**RippleDetector**

The ripple detector class includes several functions:

1. Detects ripples in the data based on the method described in Staresina et al 2015.
2. Analyzes typical ripple behavior (average, spectrum, correlation with spindles) in a given channel.
3. Analyzes correlation between spikes activity (in single or multi units) and ripples (detected on macro or micro channels).

**A. Detection:**

detectRipples

The method receives as input: the data, and a vector of sleep scoring (optional), IIS times (optional). The sleep scoring is a vector in which each element represents the sleep stage for an epoch of the duration stored in the property sleepEpochs (e.g. 1 second). The method will detect ripples only in the sleep stages specified by the property sleepEpochs (in Staresina et al: they detect only in stages 2-4 of the NREM). IIS times is a vector with the times of interictal spikes (for example the output of SpikeWaveDetector).

The method performs the following steps:

* Filters the data using a bandpass filter in the range specified by the properties minFreq and maxFreq (by default, as in Staresina et al, 80-100 Hz).
* Changes all the points in the data which do not belong to the required sleep stages (which are stored in the property sleepEpochs) to NaN.
* Removes IIS – turns to NaN windowAroundIIS ms before and after each IIS (by default 500 ms).
* Calculates the root mean square of the filtered data over a moving window of RMSWindowDuration milliseconds (by default, 20 ms).
* Calculates a threshold which is the rippleThreshPercentile of the calculated r.m.s. signal (by default, the 99th percentile).
* Finds segments of at least minDurationAboveThresh milliseconds which pass the threshold (by default 38 ms). These are the proposed ripples.
* Goes over the proposed ripples, for each proposed ripple:
  + Check whether more than minPercZerosAllowed (0.1 by default) of the ripple segment is zero, in which case it is excluded (note: this is an addition to Staresina’s method I added to deal with the segments of zeros in our data).
  + Smooths the original raw data in the proposed ripple segment using an averaging filter of 3 consecutive points.
  + Finds local minima and maxima in the smoothed segment, a ripple is verified only if there are more than minNumOfExtreme minima or minNumOfExtreme maxima.
  + The index of the ripple is set at its highest peak.

getRipplesFromMicroElectrodesInArea

The ripples detection on micro channels is performed per area – i.e. the detection takes into account all the channels in that area. In order to do that:

A. Ripples should be detected first for all single micro channels (by using detectRipples).

B. The method getRipplesFromMicroElectrodesInArea receives a cell array of ripple detections for the channels in a given area (i.e. 8 channels if the area has 8 micro channels) and a cell array of start-end indices per ripple.

C. Ripple will be considered “legitimate” if it appears in at least two channels in the area. Ripple in channel X and ripple in channel Y will be considered the same ripple (and thus appear in both areas) if they are less than ripplesDistMicrChanForMerge ms apart (by default – 15 ms).

The method returns the merged ripple times and ripple start-end times.

saveRipplesDetectionsMicro – A wrapper for running ripples detection on micro channels. Ripple detection on micro channels is performed per area, where a ripple will be considered “legitimate” if it appears in at least two channels in the area. Ripple in channel X and ripple in channel Y will be considered the same ripple (and thus appear in both areas) if they are less than ripplesDistMicrChanForMerge ms apart (by default – 15 ms).

By default, the wrapper detects and saves ripples per area, which means: A. Running and saving ripples for all of the micro channels in that area, B. Running the ripples merge method (getRipplesFromMicroElectrodesInArea) and saving the merge ripples for the area.

The list of areas on which the wrapper will run on is provided as part of the runData input struct, if left empty then the wrapper will run on all the areas for that patient (based on the micro montage).

Another option is to run ripple detection per micro channels, without the merging per area step. This is possible if the input parameter runByChannel is set to true (by default it’s false) and the list of channels to run on is provided as part of the runData input struct.

By default when requested to run on an area the method will first run the detections on all the channels in that area and save them. If useExistingRipples is true, then before running the detection on a micro channel it will first check whether a ripples file for that channel already exists and if it does will load it instead of running the detections – this is useful if the detections for some or all of the micro channels in the area were already run and saved and we only want to run the missing ones + the merging step or only want to run the merging step if all of them were already saved.

The wrapper receives as input runData which is a struct array with the length as the number of patients. Each element (=patient) should have the fields:

PatientName

areasToRunOn – a list of areas on which the ripples detection per channel and per area will be run. If left empty the method will run on all the areas for that patient. If the input parameter runByChannel is true then this field is ignored and the run is by channel and not by area (by default it’s false).

channelsToRunOn – a list of channels for which the ripples detection will be performed and saved. This field is only relevant if the input parameter runByChannel is true (by default, false), otherwise this field is ignored.

microMontageFileName – the file name (including path) of the micromontage.

