Video Article

A Visual Guide to Sorting Electrophysiological Recordings Using 'SpikeSorter'

Nicholas V. Swindale[1], Catalin Mitelut[1], Timothy H. Murphy[2], Martin A. Spacek[1,3]

[1]Ophthalmology and Visual Sciences, University of British Columbia

[2]Department of Psychiatry, University of British Columbia

[3]Neurobiology, Biology II, LMU München

Correspondence to: Nicholas V. Swindale at swindale@mail.ubc.ca

URL: https://www.jove.com/video/55217
DOI: doi:10.3791/55217

Keywords: Neuroscience, Issue 120, electrophysiology, multi-electrode arrays, spike sorting, software, extracellular electrodes, polytrodes

Date Published: 2/10/2017

Citation: Swindale, N.V., Mitelut, C., Murphy, T.H., Spacek, M.A. A Visual Guide to Sorting Electrophysiological Recordings Using 'SpikeSorter'. *J. Vis. Exp.* (120), e55217, doi:10.3791/55217 (2017).

Abstract

Few stand-alone software applications are available for sorting spikes from recordings made with multi-electrode arrays. Ideally, an application should be user friendly with a graphical user interface, able to read data files in a variety of formats, and provide users with a flexible set of tools giving them the ability to detect and sort extracellular voltage waveforms from different units with some degree of reliability. Previously published spike sorting methods are now available in a software program, SpikeSorter, intended to provide electrophysiologists with a complete set of tools for sorting, starting from raw recorded data file and ending with the export of sorted spikes times. Procedures are automated to the extent this is currently possible. The article explains and illustrates the use of the program. A representative data file is opened, extracellular traces are filtered, events are detected and then clustered. A number of problems that commonly occur during sorting are illustrated, including the artefactual over-splitting of units due to the tendency of some units to fire spikes in pairs where the second spike is significantly smaller than the first, and over-splitting caused by slow variation in spike height over time encountered in some units. The accuracy of SpikeSorter's performance has been tested with surrogate ground truth data and found to be comparable to that of other algorithms in current development.

Video Link

The video component of this article can be found at https://www.jove.com/video/55217/

Introduction

Anyone who records extracellular signals from the brain using methods more sophisticated than simple on-line thresholding and windowing faces the task of identifying and separating the signals from different neurons from the noisy voltage signals recorded by the electrode. This task is commonly known as spike sorting. The difficulty of spike sorting is compounded by various factors. Neurons can be very close together so that the signals recorded from them by a nearby electrode are likely to be similar and hard to distinguish. The signals produced by a single neuron may vary over time, perhaps because of movements of the electrode, variable sodium channel kinetics during periods of high firing rate, variable degrees of activation of voltage conductances in dendrites that are close to the electrode, or possibly as a result of changes in brain state. These problems can be mitigated by using multi-electrode arrays (MEAs) with many closely spaced (20 - 100 µm) recording channels which allows better spatial definition of the signals from single neurons since they are typically spread out over several channels[1,2]. However, this, combined with the fact that signals from neurons spread along the entire length of the electrode overlap in space, results in a potentially very high dimensional space within which clusters corresponding to unique neurons need to be identified. This problem becomes computationally intractable for more than a small number of electrode channels. To date, there is no generally agreed-upon best method for spike sorting, though many solutions have been proposed[3,4,5,6,7,8] and recordings from MEAs are becoming increasingly common[9,10]. Because spike sorting is not an end in itself, but is simply a necessary preliminary step before further data analysis, there is a need for an easily usable package that will read in raw recording data files and convert them to sorted spike trains with as little user input, and as quickly and reliably, as possible.

This paper provides a tutorial for the use of SpikeSorter — a program developed with the aim of meeting these needs. The program is based on algorithms described in previously published papers[11,12,13]. The goals in designing the program were that *a*) it should have a user-friendly interface requiring little or no prior knowledge of computer programming or of spike sorting methodology; *b*) few or no other specialized software components beyond standard Windows or Linux operating systems should be needed; *c*) a wide range of recording data formats for data import and export should be supported; *d*) the need for user input during sorting should be minimized, and *e*) sorting times should scale in a reasonable way, ideally linearly, with recording duration and the number of channels on the electrode. The algorithms implemented in the program include *a*) a flexible set of pre-processing and event detection strategies; *b*) an automated divide and conquer strategy of dimension reduction which clusters voltage waveforms based on the principal components (PC) distributions obtained from subsets of channels assigned to specific clusters; *c*) automated clustering of PC distributions with a fast clustering procedure based on the mean-shift algorithm[3,14], and *d*) partially automated pairwise merging and splitting of clusters to ensure that each is as distinct as possible from all others. To this, a set of procedures has been added that allow manual splitting or merging of clusters based on inspection of PC distributions, cross- and auto-correlograms of spike

trains and time-amplitude plots of spike waveforms. Recordings from tetrodes, tetrode arrays, Utah arrays as well as single and multi-shank MEAs can be read and sorted. The current limit on the number of channels is 256 but this may be increased in future.

Another cross-platform open-source implementation, "spyke" (http://spyke.github.io), is also available. Written by one of us (MS) in Python and Cython, spyke uses the same overall approach as SpikeSorter, with some differences: to reduce memory demands, raw data is loaded in small blocks, and only when absolutely necessary; clusters are exclusively displayed, manipulated, and sorted in 3D; and principal component and independent component analysis are both used as complementary dimension reduction methods. Spyke requires more user interaction, but relies heavily on keyboard and mouse shortcuts and an undo/redo queue to rapidly explore the effects of various factors on the clustering of any given subset of spikes. These factors include spike channel and time range selection, spike alignment, clustering dimensions and spatial bandwidth (sigma)[11].

The following is a brief description of the algorithms and strategies used for sorting. More complete descriptions can be found in previous publications[11,12,13] and in annotations that can be accessed *via* help buttons (identified with a '?') within SpikeSorter. After loading a raw extracellular voltage file and filtering out the lower frequency components, an initial stage of event detection results in a set of events, each of which consists of a brief voltage snapshot before and after the event time. If the electrode sites are sufficiently closely spaced (< 100 µm), single unit signals will generally appear on several neighboring channels. A central channel is automatically chosen for each event, corresponding to the channel on which the peak-to-peak voltage of the event is largest. Automated sorting starts by forming a single initial cluster for each electrode channel, consisting of all the events that were localized to that channel. A unit located midway between channels may give rise to spikes that are localized (perhaps randomly) to different channels: the clusters from these two sets of spikes will be identified as similar and merged at a later stage. The average waveform of the events in each initial cluster is then calculated. This is referred to as the cluster template. Subsidiary channels are assigned to each cluster based on the amplitudes and standard deviation of the template waveforms on each channel. Principal component values are then calculated for each cluster based on the waveforms on the assigned set of channels. The user can choose the number of principal component dimensions to use: usually 2 is sufficient. Each cluster is then split into a further set of clusters, and this is repeated until none can be further split by automated clustering.

At this point, an initial set of say, 64 clusters from a 64-channel electrode, may be split into two or three times that number, depending on the number of units that was present in the recording. But because of the variable assignment of events from single units to different channels, the number of clusters found at this stage is almost certainly larger than it should be. The next stage of sorting is to correct the oversplitting by comparing pairs of clusters and merging similar pairs or reassigning events from one to another. This stage of sorting is referred to as 'merge and split'.

Merging and Splitting

For N clusters, there are $N*(N-1)/2$ pairs and hence the number of pairs grows as N^2, which is undesirable. However, many pairs can be excluded from the comparison because the two members of the pair are physically far apart. This reduces the dependence to something that is more linearly related to the number of channels. In spite of this shortcut, the merge and split stage can still be quite time consuming. It works in the following way. Each cluster pair that is to be compared (those that are physically close together, as judged by the overlap in the channel sets assigned to each) is temporarily merged, though keeping the identities of the spikes in the two member clusters known. The principal components of the merged pair are then calculated. A measure of the overlap between the points in the two clusters is calculated based on the distribution of the first two principal components.

