	compareAwakeAnesth(strDir,[boolUseSpikes=0])	Compares awake vs. anesthetized correlation on distance dependency	**1.1 [A]**
runAnesthCluster	[matCorrAnesth,matDistancesAnesth] = ... runAnesthCluster(strDir,vecClusterSize,intIncludeCells)	Runs a cluster-based analysis for all anesthetized sessions	**1.0 [B]**
runAwakeCluster	[matCorrAwake,matDistancesAwake] = ... runAwakeCluster(strDir,vecClusterSize,intIncludeCells)	Runs a cluster-based analysis for all awake sessions	**1.0 [B]**

High-level plotting functions

Name	**Syntax**	**Purpose in short**	**Version [Group]**
plotClusterDistance	plotClusterDistance(matClusterDistances)	Used to plot the distance output from clusterCalcSession()	**1.0 [B]**
plotClusters	plotClusters(matClusterData)	Used to plot the correlation output from clusterCalcSession()	**1.0 [B]**
runClusterPlotting	runClusterPlotting(strDir,strAnesthClusterOutputFile,... strAwakeClusterOutputFile,[plotVec=5:8])	Runs analysis on the output files from runClusterAnalysis()	**1.0 [B]**

Supplementary high-level functions

Name	**Syntax**	**Purpose in short**	**Version [Group]**
testDFOFCorr	[vecCorr vecStd] = testDFOFCorr (vecNoise,intReps,intLength)	Test-runs fluorescence dummy data to check correlation dependencies	**1.0 [S]**
testSpikingCorr	[vecCorr vecStd] = testSpikingCorr(vecMeanRates,intReps,intLength)	Test-runs spiking dummy data to check correlation dependencies	**1.0 [S]**

Mid-level functions

Name	**Syntax**	**Purpose in short**	**Version [Group]**
calcCorr	dblCorrelation = calcCorr(vecIn1,vecIn2)	Outputs the correlation between the two input vectors	**1.0 [G]**
clusterCalcSession	[matOut,matDist] = … clusterCalcSession(strDir,strSession,vecClusterSize,… [boolUseSpikes=1],[intIncludeCells=0],[boolDoPlotting=1])	Calculates correlation and distance for a cluster sizes for one session	**1.1 [B]**
makeSlidingWindow	[vecMeanCorr vecMeanActivity] = makeSlidingWindow(strDir,strSession,… [boolUseSpikes=false],[dblBinDur=0.5], [intWindowBins=10],… [intStartEpochAt=10],[intStopAtBin=260],[boolMakeGraph=true])	Makes a sliding window timeseries of a single session	**1.0 [D]**
movieViewer	[none], it's a GUI	Load pictures/movies to edit and export	**1.1 [G]**

Low-level functions

Name	**Syntax**	**Purpose in short**	**Version [Group]**
makeDFOFData	[vecData] = makeDFOFData(intLength,dblNoise)	Creates dummy fluorescence data	**1.0 [S]**
makeSessionIndex	[vecSessionIndex cellSessionIndex] = makeSessionIndex(cellSessions)	Transforms the input cell-array so that sessions are indexed and can be accessed more easily	**1.0 [G]**
makeSpikingData	[vecData] = makeSpikingData(intLength,dblMeanSpikingRate)	Creates dummy spiking data	**1.0 [S]**
processActivityMatrix	[matCorrData intCells matDistances matRawCorr dblActEpochDuration matEpochOut vecStart vecStop intMax dblAnesth matRawCorrOverall dblActBinDuration] = ... processActivityMatrix(strDir,strSession,boolUseSpikes,... [dblReqBinDuration=0.5],[vecEpochStart=10],[dblEpochDuration=190],... [intIncludeCells=0],[boolDisplayWarnings=0])	Outputs correlation matrices and session information of a single pre-processed session	**1.5 [G]**
retrieveClusterCorr	[nVec,vecDist,vecDistWeight] = retrieveClusterCorr(... xVec,xVecDist,matRawCorr,matDistances,intClusterSize)	Retrieves the mean correlation per cluster for a certain cluster size	**0.9.3 [B]**
transformSessionToMouse	vecOut = transformSessionToMouse(vecIndex,vecData)	Takes an index vector and data vector and outputs a data vector with a single data point per mouse	**1.0 [G]**
calc_dFoF	[vecdFoF] = calc_dFoF(F, samplingFreq, BaselineWindowSize)	Calculate dFoF	**1.0 [G]**
calc_expFit	[vecExpFit vecSpikeFrames vecSpikeAmount] = calc_ExpFit(F,samplingFreq,tau)	Calculate spikes	**1.0 [G]**
smooth1D	[sData, kernel] = smooth1D(data, kernelSize, kernelMethod)	Required for dFoF calculation	**1.0 [G]**

General functions

Name	**Syntax**	**Purpose in short**	**Version [Group]**
compressMatrix	matBinned = compressMatrix(matRaw,intBins,['binsize'/'binnumber'])	Compresses a 1D or 2D matrix by its 2nd dimension	**1.0 [G]**
defaultValues	[variables] = defaultValues(varargin,defaultvalues)	Outputs the supplied argument if any and otherwise your default values	**1.0 [G]**
findmax	[C,I] = findmax(A,maxnum)	Finds the highest n values from a vector	**1.0 [G]**
findmin	[C,I] = findmin(A,minnum)	Finds the lowest n values from a vector	**1.0 [G]**
linearFunction	y = linearFunction(x,beta)	y = beta(1)*x+beta(2)	**1.0 [G]**
makeBins	[nVec,meanVec,stdVec] = makeBins(xVec,yVec,binVector)	Bins a 1D or 2D matrix by its 2nd dimension	**1.0 [G]**
mat2index	[valueVector,yVector,xVector] = mat2index(matrix)	Transforms a matrix into a triple of indexed vectors; so that *v(i)* corresponds the original value of the input matrix at y-location *y(i)* and x-location *x(i)*	**1.0 [G]**
MLFit	[paramBest, mse] = mlfit(@fn, params0, xvals, yvals, stdev)	Maximum likelihood fitting procedure written by Pascal Mamassian	**8-7-2009 [G]**
runBootstrap	[matParams] = runBootstrap(xList,matData,paramInit,… functionHandle,[maxIter=1000])	Runs a bootstrap procedure	**1.0 [G]**
zScale	matOut = zScale(matIn)	Returns a z-scaled matrix calculated per row (1D or 2D input)	**1.0 [G]**