UltraMegaSort2000 Manual

By

Daniel N. Hill
Samar B. Mehta
David Kleinfeld

February 27, 2011

Please send questions or bug reports to **ultramegasort2000@gmail.com**.

Table of Contents

1 Introduction to UltraMegaSort2000

Congratulations on downloading **UltraMegaSort2000**! You are about to experience the world's finest software package for spike sorting, featuring:

- Fully automated initial spike sorting
- An efficient implementation that allows for data sets containing many 10s of thousands of spike events
- Easy manual correction tools implemented in a sophisticated GUI
- Flexible data structures that allow for multi-electrode, multi-trial data sets
- Quality metrics to verify the completeness and purity of your spike trains
- The convenience and customizability of the MATLAB analysis environment

To get the most out of your UltraMegaSort2000 experience, please read the following manual carefully.

This software contains MATLAB code for spike sorting of extracellular neurophysiological data. The code performs automatic detection and sorting of putative single-unit spike trains from filtered data. The requirements on the data set are very general. The data can be taken with multi-channel electrodes and include multiple trials. This package also contains tools for the manual inspection and correction of automated spike sorting. The user can manually manipulate spike clusters by merging or splitting them and removing outliers. Finally, there is also included a set of quality metrics that allow the user to quantify the contamination and completeness of a cluster of spikes.

The spike sorting process has 4 major steps, and this manual presents these steps in this order:

(1) Format data set
(2) Automated detection of spike waveforms and clustering
(3) Manual inspection and correction
(4) Quality metrics

As a quick reference, this package also includes a file called **demo_script**. This script contains example code for a spike sorting session, a list of all the major visualization functions, and code to generate a simulated data set.

This software was written in the Kleinfeld lab at the University of California, San Diego. For a review of spike sorting, see (Lewicki 1998).

2 Automatic spike sorting

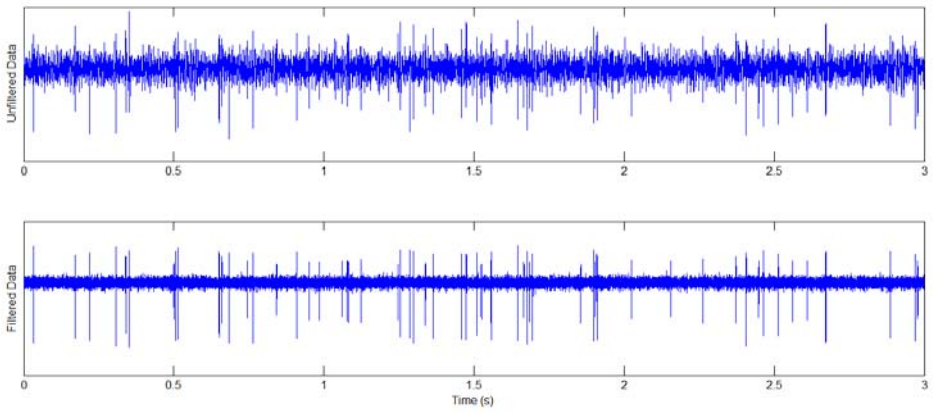
In this software, the spike sorting process begins with an automated algorithm that turns filtered extracellular data into sorted spikes. A typical spike sorting session is run with the following block of code:

```
spikes = ss_default_params(Fs);
spikes = ss_detect(data,spikes);
spikes = ss_align(spikes);
spikes = ss_kmeans(spikes);
spikes = ss_energy(spikes);
spikes = ss_aggregate(spikes);
splitmerge_tool(spikes)
```

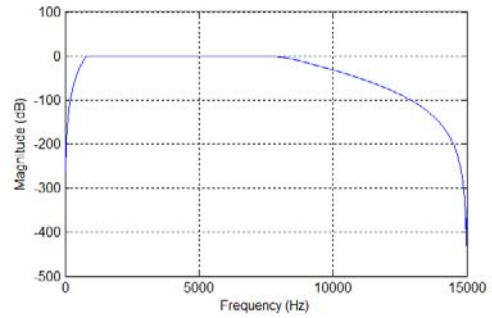
where **Fs** is the sampling rate in Hz, **data** is a data block containing the filtered extracellular data, and the last line initiates the manual inspection of the sorted spikes. The **spikes** object contains all the parameters, spike data, cluster assignments, and processing information for the spike sorting session. A full reference on the fields of this structure are given in section 4. This section explains how to format your **data** and what the sorting functions do. For reference, the automated algorithm is based largely on the method of (Fee 1996b).

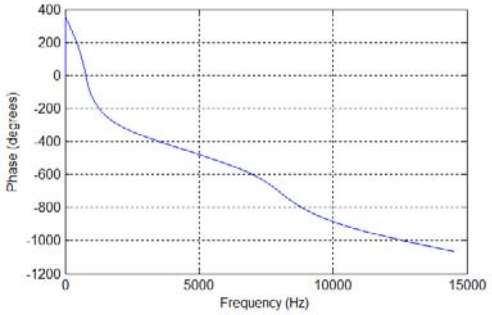
2.1 Filtering

This software assumes that extracellular data has already been properly filtered, but here we include a few tips on sorting data. The goal of filtering is to increase the signal-to-noise ratio of your data without distorting spike waveforms.



This is most easily described in the frequency domain where the specification for the filter is that it is zero-phase and removes frequency components where the spike waveform has little or no spectral power. As spike waveforms are typically no longer than 1.5 ms, any signal below 700 Hz can be removed safely. On the high frequency end, noise tends to dominate signal around 8000 Hz [Ref]. Show below is the frequency and phase response for a Butterworth filter with these cut-offs.





Note that the phase response is non-zero and non-linear. This can be remedied by running the filter both forwards and backwards using a Matlab function called **filtfilt**. It should be noted that running the filter in both directions means that the gain of the function is squared. The following fragment of code implements this filter:

```
Wp = [ 700  8000] * 2 / Fs;       % pass band for filtering
Ws = [ 500 10000] * 2 / Fs;       % transition zone
[N,Wn] = buttord( Wp, Ws, 3, 20); % determine filter parameters
[B,A] = butter(N,Wn);             % builds filter
data = filtfilt( B, A, data );    % runs filter
```

The 2nd line of code specifies a transition zone between the stop and pass bands of the filter. This transition should not be instantaneous because it would produce an unstable filter. In the 3rd line is specified the tolerances of the filter. The filter will have 3 dB of tolerance in the pass-band, or roughly a gain that can vary between 0.5 and 2. The filter will have an attenuation of at least 20 dB in the stop-zone, or a factor of 100. When desigining your own filter, you can see its frequency and phase response by using the function **freqz**.

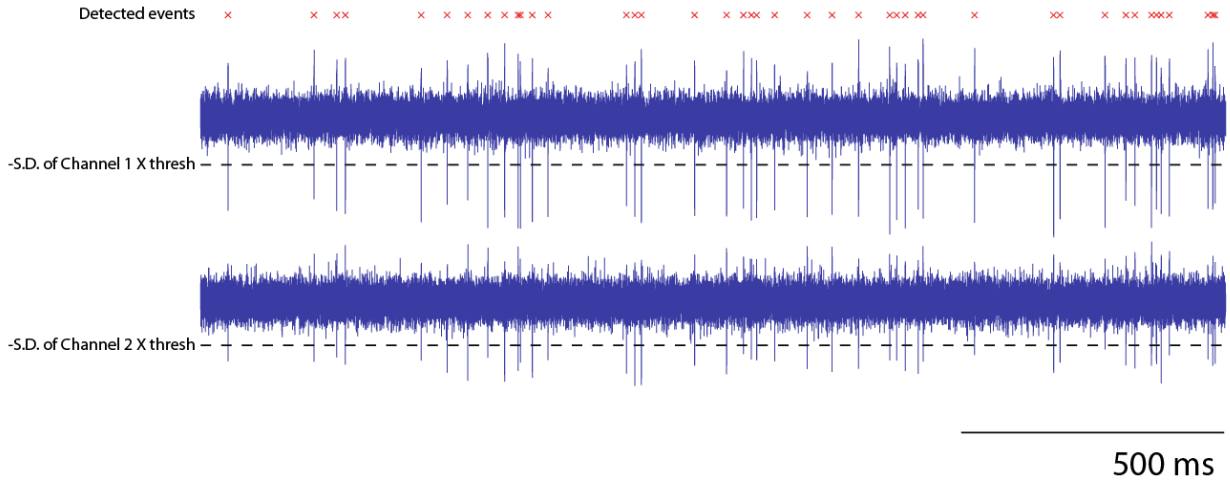
2.2 Defining parameters

All parameters for spike sorting are stored in **spikes.params** as described in section 4. The default values for these parameters are defined in the function **ss_default_params**. This function creates an empty **spikes** object with a fully populated params field. The user should edit this file or make their own copy. One parameter, the firing rate in Hz (**Fs**), is taken into **ss_default_params** as an argument.

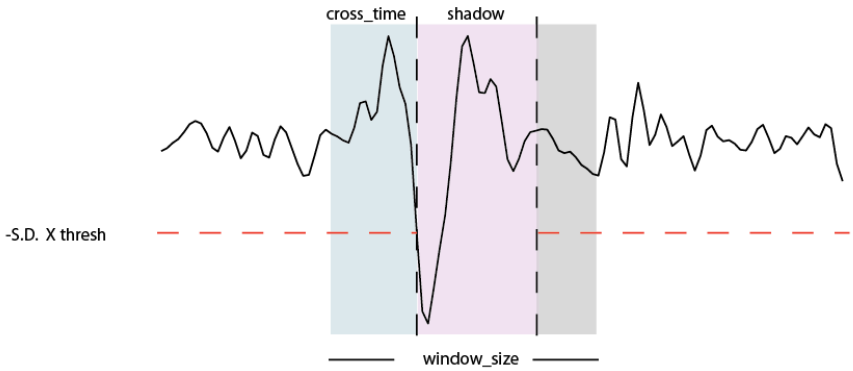
2.3 Spike event detection

First the user must put the data set into the proper format. The input **data** for **ss_detect** can either be in a matrix format of the form **[trials x sample x channels]** or as a cell array of the form **{trials}[samples x channels]**. If a data set is too large to fit into memory, **ss_detect** can be called multiple times with new data sets. Trials in the new data set will be concatenated to the trials from the previous data set.

Spike detection is performed by setting a negative threshold and identifying events that cross this threshold. The user can either set this threshold manually or set a number of standard deviations above the mean. In the latter case, the standard deviation is calculated for each channel, then multiplied by **spikes.params.thresh** and then used as the threshold. Note that if **ss_detect** is called multiple times, the standad deviations will be based only on the first data set. When there are multiple channels, an event is detected when any channel crosses threshold. In order to avoid one event triggering multiple threshold crossings, there is a period of time set by **spikes.params.shadow** (ms) that turns off detection for a short period of time after every threshold crossing. This is also called "censoring".

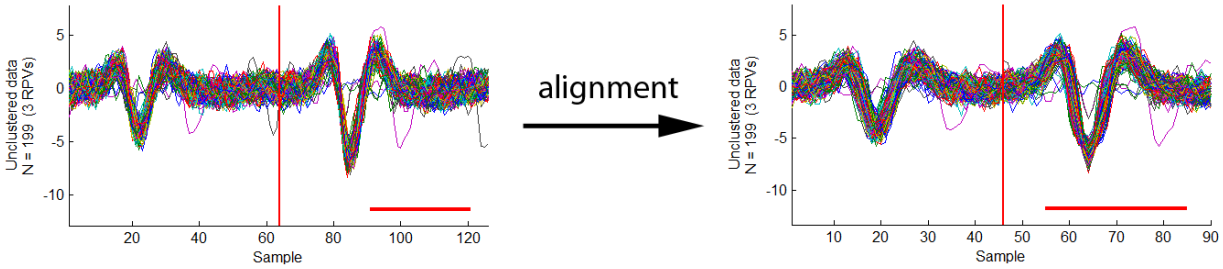


For every detected event, **ss_detect** extracts a window from each channel. Two parameters determine this window. The total length of the window is set by **spikes.params.window_size** (ms). The position of the threshold crossing event within the window is set by **spikes.params.cross_time** (ms)**.** These waveforms are stored in **spikes.waveforms** with the format **[event x sample x channel]**. The time of the event and its trial number are stored in **spikes.spiketimes** (s) and **spikes.trials** respectively. The array **spikes.spiketimes** stores the time of the spike event within a trial. The absolute time of the spike event is stored in **spikes.unwrapped_time** by appending each trial. A short buffer time is imposed between each trial as specified in **spikes.params.trial_spacing** (s).