The way the overlap measure is calculated is described in more detail elsewhere[11]. Its value is zero if the clusters do not overlap at all, *i.e.* the nearest neighbor of each point is in the same cluster. Its value is close to 1 if the clusters overlap completely, *i.e.* the probability of the nearest neighbor being in the same cluster is the same as that predicted from a uniform mixing of points.

Various decisions are made which take the overlap measure into account. If the overlap is greater than a certain value, clusters may be merged. If the overlap is very small, the cluster pair may be defined as distinct and left alone. Intermediate values, indicating incomplete separation of the cluster pair, may signal that the pair should be merged and then re-split, the desired outcome being a pair of clusters with less overlap. These procedures are run first in an automated stage and then in a manually guided stage.

In the automated stage, cluster pairs with a high overlap value are merged; then cluster pairs with intermediate to low overlap values are merged and re-split. In the second, user-guided stage, the user is presented with all the remaining ambiguous cluster pairs (*i.e.* those with overlap values in a defined intermediate range) in sequence and is asked to choose whether *a*) to merge the pair, *b*) merge and resplit the pair, *c*) to declare the pair to be distinct (which will override the significance of the overlap measure), or *d*) to define the relation between the pair as 'ambiguous' indicating that the spikes in the pair are unlikely to be well sorted. Various tools are provided to help with these decisions, including auto- and cross-correlograms and time series plots of spike height and PC values.

Ideally, at the end of the merging and splitting stages, every cluster should be distinct from all others, either because it has few or no channels in common with other clusters, or because the overlap index is less than a defined value. This value is user-selectable but is typically 0.1. Clusters (units) that pass this test are defined as 'stable', those that do not (because the overlap with one or more other clusters is greater than the threshold) are defined as 'unstable'. In practice, the great majority of units end up being defined as 'stable' at the finish of sorting, leaving the remainder to either be discarded or treated as potentially multi-unit.

Software Requirements

SpikeSorter is compatible with 64 bit versions of Windows 7 and Windows 10, and has also been run successfully under Linux using the Wine emulator. Data files are loaded completely into memory (for speed) hence available RAM needs to scale with the size of the recording (allow about 2 GB for the program itself). Electrophysiological data files larger than 130 GB in size have been successfully sorted in both Windows and Linux environments. Options are accessed through standard Windows menus, a toolbar and dialogs. The layout of items on the menu matches roughly the order of operations in sorting, beginning with the 'File' menu on the left for data input and the 'Export' menu on the right allowing for export of sorted data. Toolbar buttons provide shortcuts to commonly used menu items.

The Channel Configuration File

Many recording data formats do not store channel locations. However, knowing these is essential for spike sorting. Channels may also be numbered in various ways by acquisition software: SpikeSorter requires that channels are numbered in sequence, beginning with channel 1. Thus, an ancillary electrode configuration file has to be created that can remap channel numbers to follow the sequential rule, and to store channel locations. The channel configuration file is a text file with a single row of text for each channel. The first line of the file stores a text name, up to 16 characters long, that identifies the electrode. The numbers in subsequent lines can be separated by tabs, a single comma, or spaces. There are four numbers in each row providing (in order): the channel number in the file, the channel number to which it is to be mapped (*i.e.* the number that will be used by SpikeSorter), and the *x* and *y* coordinates of the channel, in microns. The *x* coordinate would normally be taken as perpendicular to the direction of electrode insertion and the *y* coordinate accordingly would be depth into the tissue. The configuration file has to be placed in the same directory as the recording file. There is some flexibility in how it can be named. The program will first search for a file that has the same name as the raw data file but with a .cfg extension. If that file is not found, it will search for the file 'electrode.cfg'. If that file in turn is not found an error message is generated to indicate a lack of channel layout information.

Protocol

1. Program Setup

1. Go to http://www.swindale.ecc.ubc.ca/SpikeSorter to download the program. Copy the supplied executable file to the directory of your choice. Read the accompanying documentation.
   NOTE: No formal installation or compilation is required.
2. Before opening any file to be sorted, ensure that there is enough free RAM to contain the entire duration of the recording. Also make sure a valid channel configuration file, as described in the documentation, is present in the same directory as the data file.
3. Start the program, then go to 'File - Open' and select the recording file format from the drop-down list at the bottom right of the resulting open file dialog. Select the file to be opened, then click 'Open'.
4. Once reading is complete, inspect the voltage recording display. Double click on the display (or go to 'View - Voltage Record') to bring up a dialog with controls that allow any part of the recording waveform to be viewed.
   NOTE: Double clicking on other display windows will often bring up associated dialogs.
5. After the dialog is exited, hover the mouse over the waveforms to display particular voltage values in the top left corner of the display. Use the scroll wheel to zoom in on any part of the display. Hold down the left mouse button to drag the window contents.
   NOTE: This display is frequently updated to reflect the addition of newly detected events, or to indicate, by means of colors and/or numbers, their cluster assignments after clustering.
6. If the recording is unfiltered and contains the local field potential, remove it by going to 'Pre-Process - Transform/filter' (or click on the filter icon in the toolbar). Select 'High-pass Butterworth Filter', then a suitable cut-off frequency and the number of poles, and then press 'Do-It!'. Once filtering is finished, inspect the new waveform in the voltage waveform window.
   NOTE: Filtering is done in the Fourier domain, is non-causal, and does not introduce phase distortion of the waveforms. For a long recording, filtering may take several minutes.
7. Next, check for channels that may be faulty and need to be masked. Go to 'Pre-process - Channel check' (or click on the channel check icon) and then inspect the graph that appears. The graph shows the change in signal correlation between channel pairs as a function of their spatial separation[5]. Channels that violate this relation may not be functioning properly. To see any such outliers, click on 'single channel net deviations'.
   1. To mask an outlying channel either select the channel number, or select it from the problem list. When this dialog is exited, click on 'Yes' at the prompt to save the mask values.
      NOTE: This file will have the same name as the recording data file but with the extension .msk. It will be read automatically whenever the same data file is opened.

2. Event Detection

1. Go to 'Pre-process - Event Detection' to bring up the event detection dialog (**Figure 1**). This dialog also offers the option of masking channels based on their noise levels (though these will often be detected by the previous tests). For example, a channel that has been intentionally grounded may have a very low noise level.
2. Use the slider on the top right to inspect the noise level on particular channels. Careful inspection of the voltage display may also reveal silent or unusually noisy channels that need to be masked.
3. Choose a thresholding method for event detection. Use the help button in the group box for more information about the options. 'Variable' thresholding, with a threshold of 4.5X - 6X noise[7], is recommended. Use the controls on the top left to choose how the noise level is calculated for this purpose.
4. Choose the detection method from the drop-down list. 'Dynamic multiphasic filter' is the recommended method. This requires specification of a temporal window. Set the window to be roughly half the width of a typical spike. Very narrow values will bias detection to narrower spikes though the effect is not large. Values in the range 0.15 - 0.5 ms are recommended[12].
   NOTE: The values displayed are in integer multiples of the sampling interval (reciprocal of the sampling frequency).
5. Select the alignment method. Choose the option that best identifies a single, temporally localized feature of the spikes that are being sorted, *e.g.*, a 'positive peak' may be a poor choice if many spikes have more than one positive peak. For many recordings, a 'negative trough' will be the best choice. Other options can usually be left at their default values. Press 'Start'.
   NOTE: Event detection may take from several seconds to several minutes, depending on the length of the recording and the number of channels.
6. Press 'Done' to exit the dialog. Inspect the events, shown in grey, in the voltage waveform window. Check that signals that look like events have been detected.