MicroDataFolder – The folder in which the raw micro data is stored. The property fileNamePrefix of the class includes the prefix for the data filenames (by default: ‘CSC’).

microRipplesFileNames – name (including path) of the ripple mat files in which the ripple times for the micro channels should be saved. The method will save ripples per channel as microRipplesFileNames<#channel index> and ripples per area as microRipplesFileNames<area name>.

MicroSpikesFileNames – The filenames (including path) of the mat files that include the spike times. The method assumes the filename format is SpikesFileNames<#area\_name>. That means that for the detection of all the ripples on micro channels in the RAH area (for example) the same spikes file is loaded with the name <MicroSpikesFileNames>RAH.mat.

sleepScoringFileName – file name (including path) of the sleep scoring mat file. If not provided all the data will be used.

plotRipples

Plot the single ripples. Receives as input the data and the ripple times. The plots contain both the original raw signal and the filtered signal, the ripple time is exactly at the middle of each plot (by default 0.5 second is presented before and after, so the ripple index is at point 500 ms. The presented window duration can be set by the property secondBefAfter). The size of the subplots is set in the properties subplotSizeX and subplotSizeY.

If the parameter folderToSave is provided the method will save the plots into the folder (and not present them on screen).

PlotRipplesBiPolar

The method runs ripple detection on channels with bipolar referencing and saves figures with the results. The method goes over the required channels, references them to the provided reference, runs ripple detection on the referenced data, and plots single ripples into the required folder. Note: the spike times which are taken into account when detecting (i.e. – no ripples will be detected in the vicinity of a spike) are the unity of spikes from both channels.

Input:

runData –

A struct array with the length as the number of patients. Each element (=patient) should have the fields:

biPolarCouples – a matrix with the size <# of chans>\*2. Element <i, 1> includes the channel index on which we want to detect ripples, element <i, 2> includes the channel to which it is referenced. i.e. the detection is run on data\_<i,1> - data\_<i, 2>.

DataFolder – The folder in which the raw data is stored. The property fileNamePrefix of the class includes the prefix for the data filenames (by default: ‘CSC’).

SpikesFileNames – The filenames (including path) of the mat files that include the spike times. The method assumes the filename format is SpikesFileNames<#chan index>.

folderToSave – folder name into which to save the plots of the detected ripples. The method uses plotRipples to create the figures and the figures’ properties are set by the same properties as the ones setting plotRipples’ output.

PlotRipplesMicro

The method plots ripples detected on micro channels. Ripple for single micro channels should be detected and saved in advance. The method loads ripples for single micro channels in the area, merges them and plots all the single ripples in figures where each column is a channel and each row is a ripple. Red circles appear in all the channels for which a ripple was detected – i.e. each row should have at least two red circles as only ripples that are detected in at least two channels are considered legitimate.

It receives as input:

A. runData – a struct containing the fields:

patientName

microMontageFileName – the file name (including path) of the micromontage.

MicroDataFolder – folder from which to read the micro data (the method assumes the prefix for the files is CSC, can be changed by the property dataFilePrefix)

MicroRipplesFileNames – name (including path) of the ripple mat files in which the ripple times for the micro channels are saved (the method assumes the name of the file is microRipplesFileNames<#channel index>

B. areaName – name of the area (as appears in the micro montage) to plot the ripples for.

C. folderToSave (optional) – folder into which to save the figures.

Comparison Between Ripple Detection Criteria in Various Publications:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Paper** | **Bandpass range** | **Data on which detection was performed** | **Ripple criterion** | **Additional Conditions** | **Electrode Referencing** |
| Staresina, Nature Neuroscience 2015 | 80-100 Hz | Artifact-free NREM stages 2-4 | >99% percentile of average of 20 ms of RMS, passed for >= 38 ms | At least 3 peaks or 3 troughs in the raw signal | Mastoids. Analysis was performed also after re-referencing the hippocampal data to another contact on the same depth electrode with the same pattern of results. |
| Staba, American Neurological Association, 2004 | 80-500 Hz (detects ripples and fast ripples together) | Wake+NREM+REM | Successive RMS amplitude greater than 5 stds of the mean (it’s not mentioned what defines “successive”) |  | Contra-lateral auricule site |
| Zhang, Nature Communications 2018 | 80-100 Hz (note: they tried also 80-140 Hz and found 95% of the ripples were in the 80-100 Hz range) | Artifact free data (IIS – removed using manual and automatic detection) | >99% percentile of average of 20 ms of RMS, passed for >= 38 ms | At least 3 peaks or 3 troughs in the raw signal | Mastoids, re-reference before analysis to the average across all depth and subdural electrodes |
| Vaz, Science 2019 | * 1. Hz | Data after removal of IIS | >2 stds of the amplitude of the Hilbert transform, for at least 25 ms, with max amplitude >3 stds | Adjacent ripples less than 15 ms apart were merged | Local – bipolar referencing, difference between pairs of adjacent electrodes |