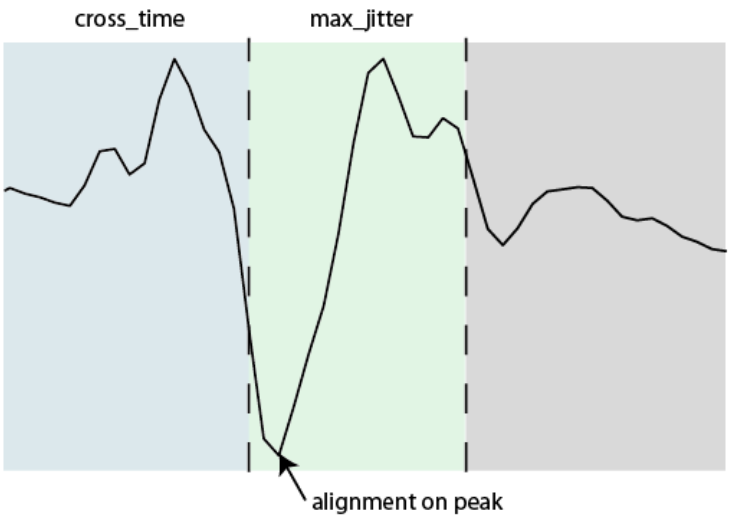


2.4 Waveform alignment

In order for spike waveforms to be compared fairly, they must be well aligned. Aligning a waveform on its sampled threshold crossing is problematic because the exact moment of threshold crossing is sensitive to noise. Therefore, the next step in spike sorting is to re-align waveforms on their peak.

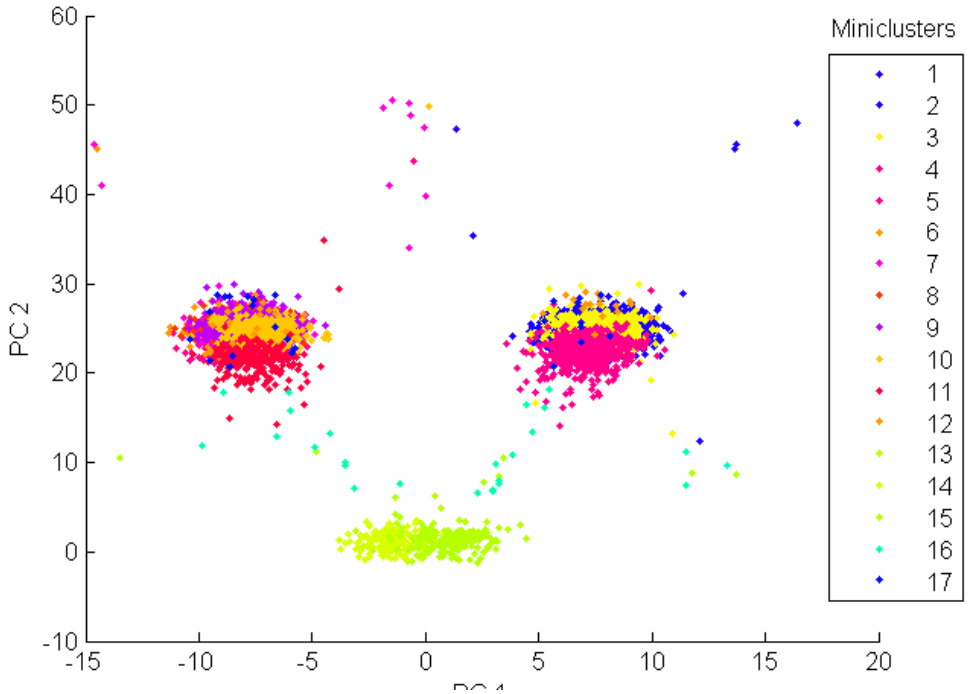


This is accomplished by looking for the peak of the waveform after the threshold crossing, and sliding the waveform window by this amount. The maximum time range that the window can slide is set by **spikes.params.max_jitter** (ms). Alignment is performed on the channel with the most significant peak and all other channel waveforms are slid by the same amount. Spline interpolation is used to find the true peak which may occur in between samples. Note that the size of the window after alignment is set by **spikes.params.window_size**. The actual size of the window before alignment is **spikes.params.window_size + spikes.params.max_jitter**.



2.5 Over-clustering with K-means

The next step in the spike sorting algorithm is to split the waveforms into many "miniclusters". A minicluster is a group of waveforms that's small enough that it is likely to contain a subset of waveforms produced by exactly 1 neuron. Later these miniclusters will be combined to form full-fledged clusters representing all the waveforms from 1 neuron. The K-means algorithm is used to quickly break the data set into many miniclusters. A single parameter, **spikes.params.kmeans_clustersize**, determines how small the miniclusters will be. It is guaranteed that no minicluster will contain more spikes than **spikes.params.kmeans_clustersize** by more than a factor of 2.

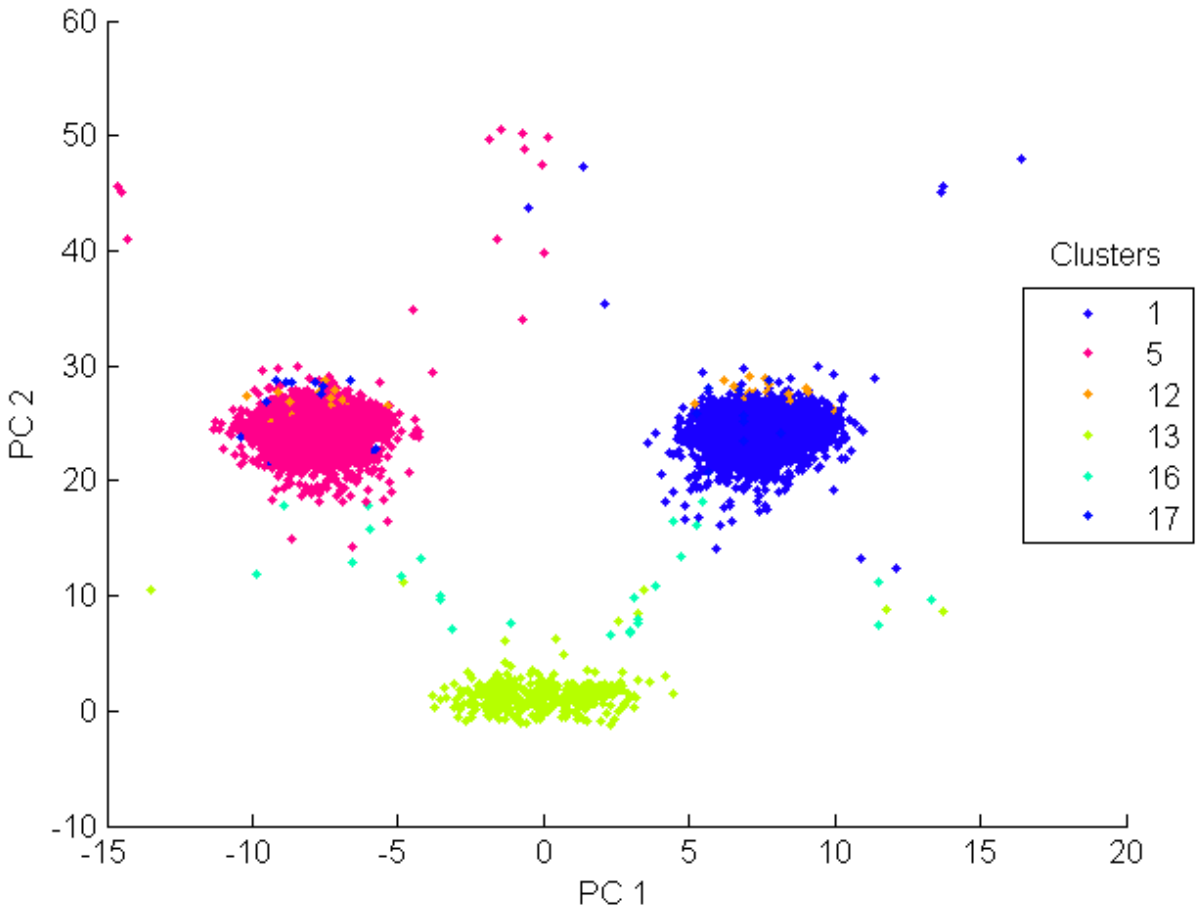


2.6 Cluster similarity and interface energy

It is difficult to form a statistical model of a single-unit waveform cluster a priori. Here we only assume a single-unit cluster should form a continuous cloud of data points. We define continuity using a quantity called the interface energy. The details of this calculation can be found in (Fee 1996b). In brief, the interface energy is a non-linear similarity. The similarity metric was designed to only produces large values when 2 clusters are very close to each other, such as when 2 clusters really represent a single cluster that was cut in half. In this case, the 2 clusters will have a large **energy** because of their common **interface**. In this step of the algorithm, the interface energy between every possible pair of miniclusters is calculated. The logic of creating a set of miniclusters and then combining them is that this interface energy is more easily calculated on miniclusters than on every pair of spikes contained in the entire data set.

2.7 Aggregation

Finally, miniclusters are aggregated. The pair of clusters that has the highest interface energy are merged together and the then its interface energies are recalculated. The stop criterion for aggregation is set by **spikes.params.agg_cutoff**. Higher values of this cutoff allow for more aggregation. It is difficult to know what this value should be, but it is easy to merge or split clusters manually in **splitmerge_tool**, so **spikes.params.agg_cutoff** does not need to be set precisely.


60
50
40
30
20
10
0
-10
-15
-10
-5
0
5
10
15
20
PC 2
PC 1
Clusters
1
5
12
13
16
17

3 Manual inspection of sorted spikes

It is critical to examine the results of automated spike sorting as there are many ways in which clustering can fail.

- The threshold for spike detection may be inappropriate.
- Spike waveforms from the same neuron may be separated into multiple clusters.
- Waveforms from multiple neurons may be combined into a single cluster.
- Waveforms may represent non-neuronal events such as electrical noise.
- A cluster may be inconsistent over time, dropping in or dropping out of the recording session.

Manual inspection must be used to fix errors in waveform aggregation, determine which clusters represent single units, and to quantify the quality of a sorted cluster. The plots and statistics described below are provided to facilitate this process by allowing the user a variety of informative views into their data.

Several functions listed below have parameters that are set in **ss_default_params** and are defined in section 4.

Note that in the following functions, the argument **which** defines which events are displayed in a flexible manner as defined in **get_spike_indices**.

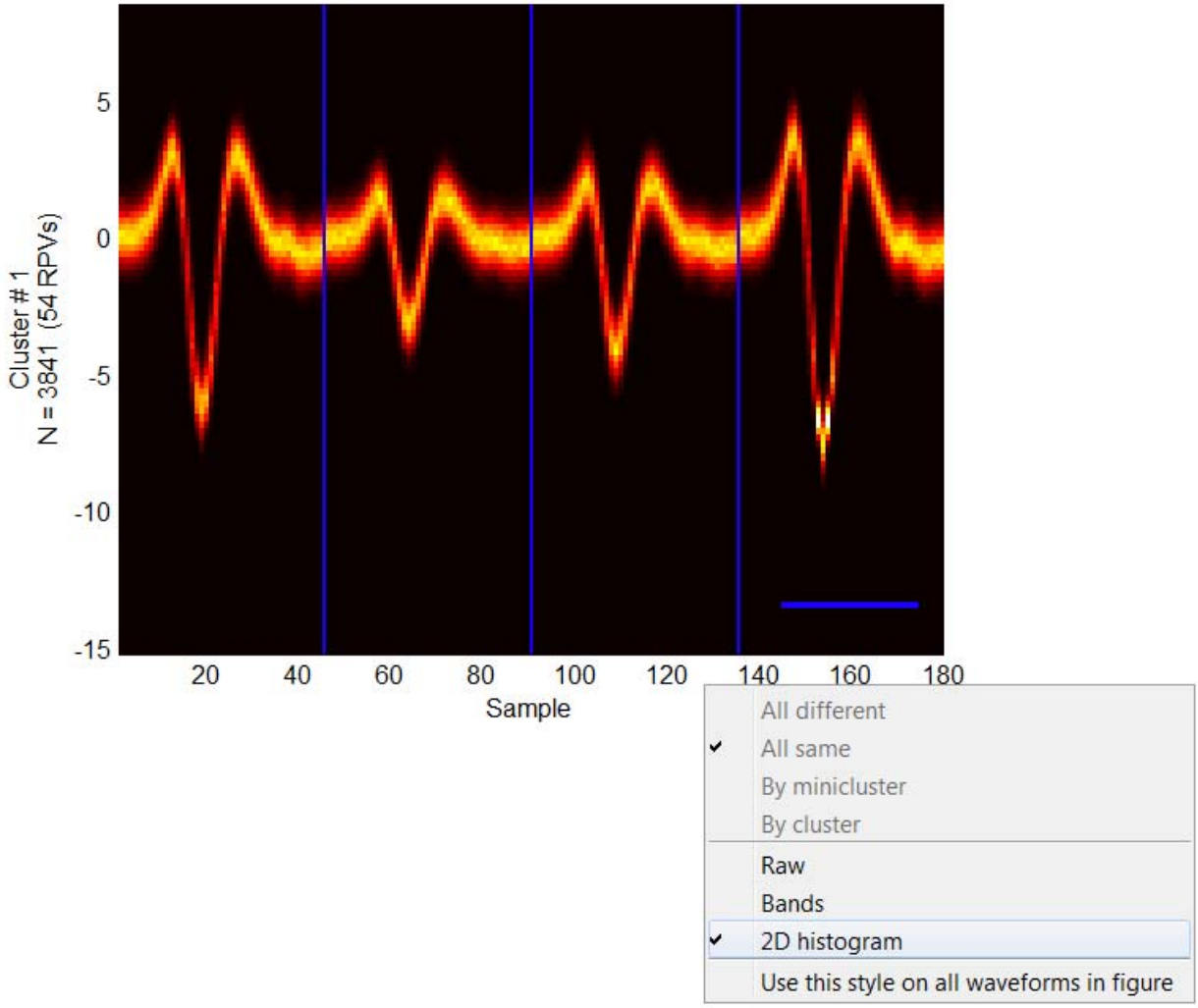
Data type of "which"	**Interpretation**
List of unsigned integers	Plots all waveforms of the listed cluster IDs. If waveforms have not yet been assigned to clusters, then minicluster IDs are used.
Boolean array	Must be same length as number of event. All indices for true values (1) are plotted.
'all'	All waveforms are plotted.