1. If not, consider re-running event detection with a lower detection threshold. Beware however that very low amplitude spikes may be difficult to sort and that large numbers of them may impede sorting of larger amplitude spikes. Also check for obvious duplicates or a failure to resolve nearby spikes and adjust spatio-temporal lockout window parameters accordingly.
   NOTE: At this stage events are identified by their times of occurrence and a channel number. Normally this is the channel on which the peak-to-peak amplitude of the spike is largest. The events are unclustered initially, so each has a cluster assignment of zero.

3. Sorting

NOTE: The next step is not normally performed before routine sorting, but it is very useful to do it when sorting for the first time, or when encountering unfamiliar data.

1. Go to 'Sort - Convert channels to clusters'. This creates a single cluster for each unmasked electrode channel, assuming that each channel has some events assigned to it. Examine these clusters by going to 'Review - View Clean and Split clusters'. This brings up another dialog (**Figure 2**). Use the spin control (top left) to select the cluster to be viewed.
   NOTE: The solid blue (cyan) line is the average of all the waveforms in the cluster and is referred to as the cluster template in what follows. The principal components (PC) distribution of the events in the cluster are shown in the window below. These will often reveal the presence of two or more subclusters.
2. Press the 'realign' button to change the time of each event (resulting in small sideways shifts of the waveforms in the display) so as to better match it to the shape of the template, doing this often makes subclusters more compact and distinct, and sometimes reduces the apparent number (**Figure 3**).
3. Select a cluster that has two or more distinct subclusters and press 'Autosplit'. If subclusters are identified in the PC display, they will be colored. As an exercise, use one of the small 'split' buttons to create a new cluster and examine it. Sorting could continue manually this way, but instead go back and use the faster autosort procedure.
4. Go to 'Sort - Autosort' (or press the autosort button on the toolbar) to begin automatic sorting. The resulting dialog is shown in **Figure 4**. It presents a variety of options.
   1. Leave the 'skip event detection' option checked if event detection has already been done. If it is not checked, event detection will be run using parameter values and choices inherited from the event detection dialog. Since event detection has already been done, leave that option checked.
   2. In the 'clustering' panel below, select a temporal window large enough to contain the entirety of the spike waveform preceding and following the alignment point, but no more. Use this window to block out regions of the spike waveform, *e.g.* long variable afterpotentials, if they appear to be interfering with (or contributing little to) sorting. Usually values in the range ± 0.5 ms are appropriate. Like other temporal windows, the window is an integral number of sample points, so the temporal values that appear are multiples of the sampling interval.
   3. Next, select a realignment option to be used during clustering. This will make use of the template waveform and works more robustly than in the initial case of event detection where the criterion has to be applied to relatively noisy individual spike waveforms. The recommended option is 'peak-weighted c.o.g.' but 'negative trough' may be better if that is a consistent feature of the spike waveforms.
   4. Choose a minimum cluster size. Clusters with less than that number of spikes will be deleted, preventing the accumulation of large numbers of small, possibly spurious, clusters during sorting.
   5. Decide on the number of dimensions in PC space that will be used for clustering. Two is generally adequate but slightly better results may be obtained with 3, albeit with a longer sorting time.
   6. Leave the other options at their default settings. Use the Help buttons to get more detailed explanations of the various options.
5. Press 'Start' to begin the autosort. Channel based clusters are first formed as illustrated in step 3.1. These are now processed in turn, forming new clusters by splitting off individual sub-clusters, one at a time. Each time a new cluster is split off, the PC values are recalculated and displayed. This continues until no individual cluster can be further split.
6. Follow the prompts in the display, where the subcluster that will be split off from the parent cluster is shown in red.
   NOTE: Occasionally the final cluster is red with uncolored outliers that do not form a distinct subcluster. These outliers will usually be deleted. During this process the number of clusters gradually increases. When it is finished, cluster overlap indices are calculated for every eligible cluster pair. Pairs that have large overlap values are automatically merged, while pairs that have intermediate overlap values (the default range is 0.1 to 0.5) are merged and then resplit. Intermediate values suggest that there are two distinct clusters but that some points are misassigned. During this stage the number of clusters typically decreases and the number of stable clusters increases.

4. Customization

1. If using the program for the first time (or possibly during the next step), customize window sizes and positions. Go to 'File-Preferences'. Choose sizes for the various windows by selecting the window type from the pull-down list and adjusting the size to suit the screen. Exit the dialog and position the windows to make best use of the screen.
2. From the dialog, choose scaling values that best suit the layout and spacing of the channels on the electrode and the spikes in the recording. There is an autoscaling option but this may not always choose the best values. Turn it off if it does not.
3. Check the Sticky Parameters option: if the option is selected, changes in sorting parameter values (*e.g.* as used in event detection) will be saved and inherited next time the program starts. This can be useful but also requires that parameter values be checked to ensure they have not been carelessly changed as various options are explored or as a result of reading in different work files. Options for changing sub-cluster colors are also available.
4. Exercise care changing the number of processor threads. The optimal number is usually 1 less than the number of physical (not virtual) CPU cores. Increasing the number of threads may not speed up processing and can even result in a severe slowdown.

5. Merge and Split

1. After the autosort is complete, press 'Next' to go to the manually guided merge and split stage. The resulting dialog shows, in the lower left corner, the number of remaining ambiguous cluster pairs that need to be examined as well as the number of stable clusters.
2. Press 'Begin'. Another dialog appears together with the first of the pairs to be examined.
3. Choose whether to merge the pair, resplit it (resulting in a lower overlap value), to label the pair as 'distinct', meaning that the value of the overlap index will be ignored, or to label the pair as 'ambiguous', meaning that it is considered uncertain whether the spikes are from the same or different units.
    1. Click on the check boxes to display a graph of spike parameters (peak-to-peak (P-P) height, or the first (PC1) or second (PC2) of the principal components) *vs.* time, and/or auto and cross-correlation histograms.
    NOTE: The display of P-P height *vs.* time is often very useful in deciding whether to merge two clusters. If the heights of the spikes in one unit blend smoothly into those of another at the same time that one unit stops firing and the other begins it is much more likely than not that they are the same unit and should be merged. Cross-correlograms may reveal a strong temporal relationship between the spike times in two clusters. If the cross-correlogram has a strong, asymmetric peak at a very short time interval (*e.g.* around 5 - 10 ms) and especially if the second spike is smaller than the first, the two units are most likely a single unit which is firing spike pairs in which the second is smaller than the first because of Na^+ channel adaptation.
    2. In cases where the decision to merge is not easy, label the pair as 'ambiguous' and treat the clusters accordingly in subsequent analyses.

4. If the merge and split option is unable to find clearly separable clusters, use the slider in the prompted dialog to manually vary a clustering parameter (a spatial bandwidth, sigma), together with the set of merging buttons, to find a split that looks satisfactory. Use the 'Revert' button to go back to the original state of the two clusters. Press 'Split as shown' to finish. Note more than two clusters can be produced by this procedure.
5. Continue with this process until there are no more pairs to inspect. The great majority of clusters should now be listed as 'stable'.
6. If some cluster pairs have very low overlap indices, so that they are ignored by the guided merge (but there is still evidence for merging them), go to the 'Review - Compare cluster pairs' menu option (or click on the associated icon in the toolbar) and open the dialog shown in **Figure 5**. Use the spin controls at the top of the dialog to select any pair of clusters for comparison.
    NOTE: As with the guided merge and split, pairs are put into a sorted list, but in this case comparison metrics additional to the cluster overlap index are available.
    1. Select the 'normalized dot product' option from the pull-down list. This calculates the correlation between the template values. It is insensitive to multiplicative scaling variations and is well suited to picking out cluster pairs that are an artefactual result of peak-to-peak height variability.
    2. Press the 'Most similar' button in the middle of the dialog to display the most similar pair. Use the horizontal spin control beneath the button to go forward or backward through the list. Use the correlation display and the P-P height *vs.* time display to make merging decisions, just as for the user guided merge and split. Note that the list is recalculated after each merging operation. This comparison stage is open ended, and it is up to the user to decide how extensively to search for evidence in favor of merges.