**B. “Ripple behavior overview” per channel**

runRipplesData

The method produces information about the ripples in a channel that includes:

A. Average ripple – before stimulation and during stimulations (short effect).

B. Spectrum of average ripple – before stimulation and during stimulations (short effect).

C. Average of TFR around ripples – before stimulation and during stimulations (short effect).

D. Average of TFR around spindles – before stimulation.

E. Polar histogram of synchronization index between spindles and ripples range.

The input runData is a struct in the length of number of patients (for which the analysis is required). In addition it receives the input parameter fileNameResults which includes the file name into which the results will be saved (optional).

Each element (=patient) in runData should include the fields:

patientName

channelsToRunOn – list of channel indices for which to perform the analysis.

DataFolder – The folder in which the raw data files are saved (the method assumes the prefix for the files is CSC, can be changed by the property dataFilePrefix).

macroMontageFileName - the file name (including path) of the macromontage.

RipplesFileNames - name (including path) of the ripple mat files in which the ripple times for the macro channels are saved (the method assumes the name of the file is RipplesFileNames <#channel index>

SpindlesFileNames - name (including path) of the spindle mat files in which the spindle times for the macro channels are saved (the method assumes the name of the file is SpindlesFileNames <#channel index>

SpikesFileNames - name (including path) of the spikes mat files in which the spikes times for the macro channels are saved (the method assumes the name of the file is SpikesFileNames <#channel index>). If not provided spikes will not be removed from the data for the analysis.

sleepScoringFileName – file name (including path) of the sleep scoring mat file. If not provided all the data will be used.

The output struct ‘results’ includes all the results of the analysis, which can then be plotted using plotResultsRipplesData. The output struct is a struct with the length of the number of patients (=the length of runData), where each element includes:

patientName

resultsPerChan – a struct in the length of the number of channels required for the analysis per the patient. Each element in resultsPerChan includes the fields:

channelNum

area

nRipplesBefore, nRipplesStim - number of ripples before stimulations and after stimulations (short+mid effect together) respectively

avgBefore, avgStim – average ripples before stimulations and after stimulations (short+mid effect) respectively

stdBefore, stdStim – std of ripples before stimulations and after stimulations (short+mid effect) respectively

specBefore, specStim – spectrum of ripples average before stimulations and after stimulations (short+mid effect) respectively

meanTFRRipBefore, meanTFRRipStim – mean of ripple triggered TFR before stimulations and after stimulations (short+mid effect) respectively

SIanglesSpRip – Synchronization Indices of spindles-ripples before stimaulations (an array with the length as number of spindles)

R – results of the r-test for the polar histogram (of SIanglesSpRip)

V – results of the v-test for the polar histogram (of SIanglesSpRip)

meanSpecs – mean spindle-triggered TFR before stimulations.