For example, **plot_waveforms(spikes, 1)** plots the waveforms of cluster #1.

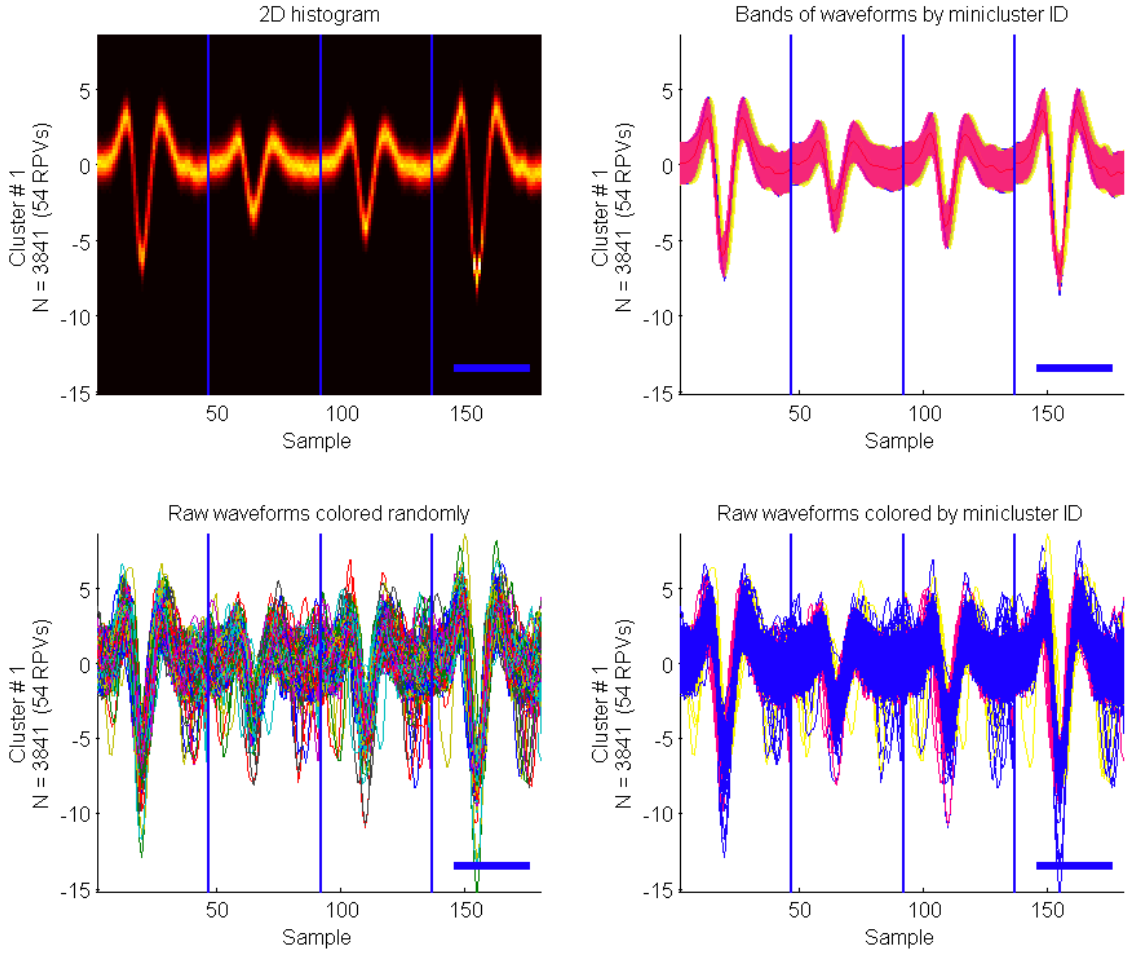
3.1 Plots to examine single clusters

3.1.1 plot_waveforms(spikes, which)

This is the primary function to view waveforms.



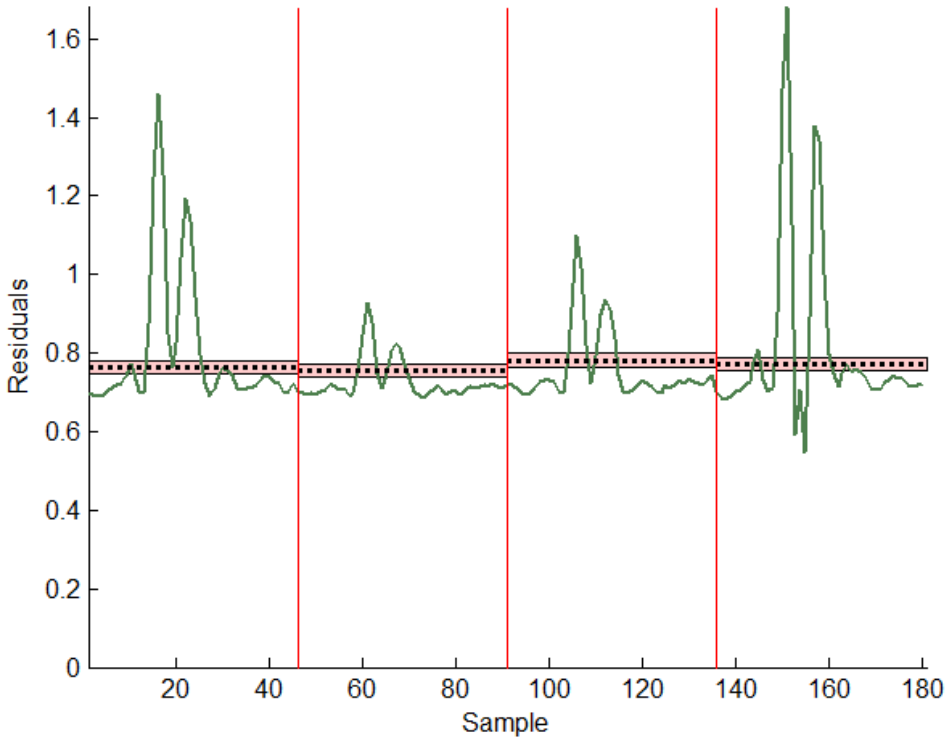
The vertical and horizontal lines are colored to match the color associated with the given cluster. Vertical lines separate waveforms from different electrodes. The horizontal line is a time scale bar whose length is set in **spikes.params.display.** The y-axis gives the number of waveforms in this cluster as well as the number of interspike intervals shorter than the defined refractory period (RPVs). The limits of the y-axis are set to cover the largest event in the entire **spikes** object. There is a context menu that can be accessed by right-clicking inside the plot. When the plot showsa 2D histogram, one must right-click slightly outside the axes. Via this context-menu, the user can select how to display and color the waveforms.



The choice of how to plot the data can either be shown as a 2D histogram, as bands representing 95% of the waveform data, or as the raw waveforms. The user can also choose whether to separate and color the data by cluster ID, minicluster ID, all different, or all same. Different options will be available depending on whether the data has been clustered yet and on how the data is being displayed. When bands or raw waveforms are displayed, the user can left-click (right-click) a band or waveform to raise (lower) all data of the same color. A final option in the context-menu allows the current view to be applied to all instances of **plot_waveforms** in the same figure.

3.1.2 plot_residuals(spikes, which)

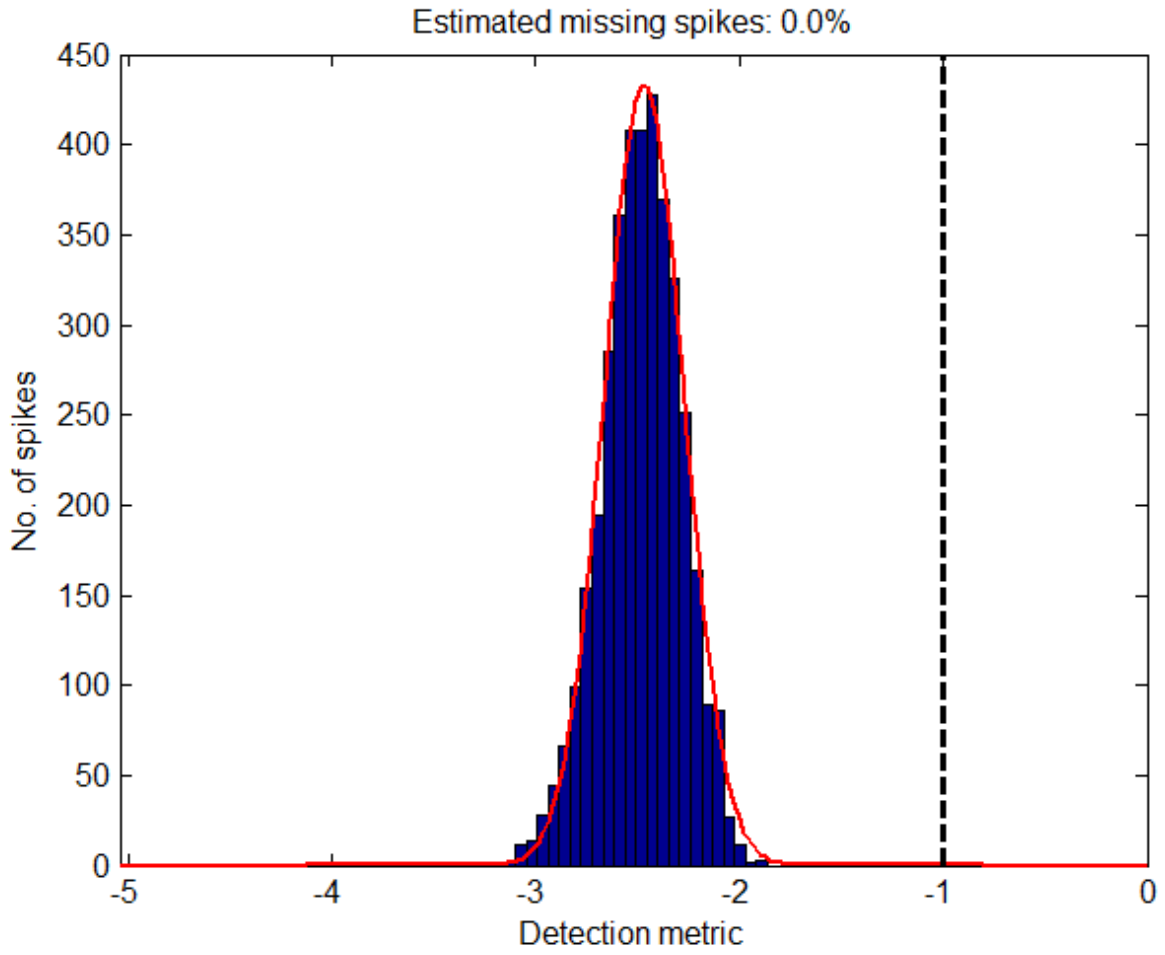
Residuals are the standard deviation of the mean waveform as a function of sample. In the ideal case, the residuals are equal to the standard deviation of the background noise. Although sometimes difficult to interpret, **plot_residuals** can be used to diagnose whether there is unusual structure in the variability of a cluster.



This plot shows the residuals as a function of sample. The red vertical lines separate data from different channels. The dashed line is the standard deviation of all data for that channel. The pink band around the dashed line is the 95% confidence interval.

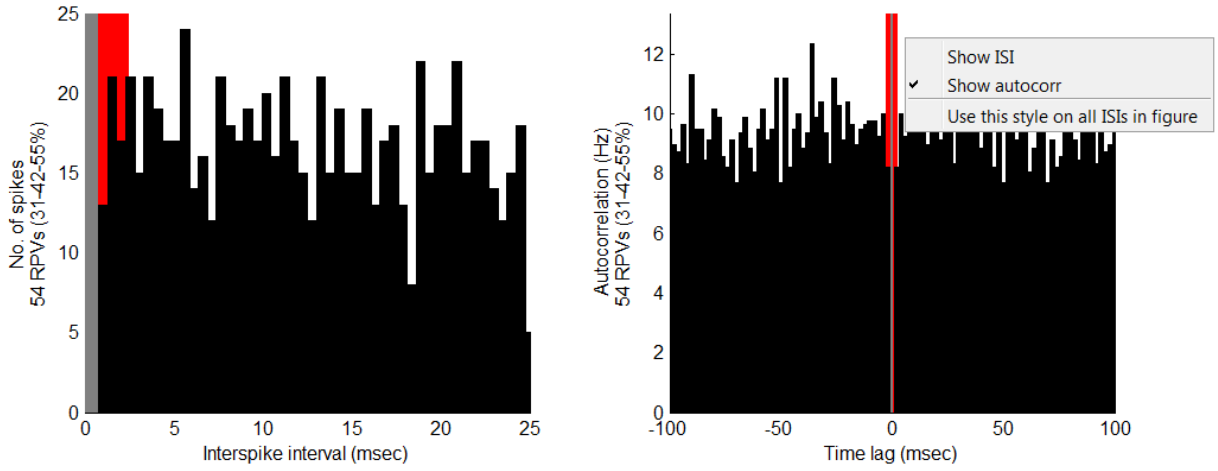
3.1.3 plot_detection_criterion(spikes, which)

It is important to check whether a cluster is well-separated from the threshold for spike detection. This function plots a histogram of the value of the negative peak of each waveform. This value is normalized by the threshold for detection so that a value of -1 is just at threshold. For multi-channel data, only the waveform on the channel with the largest negative peak is used. A cluster is well-separated from threshold if its distribution is far from -1. The estimated % of spikes that did not cross threshold is given above the plot. This value is estimated by first fitting the histogram with a Gaussian distribution (red line). Then the % of the Gaussian that is above -1 is easily calculated. Note that this estimate is only meaningful if the peaks are truly Gaussian distributed. See quality measures for more information on estimating detection errors.



3.1.4 plot_isi(spikes, which)

By definition, a real neuron cannot fire spikes during its absolute refractory period. Therefore, the number of interspike intervals in a cluster that are less than the refractory period can indicate how contaminated the cluster is. Further, a real neuron typically has a longer relative refractory period where the probability of firing is reduced from the normal mean firing rate. This function allows the user to examine the firing statistics of a cluster on both time scales.



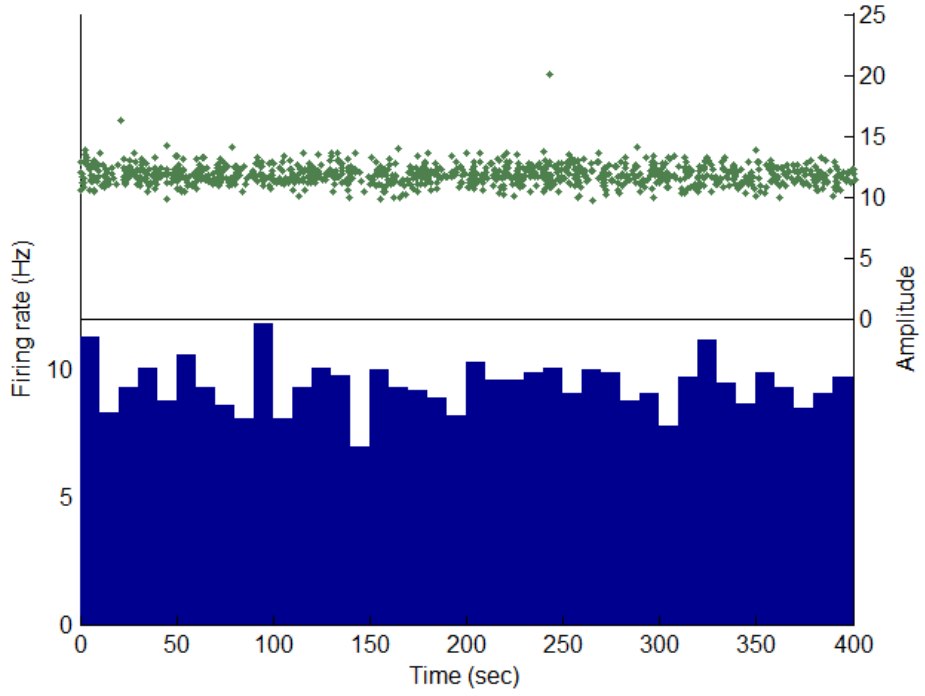
On the left is the interspike interval distribution for a cluster. On the right is the autocorrelation function of the spike train. The user can access a context menu by right clicking the background of the plot. There the user can switch between these two modes or apply the current mode to all instances of **plot_isi** in the current figure. The red vertical band represents the absolute refractory period. The gray region is the “shadow” region. No interspike interval can be shorter than this limit because of the way that spike detection is performed. The width of the histogram bars and the range of the x-axis are settable parameters.

The y-axis label gives the number of refractory period violations, along with 3 numbers that estimate the percent contamination of the cluster. Spikes that occur less than a refractory period apart are assumed to be misclassified spikes. We use the rate of such events to determine the overall rate of contamination. This requires the assumption that misclassified events occur at random, *i.e.*, their event times are not correlated with the event time of correctly classified spikes. See the description of **ss_rpv_contamination** below or its comments for more information.

In the parentheses on the y-axis, the 2nd number is the estimate of percent contamination based on refractory period violations. The 1st and 3rd numbers are a 95% confidence interval on this estimate. Note that the confidence interval makes an additional assumption of Poisson statistics for the contaminating spikes.

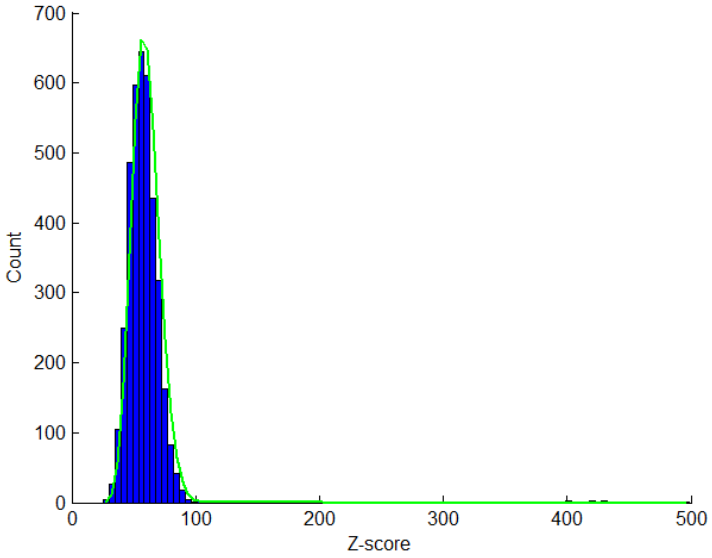
3.1.5 plot_stability(spikes, which)

The statistics of a cluster may not be stationary over time such as occurs during electrode drift. This function displays the amplitude (min to max voltage) and firing rate of a cluster over the duration of the experiment. If the experiment consists of multiple trials, the absolute time of a spike during the experiment is modeled by adding a constant time delay between trials. For economy of space, the amplitude and the firing rate are shown on the same plot. If the cluster is large, only a subset of amplitude data is shown in the scatter plot.



3.1.6 plot_distances(spikes, which)

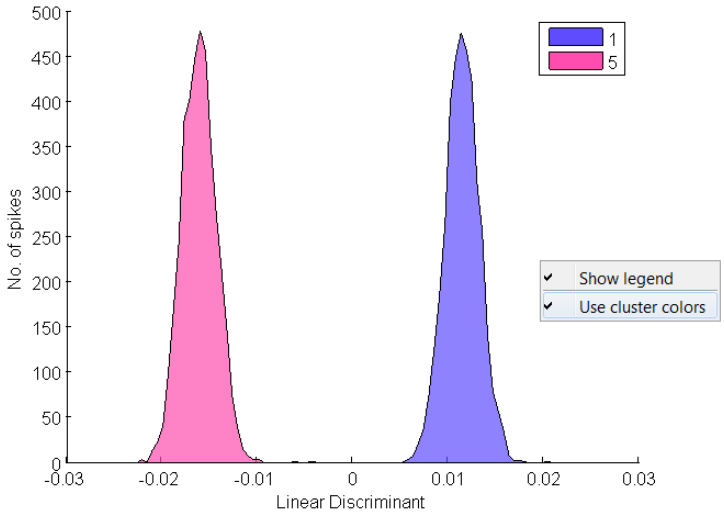
If a cluster is well-described by a Gaussian distribution, then the Mahalanobis distance from the mean of the cluster to each waveform will follow a χ^2 distribution. This function plots a histogram of the Mahalanobis distance of all waveforms. The green line is the expected distribution assuming a χ^2 distribution. Histogram values that are far out in the tail of the distribution can be interpreted as outliers.



3.2 Plots to compare two clusters

3.2.1 plot_fld(spikes, which1, which2)

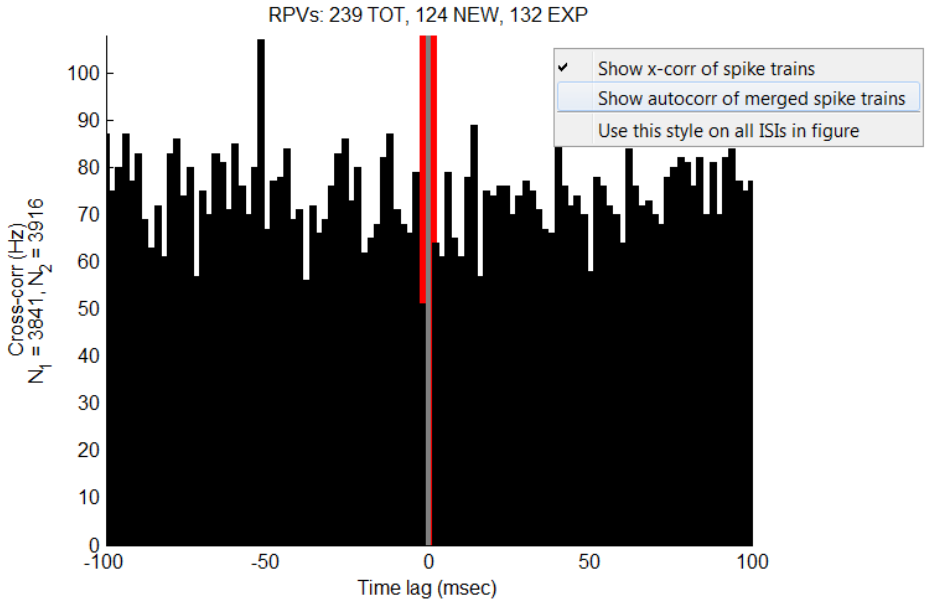
This function plots histograms of the projection of two clusters onto their Fisher linear discriminant. The Fisher linear discriminant is the projection that most separates two multi-variate Gaussian distributions. This projection is a quick way to see how well separated two clusters are from each other. In general, two clusters will be more separate than they appear in this dimension because two clusters may be more separable in a higher dimensional space. A context menu can be accessed by right-clicking the background of the plot. It toggles whether the legend is shown and whether the histograms should use the cluster colors or the default colors (red and blue). Left (right) clicking on either histogram will send it to the front (back).



3.2.2 plot_xcorr(spikes, which1, which2)

If two clusters contain waveforms from the same neuron, their spike trains should show some structure in their cross-correlation. This function plots the cross-correlation between the spike trains of two clusters. As an alternative, the data can be plotted as an auto-correlation of a spike train formed by merging the two spike trains. A context-menu allows the user to switch between these two modes. The user can also apply the current mode to all instances of **plot_xcorr** in the current figure.

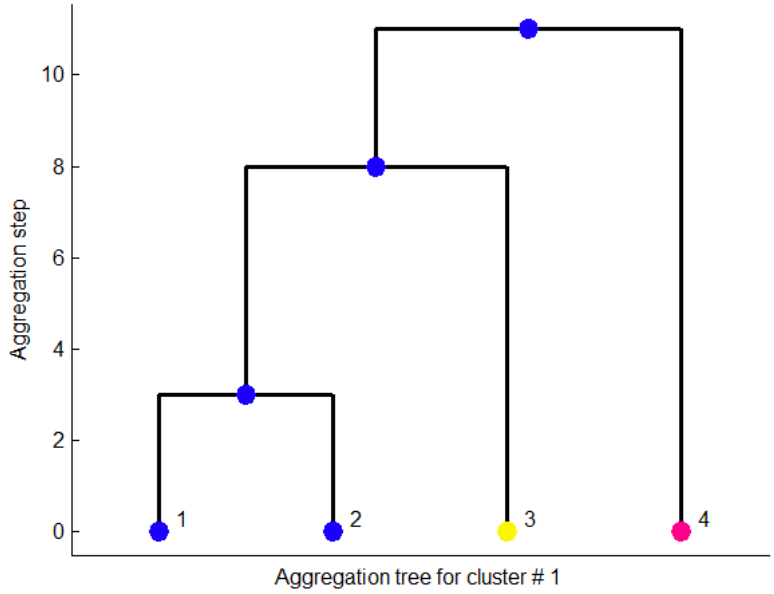
The label of the y-axis indicates how many events are in each cluster. The title indicates how many refractory period violations (RPVs) there are in the merged spike train (TOT), how many additional RPVs were created by merging the two spike trains (NEW), and how many would be expected if the two trains were uncorrelated (EXP). The vertical red band indicates the RPV region while the gray vertical band represents the “shadow” used in spike detection.



3.3 Plots to inspect aggregation tree

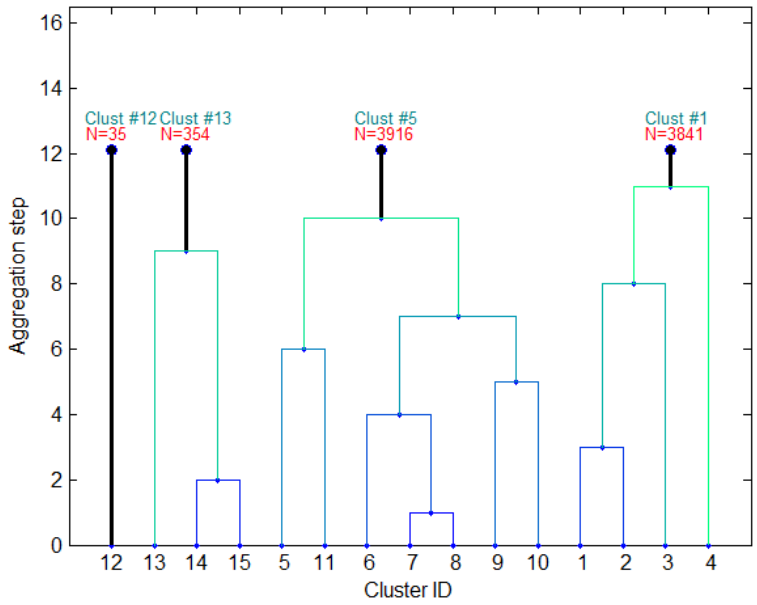
3.3.1 plot_cluster_tree(spikes, clusID)

This function plots the aggregation tree for a particular cluster. During each iteration of the aggregation procedure, a pair of subclusters is merged. This is represented on the graph by two nodes being joined by black lines at a higher node. The “Aggregation step” of the y-axis gives the iteration number when the merging occurred. Therefore, the aggregation procedure progresses from bottom to top. Nodes are labeled by their minicluster ID. The color of the node is the color associated with the cluster ID. So in the example below, every merge produced a cluster with an ID of 1.



3.3.2 plot_agg_tree(spikes)

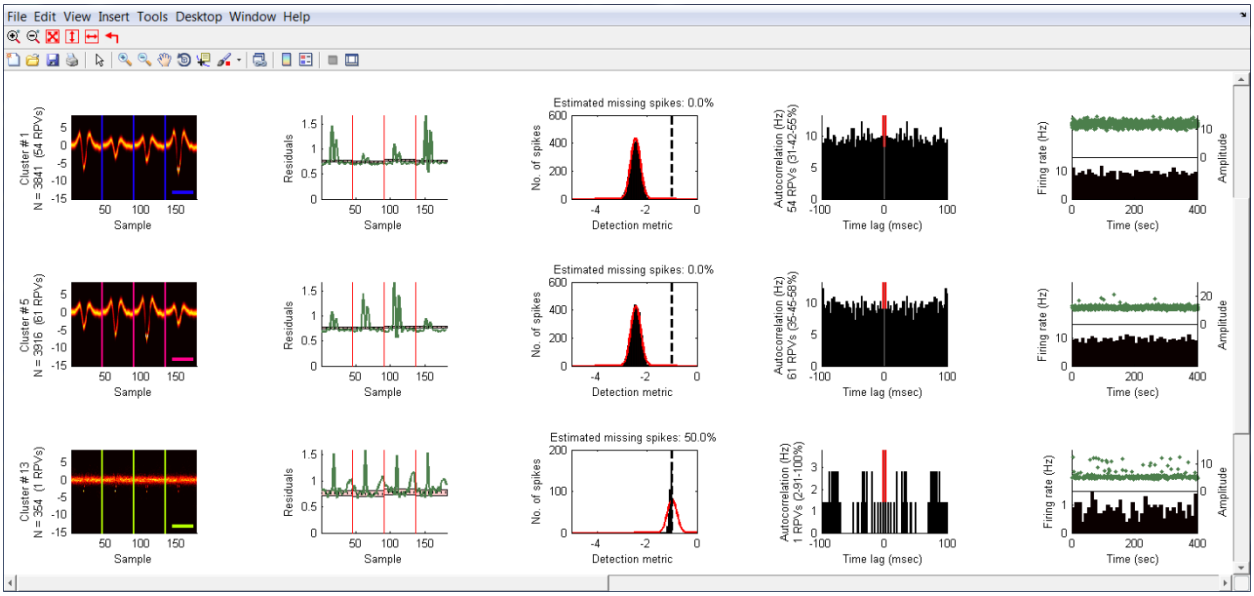
This function allows you to see all aggregation trees simultaneously. The top of each tree shows the final cluster ID and the total number of member waveforms.



3.4 Figures to browse all data

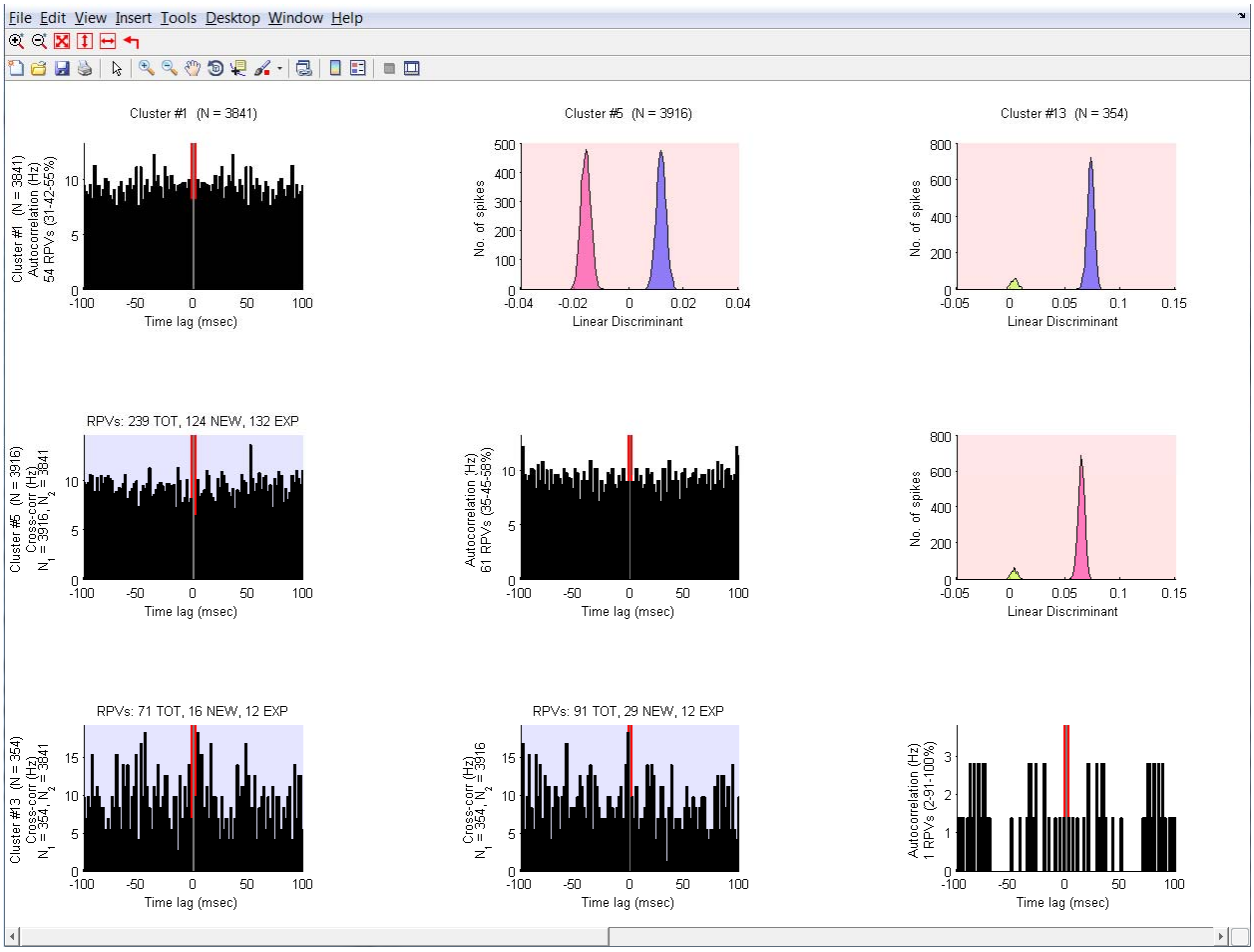
3.4.1 show_clusters(spikes, clusterIDs)

This function generates a figure that allows you to see several important plots for all clusters indicated in a list. From left to right, each cluster is plotted using **plot_waveforms, plot_residuals, plot_detection_criterion, plot_isi,** and **plot_stability.**



3.4.2 compare_clusters(spikes, clusterIDs)

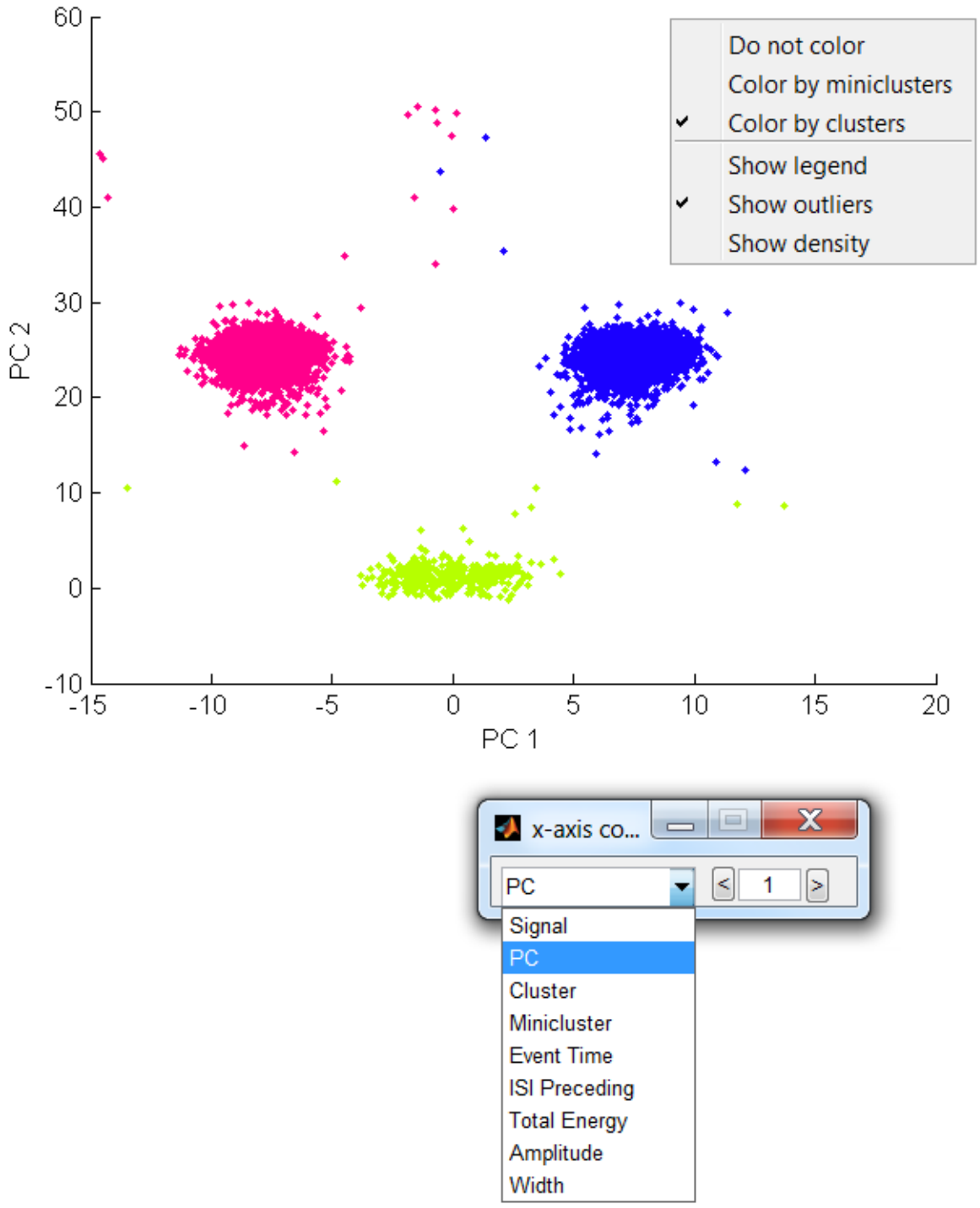
This function generates a figure that allows you to compare the waveforms and spike times pairs of clusters indicated in a list. The row and column determine which two clusters are being compared. The plots along the diagonal are instances of **plot_isi**. The plots in the lower left are generated by **plot_xcorr**. The plots in the upper right are generated by **plot_fld**.



3.4.3 plot_features(spikes, which)

This function shows a scatter plot of two statistics from the indicated spike events. The statistic plotted for each axis can be change by clicking on the axis label. Some statistics even require a parameter.

Statistic	Description	Parameter
Signal	Voltage value at a particular sample	Which sample
PC	Principal component	Which component
Cluster	Cluster ID	
Minicluster	Minicluster ID	
Event Time	Absolute time of event (s)	
ISI Preceding	Interspike interval before spike (ms). Cutoff at some threshold.	
Total Energy	Sum of squares of waveform	
Amplitude	Range of waveform values	
Width	Time difference between maximum and minimum value of waveform (ms)	



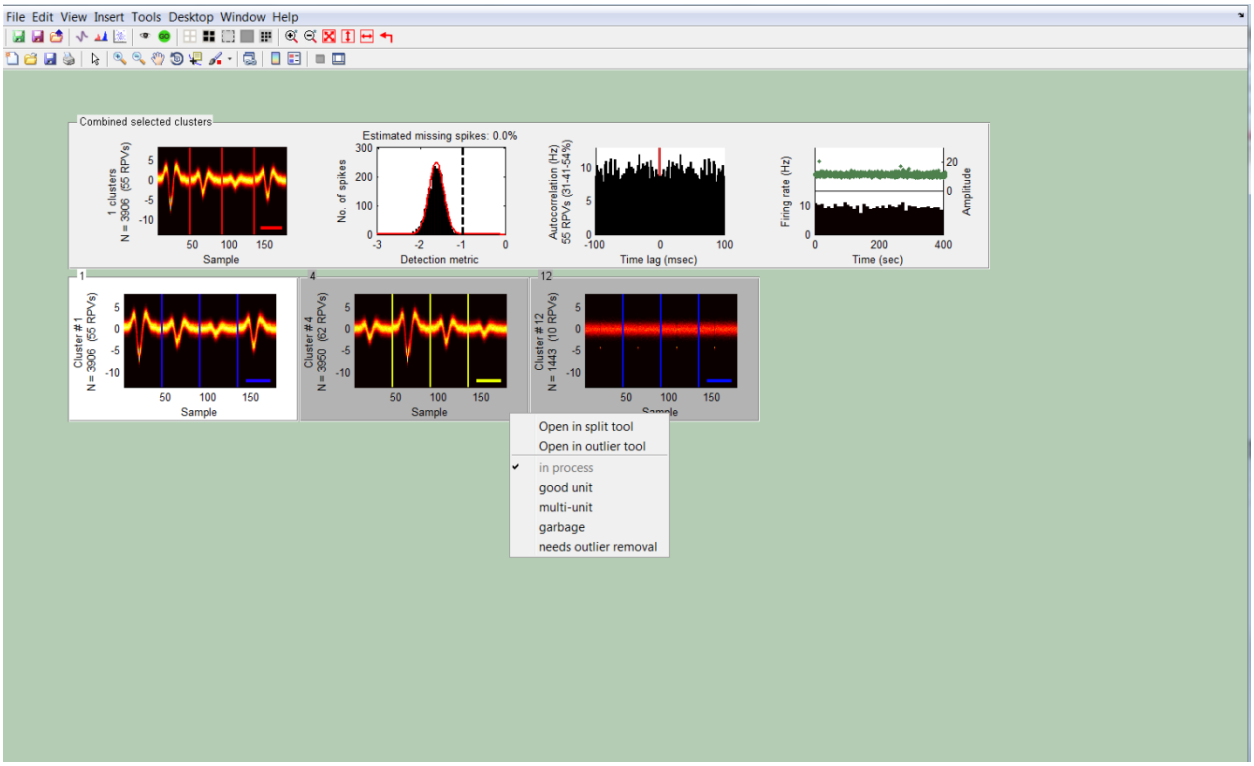
Similar to **plot_waveforms**, there is a context menu where the user can set how the data points should be grouped and colored. The user can also choose whether to show the legend, show outliers as black dots, or to replace the scatter plot altogether with a 2D histogram. Finally, the user can left (right) click a data to bring all data points of that color to the front (back).

3.5 Merge/Split/Outlier Tool

The above graphical tools are useful ways of examining data, and the **splitmerge_tool** brings these plots together while allowing the user to manipulate the results of automated spike sorting. Within this tool, the user can merge multiple clusters into a single cluster, split a single cluster into multiple parts, and remove spike events that have outlier waveforms.

3.5.1 Merge tool

The **splitmerge_tool** is initiated by calling **splitmerge_tool(spikes).** The tool begins with the **merge_tool** which allows the users to combine several clusters. This tool displays all clusters in a series of panels. These panels can be selected by left-clicking them with the mouse which causes them to turn white. The panel at the top of the tool is initially blank, but can be populated by selecting cluster panels and hitting the eye button. This will show instances of **plot_waveforms**, **plot_detection_criterion, plot_isi,** and **plot_stability** for the set of spike events represented by all the clusters in the selected panels.



The selected clusters can be merged into a single cluster by clicking the go button. Individual clusters can be further manipulated via a context menu that can be accessed by right-clicking its panel. This menu allows the user to open **split_tool** and **outlier_tool** (see below). It also allows the user to label clusters as defined in the parameters (see below) causing the panel to change color. Several buttons allow additional functionality.

Buttons (with hotkeys)

 **(s)** Saves changes to **spikes** object to the MATLAB workspace.

 Saves **spikes** object to a file.

 Loads **spikes** object from a file. File must be .MAT file containing a variable called **spikes.**

 Opens an instance of **show_clusters** for the selected clusters.

 Opens an instance of **compare_clusters** for the selected clusters.

 Opens an instance of **plot_features** for the selected clusters.

 (e) Populates top panel with information about the combined spike events from all of the selected panels.

 **(x)** Merges the clusters from the selected panels.

 (a) Selects all panels.

 **(d)** De-selects all panels.

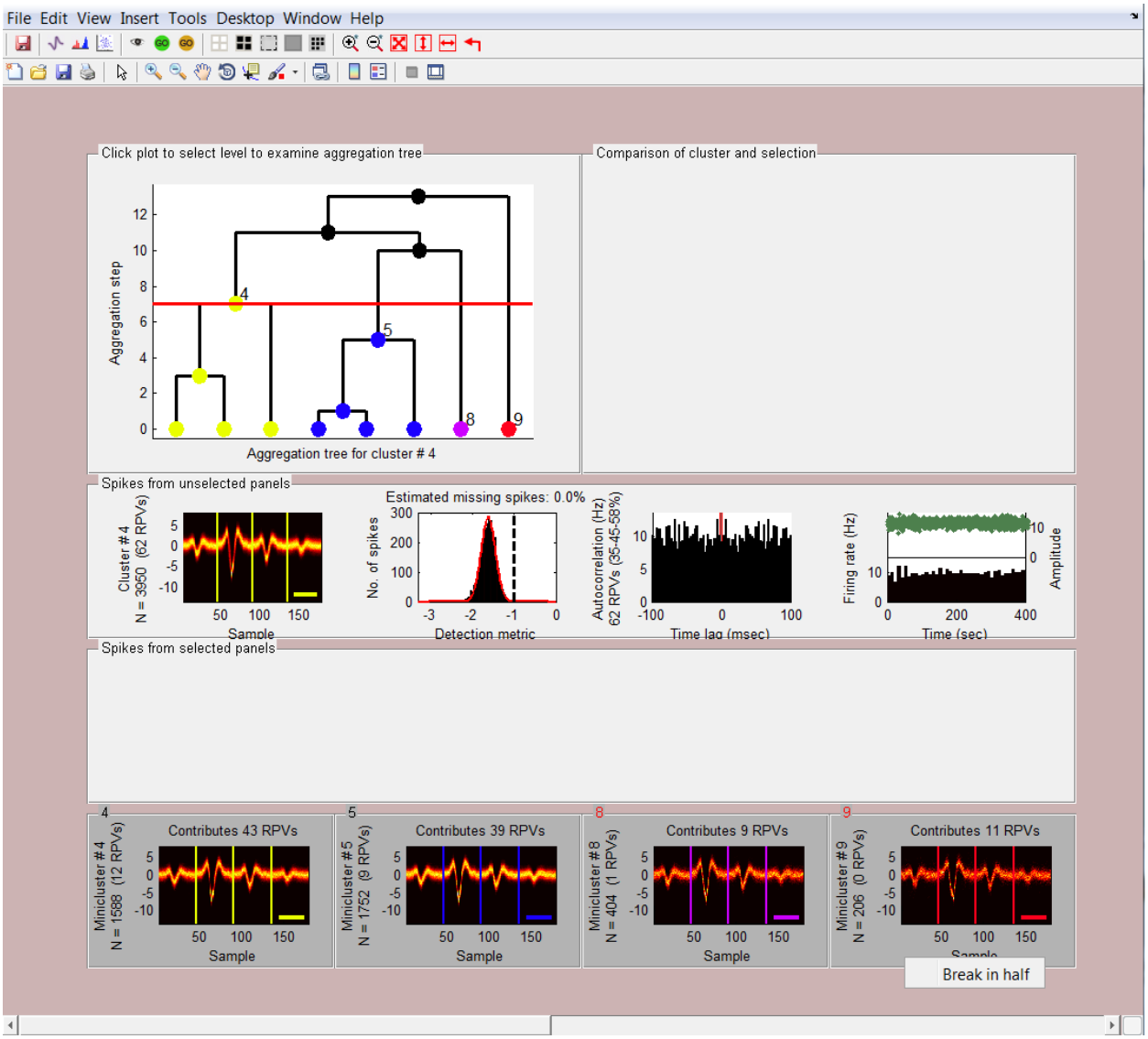
 **(h)** Hides the selected panels.

 **(r)** Reveals any hidden panels.

 **(p)** Rearrange the panels to fit the width of the figure.

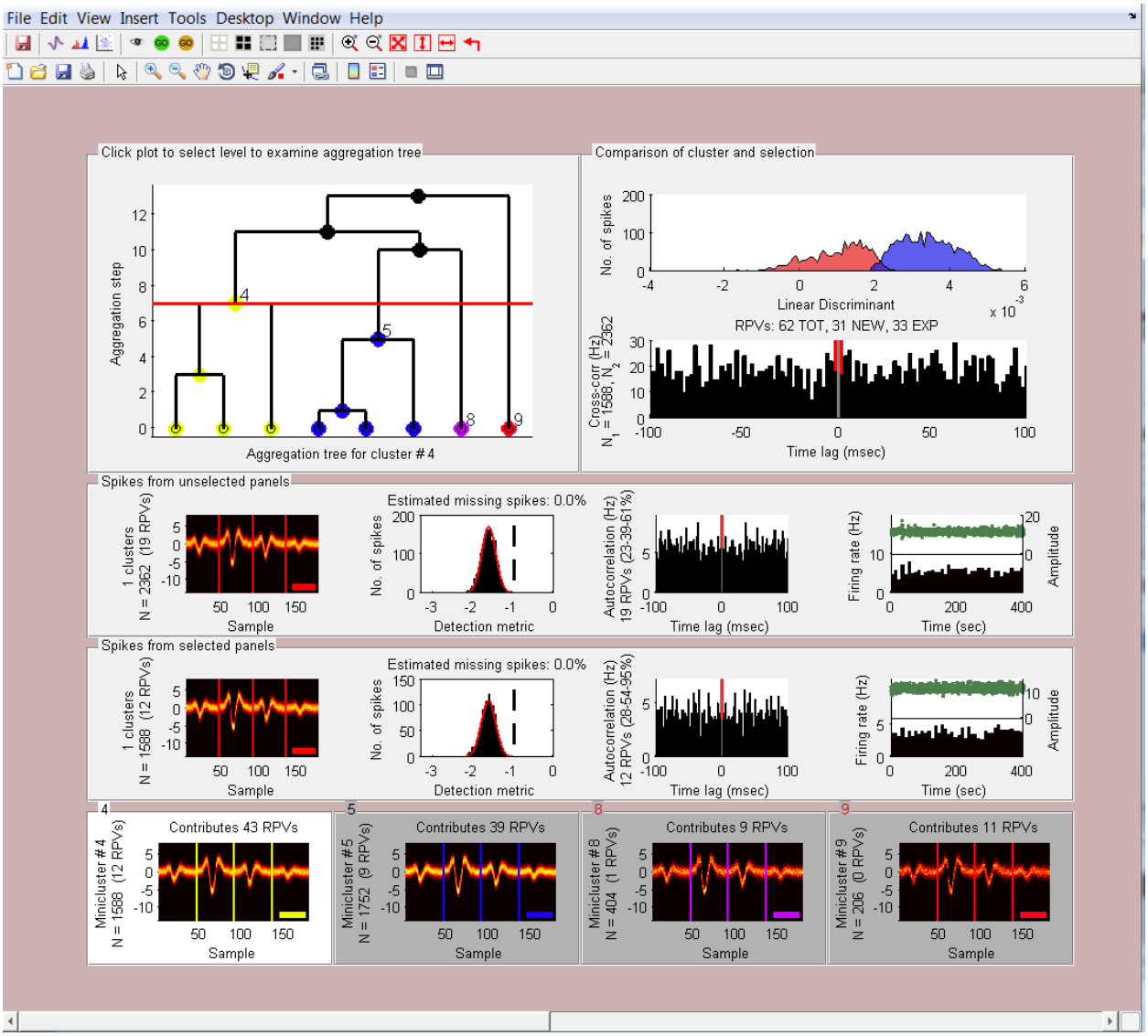
3.5.2 Split tool

Every cluster is formed by aggregating miniclusters together. This process can be described by a tree where each node in the tree represents the merging of two sets of miniclusters. The **split_tool** allows the user to examine the branches of this tree and remove any branches that may have been included erroneously.



The top-left panel of this tool displays the tree for the given cluster using **plot_cluster_tree**. A red horizontal line on this tree determines which sub-branches of this tree are viewable. Wherever this line cuts the tree, the node immediately below can be viewed. The tree is colored so that all sub-branches have nodes of the same color. The user can change the level of the red line by clicking elsewhere in the tree plot.

At the bottom of the figure is displayed an instance of **plot_waveforms** for the spike events of each subcluster currently highlighted in the tree. These panels can be selected and deselected the same way as in the **merge_tool.** The two wide panels in the middle of the tool each can show an instance of **plot_waveforms, plot_detection_criterion, plot_isi,** and **plot_stability.** The upper one shows these plots for all unselected panels. The bottom one shows these plots for all selected panels. These two panels can be updated by hitting the EYE button.



When the eye button is hit, the upper-right panel is also updated. This panel displays **plot_fld** and **plot_xcorr** comparing the spike events from the selected and unselected panels.

If the user determines that the selected panels contain subclusters that should not have been included in the main cluster, then the GO button can be pushed to remove the selected subclusters from the main clusters. They will become full clusters in the **spikes** object and will appear in the **merge_tool** after the SAVE button is pressed. If the green GO button is pressed,

the removed subclusters will each become their own cluster. If the yellow GO button is pressed, the removed subclusters will be merged into a single cluster.

Finally, it can occur that a minicluster itself needs to be split. This option can be selected by right-clicking in the panel of a minicluster, which is indicated by red title for the panel. The minicluster is then split exactly in half along its 1st principal component. It is suggested that the user view every minicluster of cluster to determine whether the minicluster was included by accident or whether the minicluster contains waveforms from multiple units.

Buttons (with hotkeys)

(s) Closes the **split_tool** and saves any changes made to the **merge_tool**.

Opens an instance of **show_clusters** for the selected subclusters.

Opens an instance of **compare_clusters** for the selected subclusters.

Opens an instance of **plot_features** for the selected subclusters.

(e) Divides the set of waveforms into those from selected panels and those from unselected panels. The top panels are updated to show various plots showing and comparing the two sets of spikes.

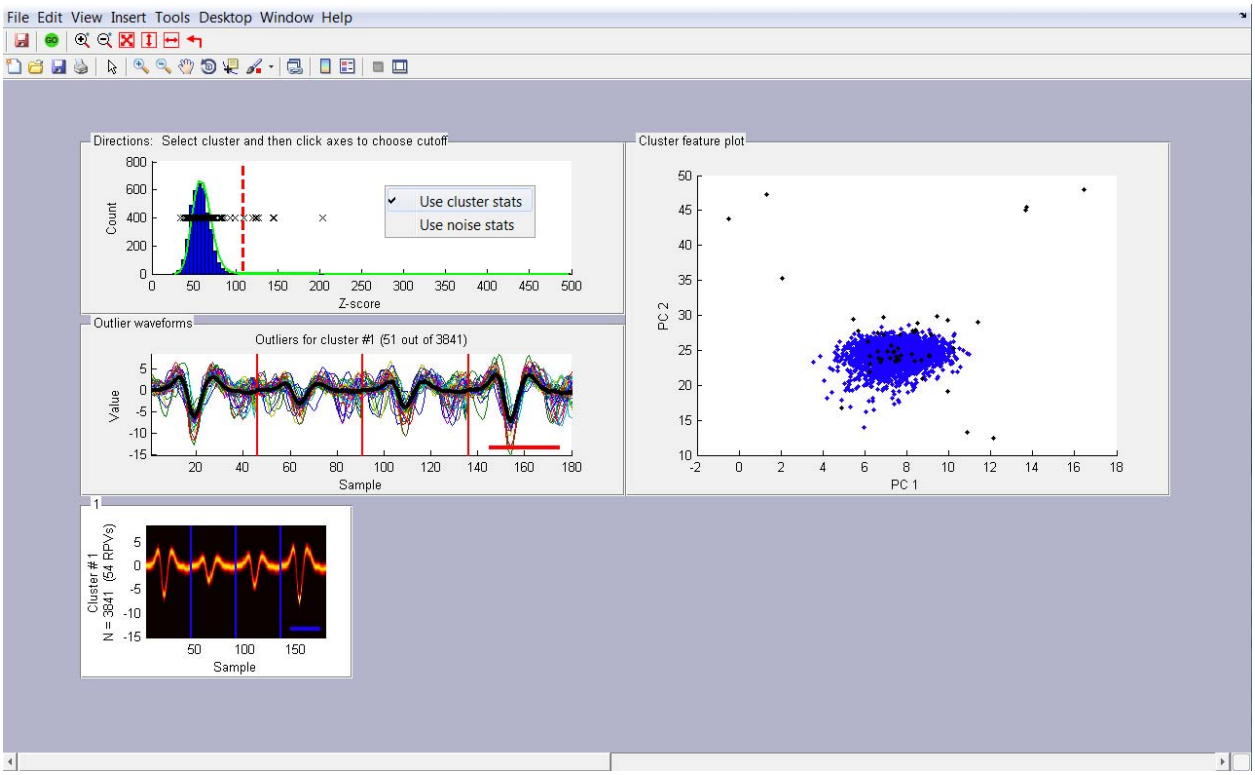
(x) Removes the subclusters from the selected panels from the overall cluster. Each selected panel becomes its own cluster.

(m) Removes the subclusters from the selected panels from the overall cluster and then merges them.

 These buttons act similarly to the ones described above for the **merge_tool**. They allow the user to select all, deselect all, hide, show, and rearrange subcluster panels.

3.5.3 Outlier tool

Unusual waveforms, such as noise events or overlapping waveforms, are assigned to the nearest cluster. Therefore, the user must remove these events manually at the end of automatic spike sorting. This is accomplished by opening the cluster in the **outlier_tool.**



In the **outlier_tool**, the cluster itself is displayed at bottom in an instance of **plot_waveforms.** The main tool for removing outliers is the plot at top left which is a special instance of **plot_distances.** It contains a histogram of the Mahalanbois distance of each waveform from the cluster center. A context-menu that can be accessed with a right-click controls whether the covariance matrix used in this calculation arises from the statistics of the cluster or the statistics of background noise. Overlaid on this is a green line representing the prediction of the histogram by the χ^2 distribution. When the histogram exceeds this prediction, that is evidence these waveforms are statistical outliers. Also shown is a scatter plot of **x**'s representing the location of spikes that contribute a refractory period violation. This is included to indicate whether cutting outliers will reduce the number of refractory period violations.

The striped vertical line in this plot is used to select a cutoff for outliers. Its position can be changed by left-clicking within the plot. When its position is updated, the plot below and the plot to the right are also updated. The plot labeled "Outlier waveforms" shows all of the waveforms to the right of the cutoff line. The thick black trace is the mean waveform for the cluster. The title indicates how many outliers were identified out of how many waveforms total. The right-

most plot is an instance of **plot_features**. The waveforms to the right of the cutoff line are represented by black dots.

The indicated outlier waveforms can be removed by hitting the GO button. All information about the waveforms is then stripped from the main fields of the **spikes** object and stored in the structure, **spikes.info.outliers**. If needed, these waveforms can be reintegrated into the main **spikes** structure by calling the function **reintegrate_outliers.**

Note: It is not recommended to remove an entire cluster as outliers as this makes it difficult to determine whether a valid cluster is well-separated from other events.

Buttons (with hotkeys)

(s) Closes the **outlier_tool** and saves any changes to the **merge_tool**.

(x) Marks as outliers any waveforms beyond the current cutoff.

3.5.4 SliderFigure

SliderFigure.m is a utility that allows the user to zoom in and out of the figure itself. The normal MATLAB zoom function only allows one to zoom in on the data of a single set of axes. The **SliderFigure** tool adds sliders to the figure if zooming the plots causes axes to not fit on screen. This tool can be used in any figure in MATLAB simply by calling **SliderFigure.** The figure must only contain uipanels and axes**.** See the m-file for more details on the parameters to this tool.

Zoom in to figure by 10%.

Zoom out of figure by 10%.

Fit plots to figure size.

Fit plots to figure height.

Fit plots to figure width.

Return figure to 100% zoom.

3.6 Quality measures

Quality measures estimate the percent contamination of a cluster. The functions below estimate the faction of spikes in a cluster that were included in as false positive events or omitted as false negative events. While this can be very useful in assessing the quality of a spike sorting

session, the user should be aware that each measure makes certain statistical assumptions. The descriptions below are meant to be brief. See the comments of the specific functions or our review paper on quality measures (J. Neurosci., in review) for more details.

For each quality measure listed below, there are two functions in the **quality_measures** directory that implement it. .One is designed to work with the **spikes** struct. and has the prefix **ss**_. The other was designed independently of the spikes struct and has the same file name but without the prefix. This was done so that those who do not use the other functions in the toolbox can still have simple access to the quality measures. For more information about how to use the un-prefixed functions, see the comments in those files.

3.6.1 False positive estimate based on refractory period violations

Implemented by **ss_rpv_contamination(spikes, clusterID).** A real neuron has a brief period after each spike when it cannot fire again, called the refractory period. This function uses the number of inter-spike intervals (ISIs) that are less than the refractory period to estimate a contamination rate.

The logic of this function is detailed in **rpv_contamination**, but the essential statistical assumption is that contaminating spikes which cause refractory period violations occur at times that are uncorrelated with the spike times of true spikes in the cluster. This function also returns 95% confidence levels on the estimate which make the further assumption of Poisson statistics.

3.6.2 False negative and positive estimates based on waveform distribution of pairs of clusters

Implemented by **ss_gaussian_overlap(spikes, clusterID1, clusterID2).** This function estimates false positives and false negatives from the spike waveforms of two different clusters. An error probability is estimated by assuming that the two clusters were generated by a mixture of 2 multivariate Gaussian distributions. The parameters of this distribution are fit using the Statistics toolbox function gmdistribution.fit.

Be aware when using this function that a two Gaussian model may not be suited to your pair of clusters. Non-Gaussian variability occurs during electrode drift, bursting, poor clustering, etc… Use the visualization tools to diagnose this.

Based on the Gaussian models, this function outputs a confusion matrix that gives an estimate of the false negative and positive errors for each of the two clusters. If C is the confusion matrix, then

C(1,1) is the probability of a false positive for cluster 1
C(1,2) is the probability of a false negative for cluster 1
C(2,1) is the probability of a false negative for cluster 2
C(2,2) is the probability of a false positive for cluster 2

This function should be applied to every pair of clusters to get an overall estimate of false positive and negative rates. All false positive probabilities for a particular cluster are independent and so should be

combined by multiplying the compliments, (*i.e.*, $1-\prod_{j}(1-p_j)$). The same is true for the false negative probabilities.

Note that the false positives events estimated from refractory period violations is not independent from the false positive events estimated from Gaussian overlap. Therefore, it is recommended to use which ever estimate is larger.

3.6.3 False negative estimate based on spike detection errors

Implemented by **ss_undetected(spikes, clusterID).** This function estimates false negative errors due to a detection threshold that is too high for the cluster. The function assumes that detection was performed using a simple negative voltage threshold. A histogram is created of the peak voltage for each waveform and is fitted with a Gaussian. A special fitting function is used since the tail of the Gaussian distribution is assumed missing due to the detection threshold. The integral of the missing tail is returned as an estimate of the probability of false negatives.

In the case of multi-channel data, the waveform on each channel is normalized by the detection threshold on that channel. Then only the most negative value across all channels is saved for the histogram.

The Gaussian fit should be checked visually using **plot_detection_criterion**.

3.6.4 False negative estimate based on censored period (collisions)

Implemented by **ss_censored(spikes, clusterID).** Every detected event is followed by a brief "censored" or "shadow" period where no further spike can be detected. This feature is included in **ss_detect** so that a single spike event does not trigger multiple detection events. However, if events are detected at a high rate (> 50 Hz) then these shadow periods can become a significant percentage of the data set. This function calculate what percent of the experiment is censored by events outside of the given cluster, thus giving another false negative probability.

Note that "censored" events, "undetected" events, and the overall false negative probability estimated by Gaussian overlap are mutually exclusive. Therefore, these 3 estimates of false negative errors can be simply added to get a final estimate of false negative errors.

4 Data structure reference

This section describes the “spikes” object and all of its fields. This structure contains all waveform, timing, and assignment information for each detected spike as well as the parameters and scratch information used during the sorting process. Fields that are potentially large are set as “single” precision floating point numbers in order to save memory space. In the following text, symbols are used to represent the following values.

C = number of clusters

E = number of electrophysiology channels

N = number of detected spikes

S = number of samples in a waveform

4.1 Fields of the “spikes” object

Fields are added at different points during the spike sorting process. The spikes object contains the following fields.

assigns [1 x N] Array of integer cluster IDs for each spike event.

info Structure containing internal information automated spike sorting. See below.

labels [C x 2] Matrix of cluster labels. The first column contains the ID of a cluster. The second column contains the ID of the label for that cluster. Label IDs are defined in the **params** structure. See below.

params Structure containing parameters used for processing and display of spike data.

spiketimes [1 x N] Array of time of spike event within its trial (seconds).

trials [1 x N] Array of trial membership for each spike event.

unwrapped_times

[1 x N] Array of absolute time of spike events (seconds). The absolute time is approximated by concatenating trials and assuming a fixed time interval between each trial.

waveforms [N x S x E] Matrix of waveforms on each channel for each spike event.

4.2 Fields of spikes.params

The **params** field is created when **ss_default_params()** is called. See this function for default values.

4.2.1 Spike sorting parameters

These parameters are found as subfields of the **spikes.params** structure. They affect the procedure of spike sorting.

agg_cutoff Sets termination criterion for cluster aggregation. Higher values allow less aggregation. Lower values allow more aggregation.

cross_time Time within waveform window to place threshold crossing (ms). This value becomes the location of the peak after alignment.

detect_method Either “manual” or “auto”. If “auto” is used, then the **thresh** field is interpreted as the number of negative standard deviations to set the detection threshold for each channel. If “manual” is used, then **thresh** must contain the actual threshold values used on each channel

Fs Sampling rate of data (Hz).

kmeans_clustersize Target size for miniclusters. See **k-means** section above.

max_jitter Maximum amount by which a waveform can be shifted during alignment (ms).

refractory_period Period used to count refractory period violations (ms).

shadow Period after a threshold crossing until the next spike can be detected (ms). Used to avoid the same spike triggering multiple events.

thresh Determines the value of the negative-going threshold for spike detection. See **detect_method** above.

window_size Length of waveform window (ms).

4.2.2 Display parameters

These parameters are found as subfields of the **spikes.params.display** structure. They affect how the results of spike sorting are displayed.

4.2.2.1 Parameters used by plot_waveforms

cmap [Mx3] RGB color map used for 2D histograms. M can be any unsigned integer.

default_waveformmode

Default method for displaying waveforms. Choose 1 to see raw waveforms. Choose 2 to see bands representing 95% of waveform values. Choose 3 to see a 2D histogram.

time_scalebar Length of scale bar (ms).

4.2.2.2 Parameters used by plot_features

show_outliers Sets whether outliers should be displayed by default. (0 or 1)

xchoice Specifies what statistic is plotted as the x-axis value. Valid choices are **Signal, PC, Cluster, Minicluster, Event Time, ISI Preceding, Total Energy, Amplitude**, and **Width.**

xparam Specifies a parameter for the statistic specified by **xchoice**. See documentation for **plot_features**.

ychoice Similar to **xchoice** but for y-axis.

yparam Similar to **yparam** but for y-axis.

4.2.2.3 Parameters for plotting spike train correlations

correlations_bin_size Bin size used for cross- or auto-correlation plots (ms).

default_xcorr_mode Sets whether plot_xcorr initially shows the cross-correlation of the two spike trains (1) or the auto-correlation of the spike train formed when the two trains are merged (0).

isi_bin_size Bin size used to plot interspike interval histograms (ms)

max_autocorr_to_display Extent of x-axis for all correlation plots (s).

max_isi_to_display Extent of x-axis for interspike interval histograms (s).

show_isi Sets whether **plot_isi** initially shows the interspike interval histogram or the auto-correlation plot (0 or 1).

trial_spacing Sets the amount of time to pad between trials when concatenating spike trains for the field **spikes.unwrapped_times** (s).

4.2.2.4 Parameters used by plot_stability

max_scatter Maximum number of data points to show in scatter plot of spike amplitude.

stability_bin_size Bin size used to calculate firing rate over time (s).

4.2.2.5 Parameter used by outlier_tool

default_outlier_method Sets how the outlier tool determines the covariance matrix of a cluster. Use 1 to estimate the covariance matrix from the cluster itself. Use 2 to estimate it from the background noise.

4.2.2.6 Parameters for labeling clusters

label_categories Cell array of strings for possible labels that can be applied in **splitmerge_tool**. The first string is used as the default.

label_colors **[Mx3]** matrix of colors for the corresponding labels in **label_categories**. There must be one entry for each category.

4.2.2.7 Parameters for layout of tool figures

default_figure_size **[1x4]** Default figure position in 'normalized' coordinates.

figure_font_size Default font size used in figures.

initial_split_figure_panels Initial number of subclusters shown in **split_tool.** The aggregation tree for the cluster will be cut at a level so that this many subclusters can be viewed.

4.2.2.8 Parameters for color of tool figures

merge_fig_color **[1x3]** Background color of splitmerge_tool figure.

split_fig_color **[1x3]** Background color of split_tool figure.

outlier_fig_color **[1x3]** Background color of outlier_tool figure.

4.2.2.9 Parameters for layout of clusters within tools

aspect_ratio Ratio of height to width of plots in tools.

margin Number of pixels between adjacent plots (with no zoom).

outer_margin Number of pixels for figure margin.

width Number of pixels for width of a plot (with no zoom).

4.3 Fields of spikes.info

This structure is mainly for internal use by the spike sorting software. It contains detailed information about how the data was sorted. The structure **spikes.info** contains the following fields, listed in order of when they are added during the spike sorting process.

4.3.1 detect

Generated by **ss_detect**. Contains information relevant to extracting spikes from electrophysiology data.

align_sample Index of sample within waveform that corresponds to threshold crossing.

cov Estimated covariance matrix by sampling random data windows.

dur Array of duration of each trial (s).

event_channel Array of channel IDs for each waveform indicating which channel had the largest event.

stds Array of standard deviations for each channel.

thresh Array of negative thresholds used to detect events on each channel.

4.3.2 pca

Generated by **ss_detect**. Contains the principal components (SVD) of the waveforms.

s Diagonal matrix containing singular values.

u Matrix satisfies the equation **spikes.waveforms(:,:) = u*s*v'**

v Matrix containing the principal component vectors.

4.3.3 align

Generated by **ss_align**. Contains only a flag to show whether alignment was called.

aligned Set to 1 when alignment has been performed.

4.3.4 kmeans

Generated by **ss_kmeans**. Contains information used during the k-means algorithm. Importantly, the minicluster IDs for each spike event are stored here.

assigns Array of minicluster membership IDs for each spike event.

B Between-cluster scatter matrix.

centroids Matrix of mean waveform for each minicluster.

colors **[Mx3]** color matrix. Stores color for display of each minicluster.

iteration_count Number of iterations of k-means performed on each pass.

mse Mean-squared distance of waveforms from minicluster center.

num_clusters Number of miniclusters.

randn_state Stores random numbers used by kmeans algorithm.

T Total scatter matrix (**W**+**B**).

W Within-cluster scatter matrix.

4.3.5 interface_energy

[**MxM**] matrix generated by **ss_energy**. Each entry (j,k) specifies the interface energy between the j^{th} and k^{th} miniclusters. This matrix is no longer valid after outliers are removed or a minicluster is split.

4.3.6 tree

[**Mx2]** matrix logging the aggregation process. The first column gives the ID of the cluster that was merged into the cluster whose ID is given in the second column. The order of the merge operations is given by the row number.

4.3.7 outliers

Generated by **remove_outliers**. This structure contains bookkeeping information about each waveform that was removed as an outlier.

pca	Matrix of PCA components for each outlier event.
subassigns	Array of original minicluster membership for each outlier event.
spiketimes	Array of spiketime of each outlier event.
trials	Array of trial of each outlier event.
unwrapped_times	Arrray of absolute time, after trial concatenation, for each outlier event.
waveforms	Matrix of waveforms for each outlier event.

5 References

Fee, M. S., P. P. Mitra, et al. (1996). "Automatic sorting of multiple unit neuronal signals in the presence of anisotropic and non-Gaussian variability." J Neurosci Methods 69(2): 175-88.

Fee, M. S., P. P. Mitra, et al. (1996). "Variability of extracellular spike waveforms of cortical neurons." J Neurophysiol 76(6): 3823-33.

Hill, D.N., Kleinfeld D., Mehta, S.B. (2007). “Spike sorting.” In Observed Brain Dynamics by P.P. Mitra and H. Bokil. Oxford Press, 9:257-270.

Lewicki, M. S. (1998). "A review of methods for spike sorting: the detection and classification of neural action potentials." Network 9(4): R53-78.

Pouzat, C., O. Mazor, et al. (2002). "Using noise signature to optimize spike-sorting and to assess neuronal classification quality." J Neurosci Methods 122(1): 43-57.