6. Review – Post-processing

1. Now go to 'Review - Post-processing' (or click the appropriate toolbar icon). This dialog (**Figure 6**) offers options to add or remove events from clusters, as well as the option of deleting entire clusters with signal-to-noise ratios (SNRs) that fall below a threshold. Duplicate events (events occurring at the same time in a cluster) can be created by alignment errors during sorting. Events that are a long way removed from their original location can sometimes be relocated; they can also be removed when relocation does not work.
2. Use the alignment cleaning button to remove events from clusters that are a bad match to the template. Use the 'Recluster' button to do the reverse, *i.e.* to reassign unclustered events that are a good match to a particular template. The reclaimed events are marked as a subcluster of each parent cluster and can be inspected using the 'View, clean and split clusters' dialog. These events will remain in the cluster (and be exported as such) unless they are deleted (use the small 'delete' button for the first subcluster). Returning to the post-processing dialog, use the 'delete' button and the spin control next to it to delete clusters with a SNR less than the selected threshold.
3. Although cluster numbers go continuously from 1 to *N*, where *N* is the total number of clusters, the actual numbering of clusters at the end of sorting is close to arbitrary. Use the 'Sort' button to renumber the clusters according to a chosen criterion, *e.g.* vertical position on the electrode, or channel number. Note that, with the exception of the deletion of duplicate events, there is currently no objective evidence to support particular choices in this dialog as being better than others.
4. At any stage during the manual procedures of the sort it is possible to save a file which contains current parameter values, sorting options, event times, cluster properties and the message record. Create this file by going to 'File - Save work file'. Give the file a name that is clearly related to that of the data file and press 'Save'. Resume sorting at a later time by first opening the original recording file, followed by high-pass filtering (if done initially). Then, open the saved work file. The program will then be in a state identical to the one it was in when the work file was saved. The work file is also a record of how the sorting was done - the parameters used and of the messages issued during sorting.
5. Finally, export the clustered events. Go to 'Export - Sorted spike files' (or click on the relevant button the toolbar). Select '.csv file' (comma separated variable) from the dropdown list then click on 'Save as'. Choose a name for the file that will contain the exported csv data for the sorted units.
    NOTE: This text file will have a single line for each event containing, in order, the time of the event (in seconds to the nearest 10 µs), the cluster number (from 1 upwards) and the number of the channel that was assigned to the event. Note that the assigned channel may not be the same for all the events in a cluster if the events were not consistently larger on one particular channel.

Representative Results

Figure 7 shows the display (obtained by going to 'View - Sorted waveforms') for a typical sorted recording. The default view option is just to show the waveforms on the center channel for each cluster. A common experience is that waveforms for a cluster pair on the same channel look identical, but when the 'Compare pairs' dialog is used to examine the two clusters there are distinct clusters in the PC projection, most often resulting from waveform differences on adjacent channels. This is true, for example, of the waveforms on channel 62 in **Figure 7**.

As noted above, it is not uncommon to find cluster pairs where merging decisions must be based on amplitude-time plots and on cross-correlograms. **Figure 8** shows an example of a merging decision based in part on the cross-correlogram. A very strong, asymmetric cross-correlation at short time intervals (**Figure 8B**) coupled with a difference in the peak-to-peak height of the units and similar firing patterns (**Figure 8E**) strongly suggests that the spikes come from the same neuron. **Figure 9** shows a case where the same kind of evidence for merging is lacking. Here, the cross-correlogram is weak and not strongly asymmetric. In addition, the shapes of the autocorrelograms of the two clusters are different (**Figure 9A**). Arguably, the two units should not be merged because of the additional clear difference in the distributions of the principal components (**Figure 9C**). **Figure 10** shows a case where the P-P heights of two units blend together at the same time that one of them stops firing and the other resumes. In this case the decision to merge seems correct, though one cannot rule out the possibility that units coordinate their firing patterns in complex ways and that the similarity in heights is accidental.

These examples illustrate the difficulty in offering firm guidance on how to make merging decisions. This is compounded by the general lack of objective measures for assessing the overall quality of spike sorting and the effects of parameter changes. This is because of the lack of ground truth information, which, for spike sorting, would consist of intracellular recordings (or their equivalent) from every neuron that was close enough to a recording electrode to give rise to detectable extracellular signals. Despite this limitation, there are surrogates for ground truth data and it is not unreasonable to suppose that a change in sorting strategy that results in better performance on surrogate data will lead to better performance with real data. The surrogates include real MEA recording data in which spikes, taken from the recording, are added back into the recording at known times on different channels, where they cannot be confused with the original spikes. Such a test formed the basis of a spike sorting competition organized by G. Buzsáki and T. Harris held at Janelia Farm in 2013. Surrogate data were generated from recordings made in the thalamus or hippocampus of freely moving rats (A. Peyrache, A. Berenyi and G. Buzsáki, unpublished data). Spike signals for which there was 'ground truth' were generated by taking spikes from a unit recorded on one shank and adding them to the recording on another shank thus ensuring that the relationship of that spike train with background activity and brain states was preserved. Recordings contained actual spiking activity in addition to the added ground truth spike trains. The false positive rates for SpikeSorter were 0.26% and 0.01% for two different test sets while the corresponding false negative rates were 2.1% and 0.37% (A. Peyrache, personal communication). These rates were among the best of the competition but more importantly they are low and probably acceptable for most types of neurophysiological analysis. Another approach is to use highly detailed large scale biophysical simulations of networks of neurons to generate simulated extracellular recordings from specified MEA designs. Researchers currently working on MEA sorting methods were invited to sort test simulations of this nature[15]. Five different sorting algorithms were compared. There are various ways of evaluating sorting performance and the performance of the different groups varied according to which measures were used, with no one group being obviously better than any other. SpikeSorter's results fell within the range of results obtained by the various groups.

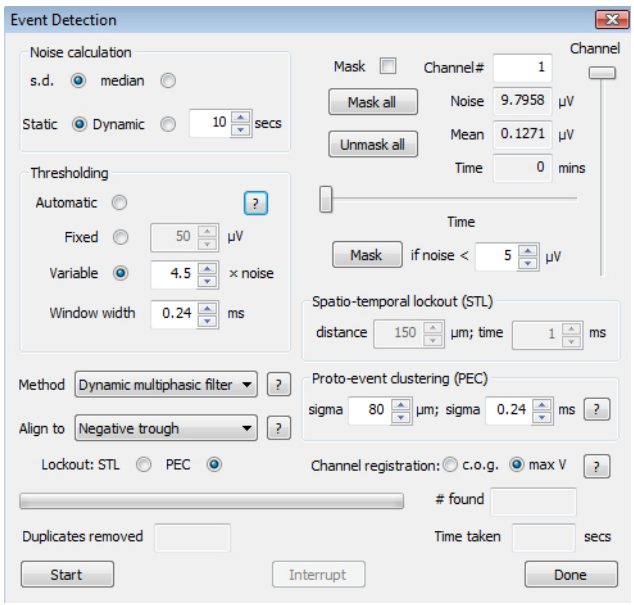

Figure 1. The Event Detection Dialog. This provides options for selecting the method of noise measurement, for masking channels, setting thresholding values and methods of applying them, and for choosing methods for avoiding event duplication. In this and other dialogs, information about the choices is provided by buttons identified by question marks ('?').

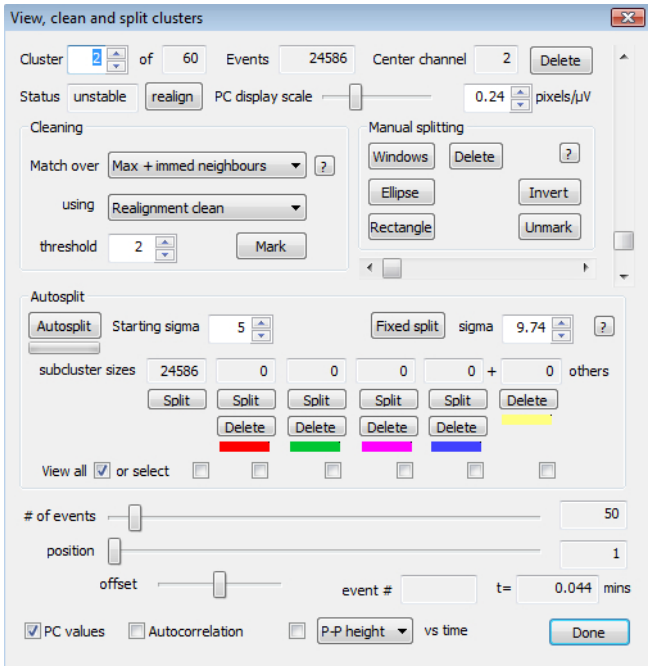

Figure 2. The View, Clean and Split Dialog. This provides options for viewing cluster waveforms, identifying and deleting outlying waveforms, for splitting clusters into one or more subclusters, and for deleting or forming new clusters from the subclusters. Subclusters are identified by the colors shown. (These can be changed in the Preferences dialog.)

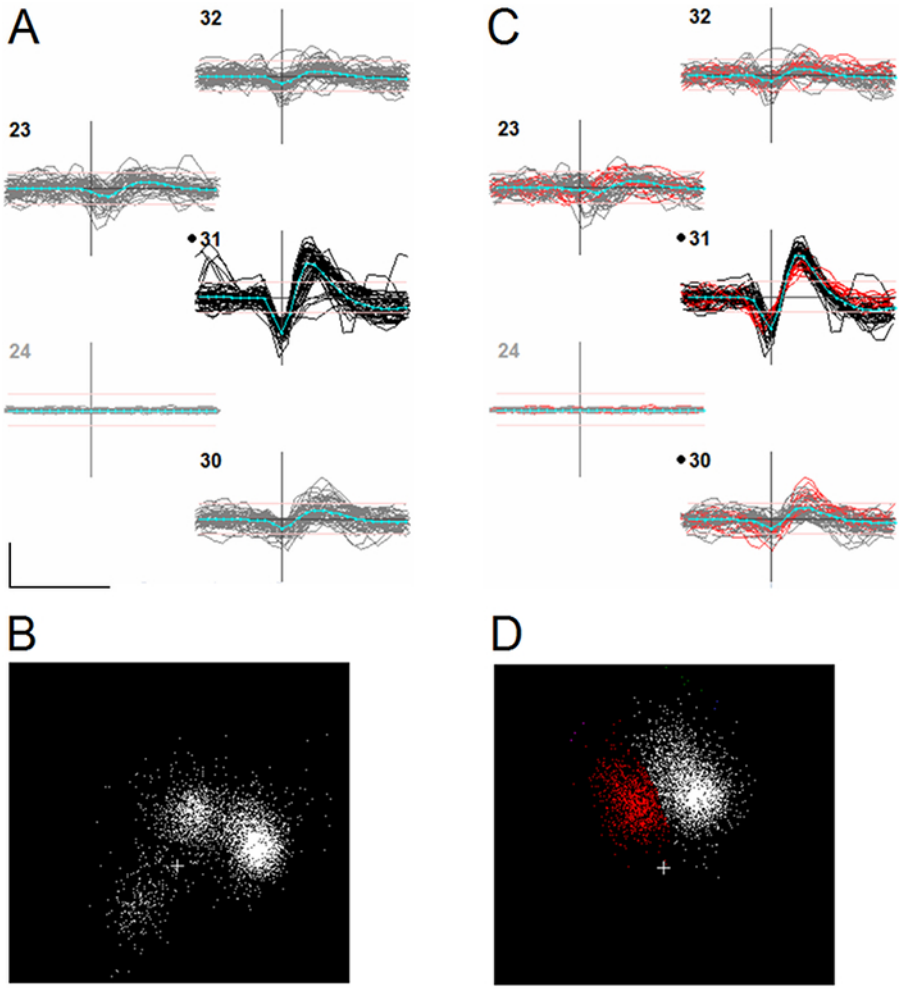

Figure 3. Effect of Aligning Events to an Unreliable Feature. The figure shows data from a single channel-based cluster, defined as the set of events whose peak-to-peak waveform voltages were largest on a particular channel. **Panel A** shows a subset of 50 event waveforms from this cluster, overplotted, on different electrode channels. Channel numbers are shown in the upper left corner of each set of waveforms. Black dots next to a channel number indicate that the channel has been assigned to that particular cluster. Channels are laid out in the same spatial order that they have on the electrode. Horizontal axes show time and vertical axes, voltage. The horizontal position of the vertical axis indicates the alignment point, *i.e.* each event is positioned so its alignment point coincides with the axis. The scalebar at the bottom left of panel A shows 0.5 ms and 100 µV. Blue lines in A indicate the average of each set of waveforms (the template). Channel 24 (greyed out) is masked. Events are aligned to the most negative local minimum of the waveform (negative trough) as determined immediately following event detection. **Panel B** shows the distribution of the first 2 principal components derived from all the waveforms in the cluster. Three subclusters are visible in this distribution. **Panel C** shows the same set of events after aligning them to the template waveform. The principal components distribution (**panel D**) now shows only two subclusters (one identified in red). Further examination showed that a spurious cluster in B was caused by the alignment of a subset of events to a second negative trough (the slower negative after-potential) which in some cases was more negative than the first one. Some of these misaligned events are visible in panel A as waveforms whose shape does not match the rest. Please click here to view a larger version of this figure.

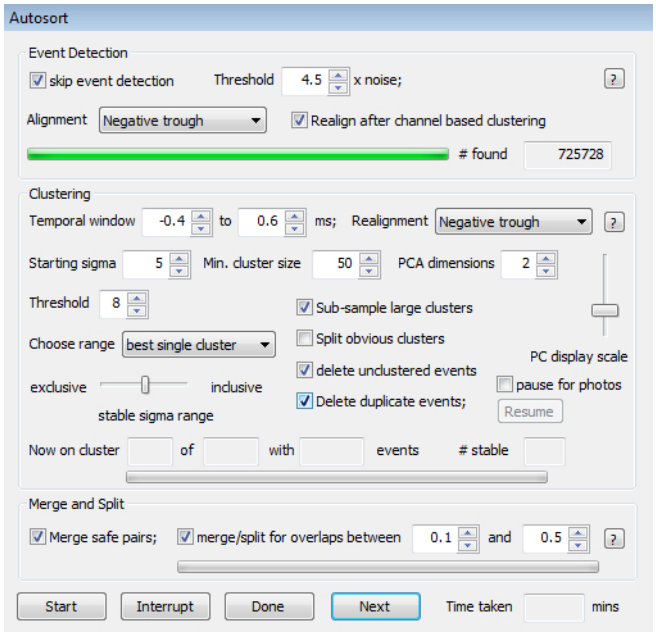

Figure 4. The Autosort Dialog. This provides event detection options, clustering options and options for automated merging and splitting of cluster pairs following the initial automated clustering stage.

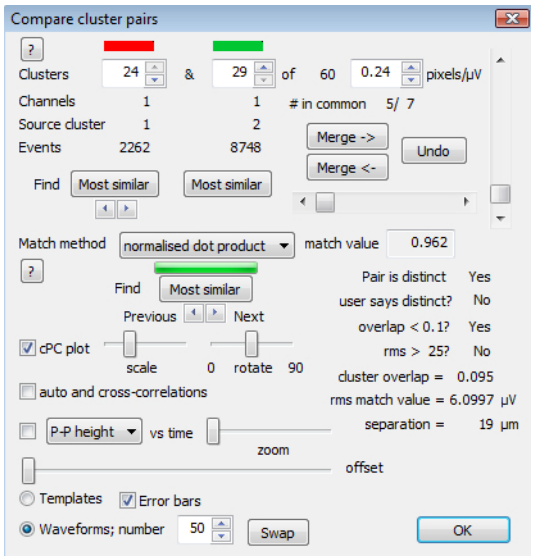

Figure 5. The Compare Cluster Pairs Dialog. This provides options for choosing cluster pairs, comparison measures (Match method), searching through lists of pairs ordered by the comparison value, options for displaying correlograms, displaying plots of P-P height (or PC1 or PC2) *vs.* time, and an option to merge pairs.

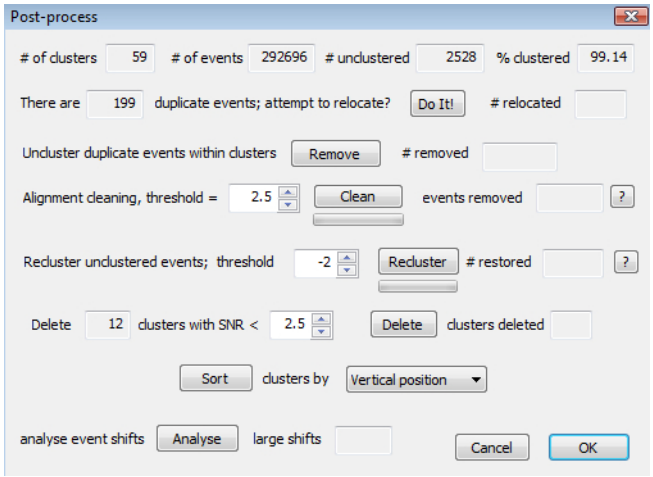

Figure 6. The Post-processing Dialog. This provides options for relocating and/or deleting duplicate events, for deleting possibly noisy events, for reclustering unclustered events, for deleting clusters with a low signal to noise ratio (SNR) and for renumbering (sorting) clusters according to different criteria.

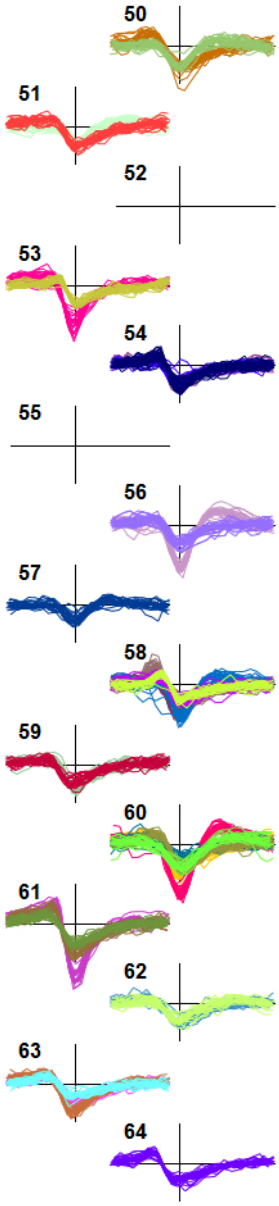

Figure 7. Display of Sorted Units Showing Randomly Chosen, Overplotted Waveforms Colored According to Cluster Number. For clarity, only the center channel waveform of each cluster is shown. Data (from Mitelut & Murphy, unpublished) show the lower 14 channels of a 64-channel electrode recording from mouse visual cortex.

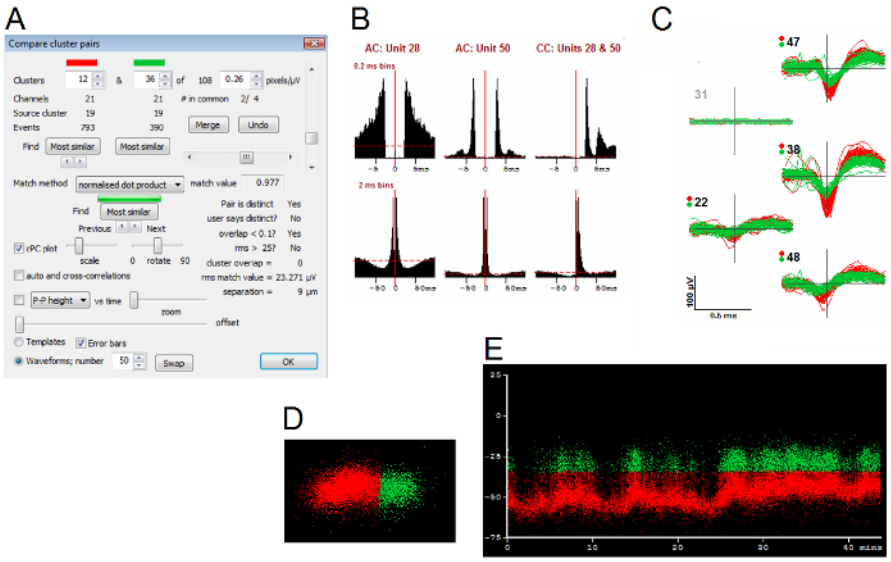

Figure 8. Example of Evidence That Can Be Brought to Bear on a Decision Whether or Not to Merge Two Clusters. The Compare clusters dialog (**Panel A**) was used to search for cluster pairs with similar waveform shapes, ignoring amplitude (normalized dot-product match method). **Panel B** shows autocorrelograms (AC) and the cross-correlogram (CC) for the two clusters, with two different bin widths (0.2 and 2 ms). These show that spikes in the second cluster (unit 53) have a very strong tendency to occur either 4 or 8 ms before spikes in the first (unit 28). **Panel C** shows the spike shapes of the two units and also shows that the second (shown in green) has a smaller spike than the first. **Panel D** shows the PC distribution of the two clusters. **Panel E** graphs PC1 (vertical axis) of the two units of the two units (red and green respectively) *vs.* time (shown in minutes) during the entire period of recording. See text for further description. Please click here to view a larger version of this figure.

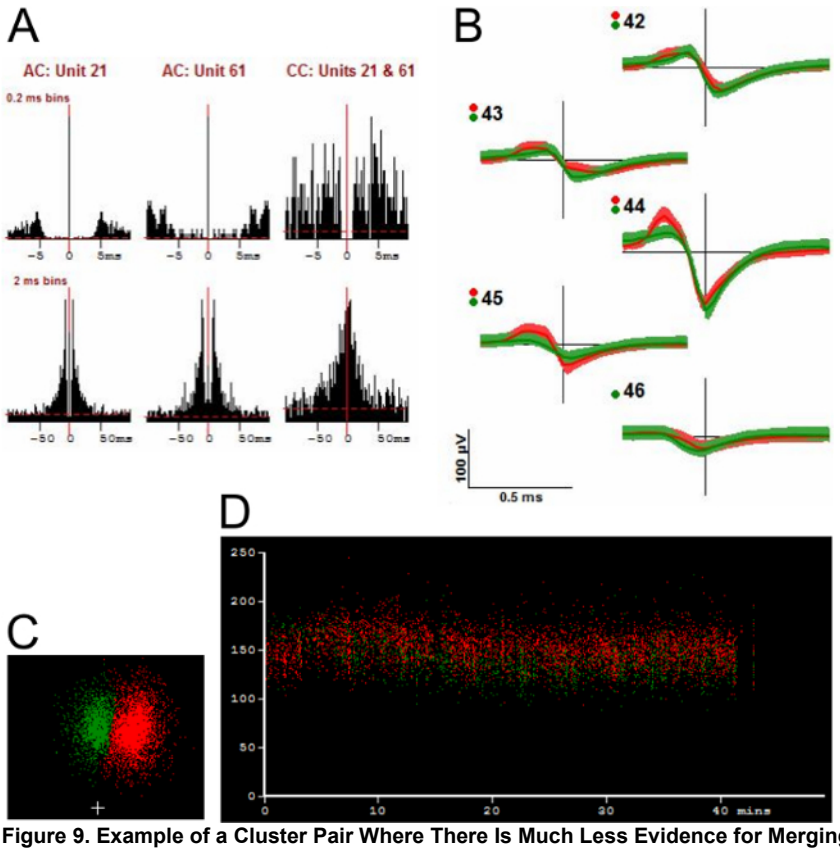

Figure 9. Example of a Cluster Pair Where There Is Much Less Evidence for Merging. Panel A shows that the autocorrelograms (AC) and the cross-correlogram (CC) for the two clusters have different shapes. **Panel B** shows the averaged template waveform and standard deviations (shading indicates 1 S.D. unit) in order to show the differences in waveform shape more clearly. **Panel C** shows the PC distribution of the two clusters. **Panel D** graphs the peak-to-peak height (vertical axis, µV) of the two units vs. time during the entire period of recording. See text for further description. Please click here to view a larger version of this figure.

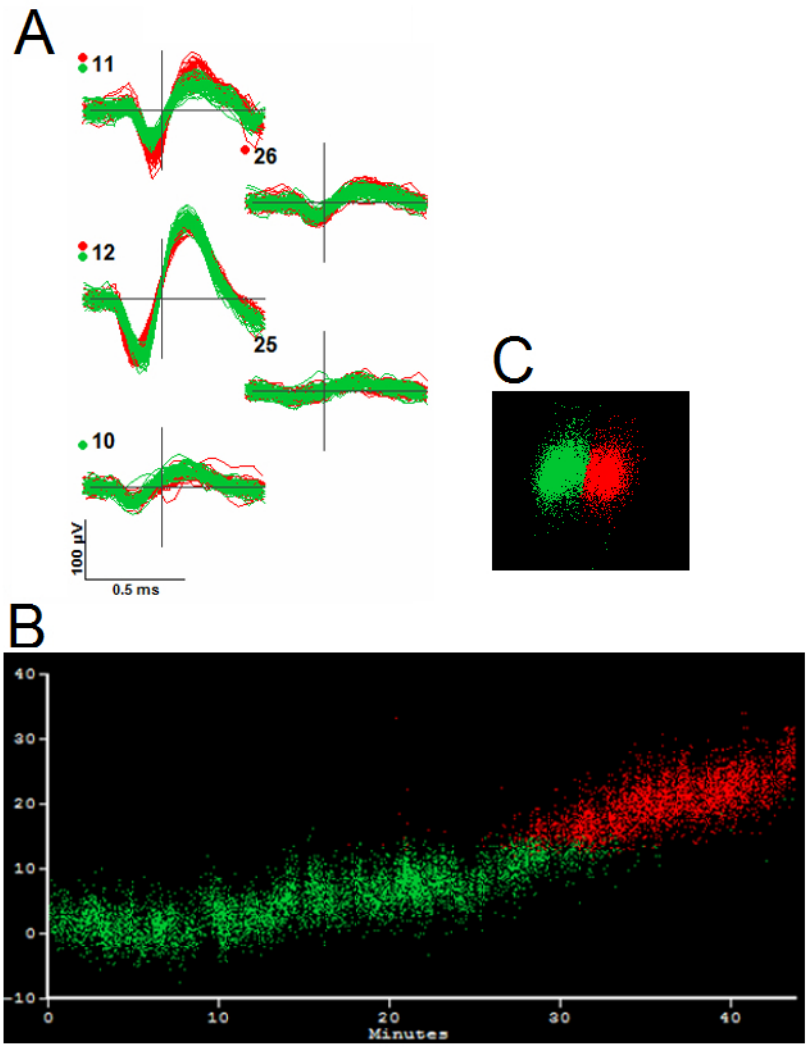

Figure 10. Evidence for Merging Based on Firing Pattern and Principal Components Variation. Panel A shows the waveforms of the two clusters (red and green). **Panel B** plots PC1 (vertical axis) vs. time (horizontal axis) for the two clusters and shows a complementary pattern of firing with PC1 values being similar at the time one unit stops firing and the other starts. This supports a decision to merge despite the presence of distinct clusters in the PC distributions (**Panel C**).

Discussion

File Formats

Currently supported file formats include Neuralynx (.ntt and .ncs), Plexon (.plx), Neuroscope (.xml + .dat), MultiChannel Systems (.mcd), Blackrock (.nev) and Intan (.rhd). For unsupported formats, there are two options. One is to request addition of the file format to an upcoming release (an email link to the developer is provided in the 'Help - About' dialog). The other is to convert the file to a supported format. A simple option is to use the time-spike-format '.tsf'. This bare bones format contains the voltage record and channel location data plus the record of events and the channel and cluster assignments following sorting. Reading these files is often faster than for other formats. Independently of dealing with unsupported formats it may be convenient to save filtered data in a .tsf file (this format is included among the Export options) since this will avoid the need for subsequent time-consuming filtering. Details of the .tsf format are included in the documentation that comes with the program.

Ancillary Files

Two ancillary files are used to store parameters, ss_prefs.sav and ss_parameters.sav. The file 'ss_prefs.sav' saves only user-chosen values that have no direct effect on sorting and are less likely to need changing, *e.g.* window sizes and positions, voltage and other scaling values. If ss_prefs.sav does not exist, it is created when the 'Apply' button in the relevant dialog is pressed, or when the program is exited. If the 'sticky parameters' option in this file is set, a separate file 'ss_parameters.sav' is used to store user-selectable parameter values and options that affect the outcome of sorting as well as many display options. This file is saved, or updated, when the program is exited *via* the normal 'File - Exit' route (but not when the program 'Close' button (top right) is used). In the absence of this file on startup, default values are used.

Program Limits

The limit to the length of recording that can be sorted is determined by the amount of RAM on the computer. A PC with 16 GB of RAM can generally handle raw files of up to 13 GB-14 GB in size (2 GB less than the total RAM) if memory is not in use for other purposes. Other limits, *e.g.* on the maximum number of channels, maximum cluster sizes, *etc.* may vary with program version and future upgrades. They can be viewed by going to 'Help - About'.

Additional Features

The View, clean and split clusters dialog offers several options for manual definition of cluster boundaries. They include use of the mouse to draw an ellipse in the PC display window, to draw a rectangle in the P-P amplitude (or PC1 or PC2) *vs.* time display, and to draw discrimination windows in the main waveform display. These can each be used to create subclusters (any existing subclustering will be overwritten). The dialog needs to be exited before any of these objects can be drawn. Pressing the related button on the dialog ('Windows', 'Ellipse' or 'Rectangle') creates the subcluster.

A strategy dialog (Sort - Strategy) displays a variety of sorting parameters that are less likely to need to be changed but which can have a significant effect on sorting. These include, for example, parameters that determine the assignment of channels to clusters and the selection of time points that contribute to the calculation of principal components for any cluster. The Manage clusters dialog offers more detailed information about individual clusters than is provided by the View, clean and split clusters dialog, or by the Post-Processing dialog. There are also more varied options for deleting clusters.

Channels are displayed in a particular vertical order, termed the 'sort order', in the voltage display window. Ideally this order will reflect the physical proximity of the channels but this may be hard to achieve given that the actual layout is in two dimensions. The sort order is generated by calculating the projection of the channel positions onto a line with a given angle relative to the *y*-axis. The sequence of channel numbers on the line is the sort order. This is calculated automatically in many cases but it is possible to generate a different one by going to 'View - Acquisition properties'. The option to display channels in numerical order is also provided. Note that the order of display has no effect on sorting.

Other Approaches

Other software packages for doing spike sorting exist. These include commercial programs such as Offline Sorter (http://www.plexon.com/products/offline-sorter), as well as free software such as MClust (A. D. Redish: http://redishlab.neuroscience.umn.edu/MClust/MClust.html), Klustakwik (K. D. Harris: https://sourceforge.net/projects/klustakwik/), Wave_clus (R. Q. Quiroga: http://www2.le.ac.uk/departments/engineering/research/bioengineering/neuroengineering-lab/spike-sorting) and the programs Neuroscope and Klusters (http://neurosuite.)[16]. A detailed comparison with these other programs, many of which are in common use, is beyond the scope of the present paper. Such a comparison would involve a variety of related criteria including ease of use, reliability, file format support, GUI design, documentation, degree of automation, dependence on hardware and software components, processing speed, adaptability to MEAs as well as tetrodes, and, to the extent that it is possible to measure it, sorting accuracy. In the absence of a detailed comparison, we believe that SpikeSorter offers a combination of options and support for spike sorting that may not be available in any other currently available standalone spike sorting package.

Sorting Quality

As mentioned above, objective outcome measures that can be used to decide whether one procedure or choice is better than another are largely lacking. The dependence on parameters and the need for frequent user input also makes it unlikely that any particular sort can ever be reproduced. This in itself would limit the use of outcome measures, if they existed. To make matters worse, it is far from certain that accurate spike sorting is possible even in principle. Extracellular recordings coupled with intracellular recordings of nearby single cells exist[17,18] but intracellular recordings from neighboring pairs of neurons are needed to prove that signals from neighboring cells can always be distinguished. The factors that can cause extracellular voltage signals from a given neuron to vary across periods of time, short as well as long, are also not well understood and in practice can add substantial variability (*e.g.* **Figures 8** and **10**) that complicates sorting. For spike sorting to be a solvable problem these changes have to be smaller, or different in nature, than the smallest differences that can occur between cells as a result of positional differences. Relying on numerical measures of cluster quality may also be problematic. For example, cells can fire at rates that differ by orders of magnitude[19,20]. The inclusion of almost all of the spikes from a low firing rate cell among those of a high firing rate cell might have little impact on any cluster quality measure, hiding the fact that the sorting quality of the low rate cell would be poor or non-existent. Given these challenges, methods for assessing the quality of sorting based on cluster overlap[8,21] or sorting stability in the face of parameter variation[22] may give a false sense of security. Instead, we suggest that it may be necessary to accept that spike sorting is based on incomplete science. Sorters may have to live with a sense of imperfection and learn that it may be better to devote the time to more productive forms of data analysis rather than continually trying to improve the quality of a sort.

Disclosures

The authors have nothing to disclose.

Acknowledgements

We thank those individuals and groups who have used SpikeSorter and who have provided requests for file format support and suggestions and feedback on how to improve it. These include Youping Xiao, Felix Fung, Artak Khachatryan, Eric Kuebler, Curtis Baker, Amol Gharat and Dongsheng Xiao. We thank Adrien Peyrache for the false positive and negative figures given in 'Representative Results'.

References

1. Buzsáki, G. Large-scale recording of neuronal ensembles. *Nat. Neurosci.* **7**, 446-451 (2004).
2. Blanche, T.J., Spacek, M.A., Hetke, J.F., Swindale, N.V. Polytrodes: High Density Silicon Electrode Arrays for Large Scale Multiunit Recording. *J. Neurophys.* **93**, 2987-3000 (2005).
3. Lewicki, M.S. A review of methods for spike sorting: the detection and classification of neuronal action potentials. *Network.* **9**, R53-R78 (1998).
4. Letelier, J.C., & Weber, P.P. Spike sorting based on discrete wavelet transform coefficients. *J. Neurosci. Methods.* **101**, 93-106 (2000).
5. Quiroga, R. Q., Nadasdy, Z. and Ben-Shaul, Y. Unsupervised spike detection and sorting with wavelets and superparamagnetic clustering. *Neural Computation.* **16**, 1661-1687 (2004).
6. Franke, F., Natora, M., Boucsein, C., Munk, M., and Obermayer, K. An online spike detection and spike classification algorithm capable of instantaneous resolution of overlapping spikes. *J. Comput. Neurosci.* **29**, 127-148 (2010).
7. Jäckel, D., Frey, U., Fiscella, M., Franke, F., and Hierlemann, A. Applicability of independent component analysis on high-density microelectrode array recordings. *J. Neurophysiol.* **108**, 334-348 (2012).
8. Rossant, C. et al. Spike sorting for large, dense electrode arrays. *Nature Neuroscience.* **19**, 634-641 (2016).
9. Vandecasteele, M. *et al.* Large-scale recording of neurons by movable silicon probes in behaving rodents. *JoVE.* **61**, e3568-e3568 (2012).
10. Schjetnan, A.G.P., & Luczak, A. Recording large-scale neuronal ensembles with silicon probes in the anesthetized rat. *JoVE.* **56**, e3282-e3282 (2011).
11. Swindale, N. V., & Spacek, M. A. Spike sorting for polytrodes: a divide and conquer approach. *Frontiers in Systems Neuroscience.* **8**, 1-21 (2014).
12. Swindale, N. V., & Spacek, M. A. Spike detection methods for polytrodes and high density microelectrode arrays, *J. Comput. Neurosci.* **38**, 249-261 (2015).
13. Swindale, N. V., & Spacek, M. A. Verification of multichannel electrode array integrity by use of cross-channel correlations. *J. Neurosci. Meth.* **263**, 95-102 (2016).
14. Fukunaga, K., Hostetler, L.D. The estimation of the gradient of a density function, with applications in pattern recognition. *IEEE Transactions on Information Theory (IEEE).* **21**, 32-40 (1975).
15. Mitelut, C. *et al.* Standardizing spike sorting: an *in vitro*, *in silico* and *in vivo* study to develop quantitative metrics for sorting extracellularly recorded spiking activity. *Soc. Neurosci. Abstr.* **598**, 10 (2015).
16. Hazan, L., Zugaro, M., Buzsáki, G. Klusters, NeuroScope, NDManager: A free software suite for neurophysiological data processing and visualization. *J. Neurosci. Meth.* **155**, 207-216 (2006).
17. Harris, K.D., Henze, D.A., Csicsvari, J., Hirase, H., Buzsáki, G. Accuracy of tetrode spike separation as determined by simultaneous intracellular and extracellular measurements. *J. Neurophysiol.* **84**, 401-414 (2000).
18. Anastassiou, C. A., Perin, R., Buzsáki, G., Markram, H., Koch, C. Cell-type and activity dependent extracellular correlates of intracellular spiking. *J. Neurophysiol.* **114**, 608-623 (2015).
19. Wohrer, A., Humphries, M.D. and Machens, C.K. Population-wide distributions of neural activity during perceptual decision-making. *Prog. Neurobiol.* **103**, 156-193 (2013).
20. Mizuseki, K., Buzsáki, G. Preconfigured, skewed distribution of firing rates in the hippocampus and entorhinal cortex. *Cell Reports.* **4**, 1010-1021 (2013).
21. Schmitzer-Torbert, N., Jackson, J., Henze, D., Harris, K. and Redish, A.D. Quantitative measures of cluster quality for use in extracellular recordings. *Neuroscience.* **131**, 1-11 (2005).
22. Barnett, A.H., Magland, J.F., Greengard, L.F. Validation of neural spike sorting algorithms without ground-truth information. *J. Neurosci. Meth.* **264**, 65-77 (2016).