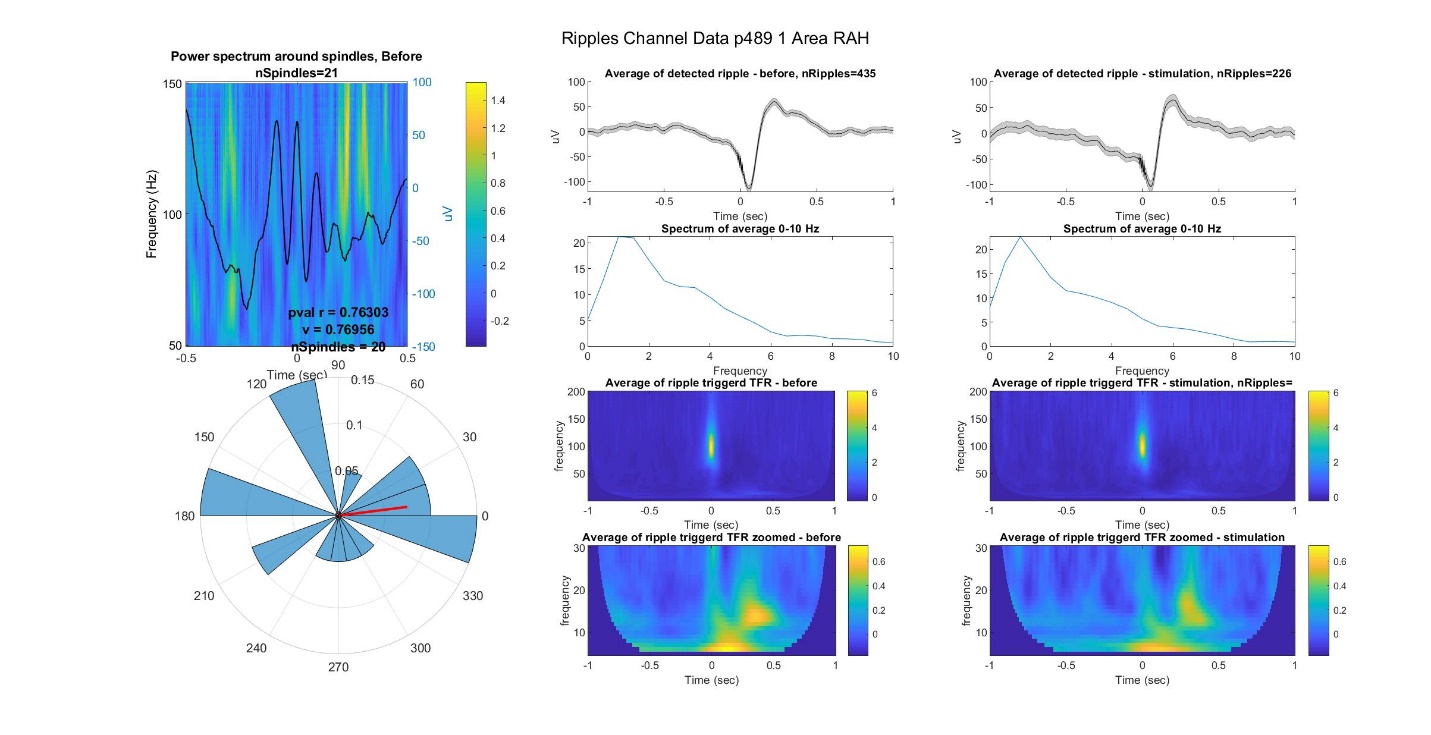
meanEpochs – mean spindle for the spindles for which meanSpecs were calculated.

nEpochs – number of spindles in the spindle-ripple analyses (all before stimatulions).

plotResultsRipplesData

The method receives as input a struct with the format of the output of runRipplesData and produces figures. The input folderToSave (optional) sets the folder into which the figures will be saved.

The figures produced are per channel and have the following format:



**C. Correlation between spikes activity and ripples**

runRipSpikes

The method produces information about correlation between ripples and spikes.

The analysis is performed per requested macro channel and examines the relationship between ripples in that macro channel and spikes in all the micro channels in the same area (based on the macro and micro montages). Analysis is performed for the macro channel vs spike rate in each single/multi unit in the area and also vs the average spike rate of the single/multi units. The property spikeMultiUnits sets whether the analysis is on single or multi units (by default it’s set to true – i.e. multi units).

The main result is the firing rate in a time window around ripples and around controls for ripples before the stimulations and for ripples during the stimulations (short+mid effect). The time window is set by windowSpikeRateAroundRip (by default 500 ms before and after the ripple) and the time window for which the p-value comparing the two states is calculated is set by windowForSignificance (by default 100ms before and after the ripple). The controls are random points which are: a. at least minDistControl (by default 300ms) before or after the ripple, b. no more than maxDistControl (by default 1300ms) before or after the ripple, c. do not coincide with a different ripple.

Single/multi units in which the spike rate is below minSpikeRateToIncludeUnit (by default 1Hz) are not included in the analysis.

The method also returns stimulation triggered spike rate and the spike rate through the entire session (before and during stimulations). The stimulation triggered spike rate is presented vs a control, which is at the time point controlDistForStim (by default 1000ms) before the stimulation.

The spike rates are calculated by using movsum with a window size firingRateWinSize (by default 100 ms).

The input runData is a struct in the length of number of patients (for which the analysis is required). In addition it receives the input parameter fileNameResults which includes the file name into which the results will be saved (optional).

Each element (=patient) in runData should include the fields:

patientName

channelsToRunOn – list of channel indices for which to perform the analysis.

ExpDataFileName – name (including path) of the EXP\_DATA for the patient.

spikeData – name (including path) of the file which includes the spike data for the patient (i.e. the file which usually has the name <patientName>\_spike\_timestamps\_post\_processing.mat).

macroMontageFileName - the file name (including path) of the macromontage.

RipplesFileNames - name (including path) of the ripple mat files in which the ripple times for the macro channels are saved (the method assumes the name of the file is RipplesFileNames <#channel index>

The output struct results includes all the results of the analysis, which can then be plotted using plotResultsSpikes. The output struct is a struct with the length of the number of patients (=the length of runData), where each element includes:

patientName

resultsPerChan – a struct in the length of the number of channels required for the analysis per the patient. Each element in resultsPerChan includes the fields:

channelNum

area

unitInds – a cell array with the single/multi unit indices for this area. If using multi units each element in the cell array is an array including all the single units per multi unit.

spikeTimes – a cell array with the length as the number of units where each element is the spike times for that unit.

fireRateRipBefore, fireRateRipStim – cell array with the length as the number of units where each element is the average spike rate around ripples for that unit for all the ripples before the stimulation and during the stimulations (short+mid effect together) respectively.

fireRateRipAvgUnitsBefore, fireRateRipAvgUnitsStim – the average spike rate around ripples averaged over all units for all the ripples before the stimulation and during the stimulations (short+mid effect together) respectively.

fireRateControlBefore, fireRateControlStim - cell array with the length as the number of units where each element is the average spike rate around controls for that unit for all the ripples before the stimulation and during the stimulations (short+mid effect together) respectively.

fireRateControlAvgUnitsBefore, fireRateControlAvgUnitsStim - the average spike rate around controls for all the ripples averaged over all units before the stimulation and during the stimulations (short+mid effect together) respectively.

allSessionFireRates - cell array with the length as the number of units where each element is spike rate across the session (before and during stimulations).

stimTriggeredFireRates - cell array with the length as the number of units where each element is a matrix where each row is the spike rate function around a stimulation.

controlForStimTriggered - cell array with the length as the number of units where each element is a matrix where each row is the spike rate function around the control per stimulation.

firstStim – time of first stimulation.

lastStim – time of last stimulation.

The method utilizes in its calculations the methods:

checkSpikeRateAtRip – a method which receives the ripple and spike times and returns the spike rate around ripples and around controls

The input is:

rippleTimes – array of ripple times

rippleStartEnd – matrix where each row is a ripple and index <I,1> is the start point of ripple I and index <I,2> is the end point of ripple i.

spikeTimes – array of spike times, can also be a cell array of arrays of spike times, in which case the analysis will be performed on the average spike rate.

dataDuration (optional) – the length of the session in ms.

The output is:

fireRateRip – matrix with the number of rows and the number of ripples where each row is the average spike rate around the ripples.

fireRateControl - matrix with the number of rows and the number of ripples where each row is the average spike rate around the controls.

getSpikeRateAtStim – receives the spike times and the stimulation times and calculates spike rates around stimulations and around control, and the spike rate through the entire session.

The controls for stimulations are the time points controlDistForStim (by default 1000ms) before the stimulations.

The spike rates are calculated by using movsum with a window size firingRateWinSize (by default 100 ms).

Input:

stimTimes – an array of stimulation times

spikeTimes – an array of spike times

dataDuration (optional) – length of session (ms)

output:

spikeRateSession – the spike rate function through the entire session

rateAroundStim – a matrix where each row is the spike rate around a stimulation

rateAroundControl – a matrix where each row is the spike rate around the control

runRipSpikesMicro – This method is the same as runRipSpikes except it uses ripple times as detected on micro channels, and not macro channels. Ripple detection on micro channels is performed for an entire area rather than just one channel – see the explanation on getRipplesFromMicroElectrodesInArea for more details. Because the detection is per area and not per channel, the difference from runRipSpikes is that in the struct runData instead of a field channelsToRunOn there should be a field ‘areasToRunOn’. If this field doesn’t exist or is empty the method will try to run on all areas for that patient according to the micro montages.

There are two more differences in the input runData struct, it should have the fields:

microMontageFileName - the file name (including path) of the micromontage (this is instead of macroMontageFileName).

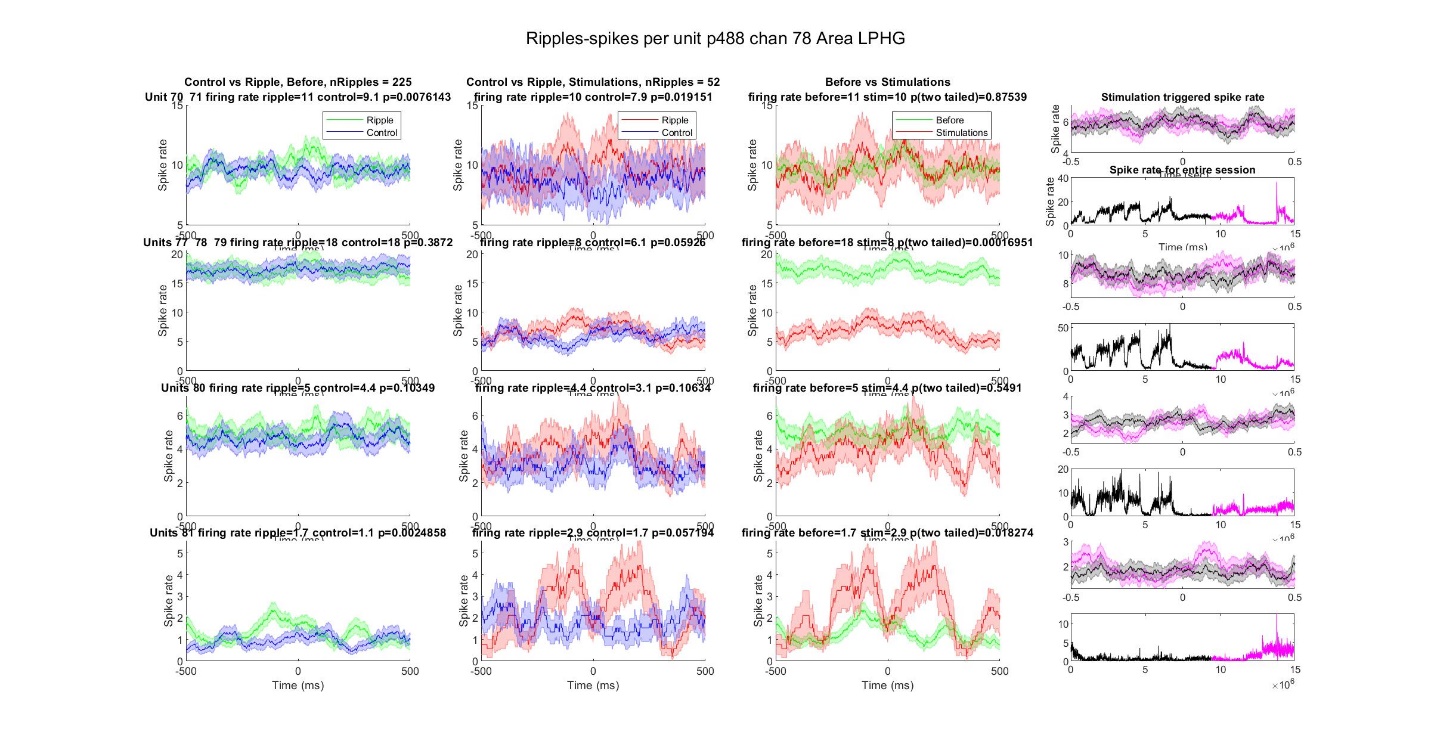
MicroRipplesFileNames - name (including path) of the ripple mat files in which the ripple times for the micro channels are saved (the method assumes the name of the file is MicroRipplesFileNames <#channel index>). This is instead of RipplesFileNames.

The output of runRipSpikes can be plotted using:

plotResultsSpikes – which receives the ‘results’ struct (the output of runRipSpikes) and folderToSave (optional) for saving the figures.

The figures per channel show the spike rate for ripple vs control before and during stimulations per single/multi unit and for the average of units. They also show the stimulation triggered spike rate per unit and the spike rate for the entire session per unit. This method also calculates the p-value of the comparison between the presented conditions (ripple vs control or before vs stimulations), using the average over the event-triggered window, the window size is defined by windowForSignificance (by default 100ms before and after the ripple).

Example of figure per unit (note: the maximal number of units shown in the single units figure is set by maxLinesInFigureRipSpike and by default is 4). The purple in the spike rate figures represent stimulation spochs.



Example for the average over units:

