
JRCLUST Documentation

Release 4.0.0

James Jun, Alan Liddell

Feb 26, 2021

Table of contents

1	Using JRCLUST	3
1.1	Tutorial	3
1.2	Migrating from JRCLUST v3	20
1.3	Input and output files	21
1.4	Getting started	25
1.5	System requirements	26
1.6	Usage	26
1.7	Getting help	27
2	The JRCLUST spike-sorting pipeline	29
2.1	Bootstrapping	29
2.2	Spike detection	30
2.3	Spike clustering	34
2.4	Cluster curation	37
3	JRCLUST parameters	49
3.1	Common parameters	49
3.2	Advanced parameters	57
4	Developer documentation	69
4.1	The JRC handle	69
4.2	Package <code>jrclust</code>	69
4.3	The Clustering interface	69
4.4	DensityPeakClustering	69

JRCLUST is a scalable and customizable package for [spike sorting](#) written in [MATLAB](#) and [CUDA](#). JRCLUST runs on a local workstation (it is recommended, but not required, that you have a GPU).

The original JRCLUST paper by Jun, et al., may be found on the [bioRxiv](#).

1.1 Tutorial

1.1.1 Installing JRCLUST

JRCLUST is hosted on [GitHub](#). If you'd like to test the latest development code, you can [clone the repository](#) to your computer. If you want to stay on a release, head to the [releases page](#) and download the latest release.

You may want to add something like the following to your [startup script](#):

```
addpath('/path/to/JRCLUST');
```

You may also need to recompile your CUDA codes if you're not on Windows. Do this with

```
jrclust.CUDA.compileCUDA();
```

Warning: You may get this error on recent versions of Ubuntu: Call to sgemv in CUBLAS failed with error status: CUBLAS_STATUS_EXECUTION_FAILED. This is a [known issue](#), but unfortunately the suggested workaround doesn't seem to work.

1.1.2 Setting up your config file

JRCLUST requires a configuration file specifying a number of *parameters*. How some of these parameters are used is explained in the *pipeline* section. You may set this up with

```
jrc bootstrap
```

or

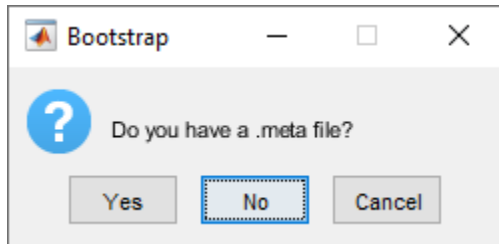
```
jrc bootstrap /path/to/metafile.meta
```

See the [bootstrap documentation](#) for details.

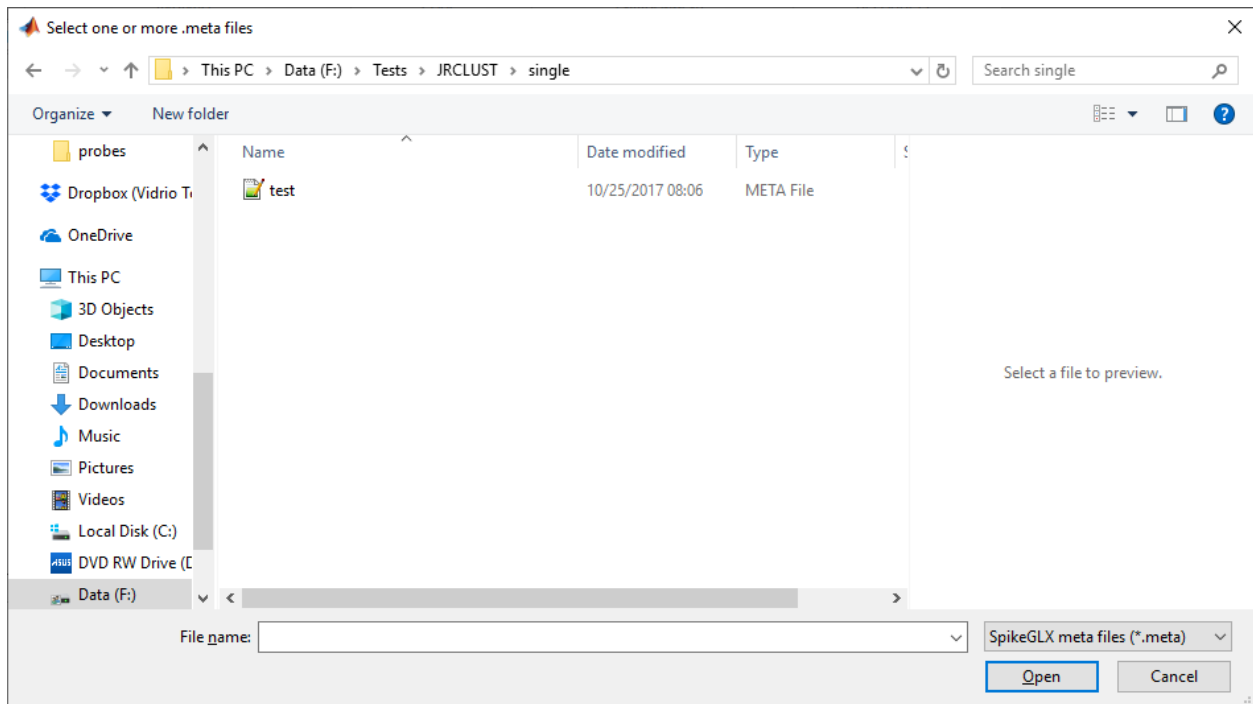
You will be guided through a series of prompts in setting up your config file.

Selecting a meta file

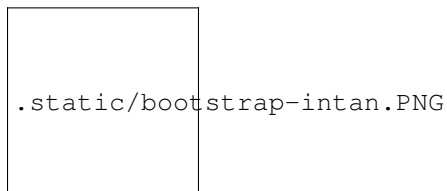
If you specify a meta file with `jrc bootstrap /path/to/metafile.meta`, then you will not be asked to select one; you can skip to the next section. After typing `jrc bootstrap`, you should get a prompt asking if you have a meta file:



If you select “Yes”, you will be prompted to select one or more with a file dialog:



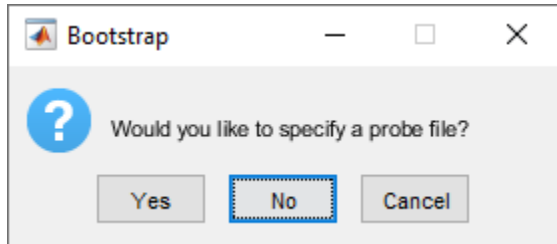
If you select “No” instead, you will be prompted to select one or more **raw recordings** with a similar dialog. You may select from among SpikeGLX `.bin/.dat` files, or Intan `.rhd` traditional-format files.



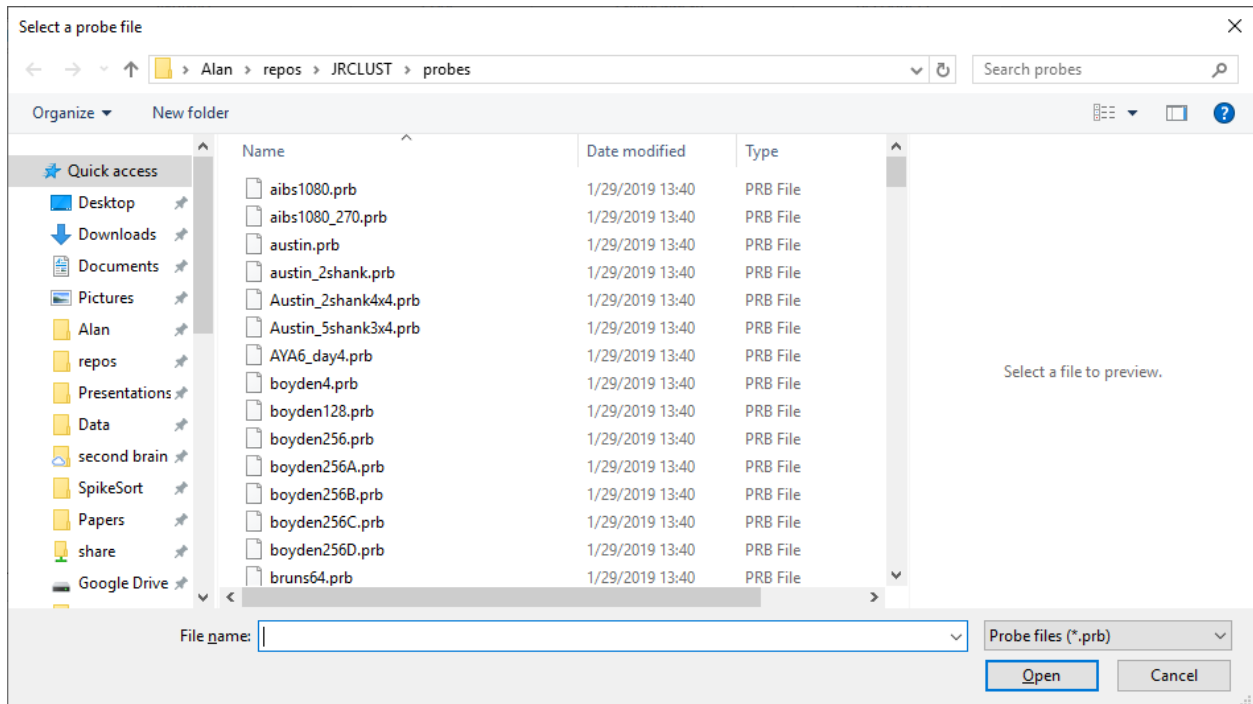
In either case, your working directory will be set to the directory containing the file or files you selected.

Selecting a probe file

You will next be asked if you have a *probe file* you want to specify:



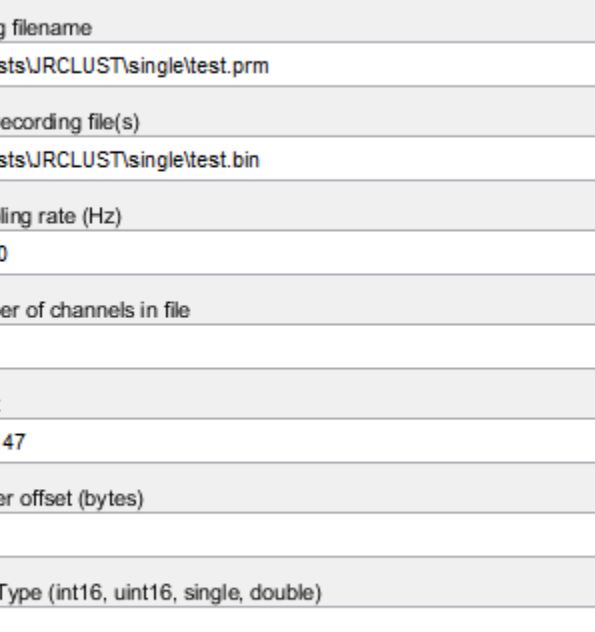
If you select “Yes”, you will be prompted to select exactly one probe file with a file dialog:



Note: If you have a probe file in your working directory, then the dialog will search there first. If you do **not** have a probe file in your working directory, the dialog will search in the default location, JRCLUST/probes, as shown above.

Confirmation

Next, you will be asked to confirm some salient data:



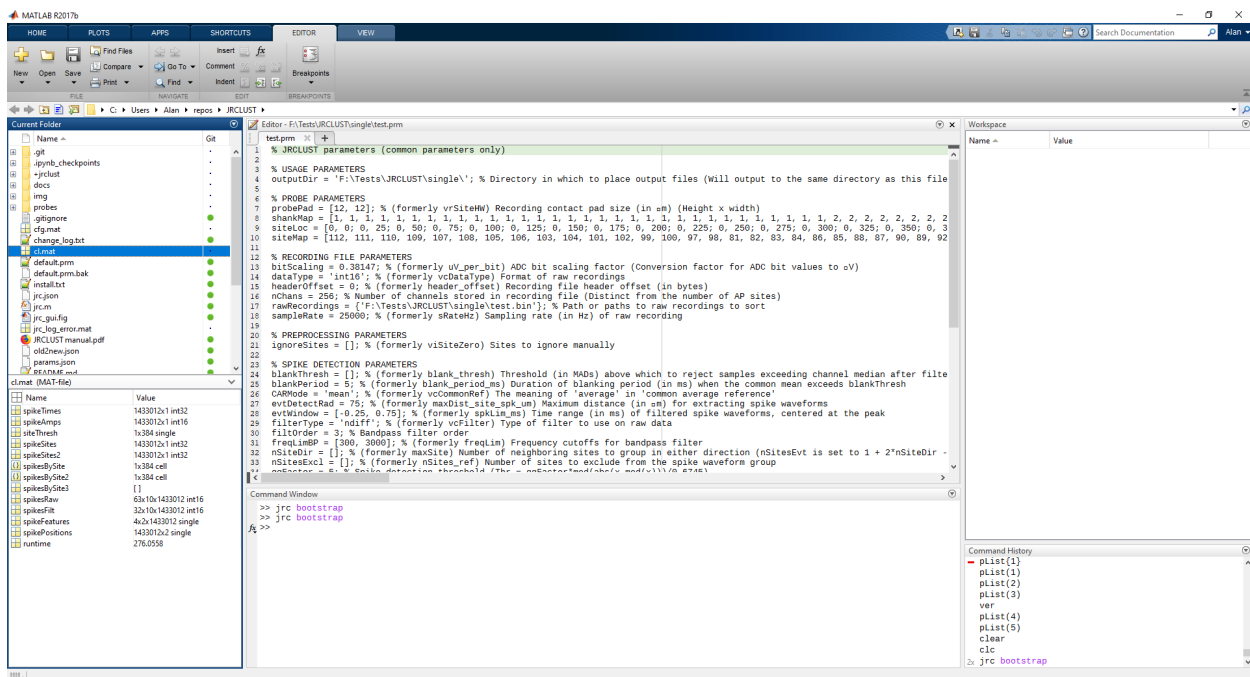
The screenshot shows a MATLAB dialog box titled "Does this look correct?". The dialog contains several configuration fields:

- Config filename:** F:\Tests\URCLUST\single\test.prm
- Raw recording file(s):** F:\Tests\URCLUST\single\test.bin
- Sampling rate (Hz):** 25000
- Number of channels in file:** 256
- $\mu\text{V/bit}$:** 0.38147
- Header offset (bytes):** 0
- Data Type (int16, uint16, single, double):** int16

At the bottom of the dialog are two buttons: "OK" and "Cancel".

Finishing up

Once you are satisfied, JRCLUST will open up your parameter file for editing:



You should end up with a parameter file that looks something like this (see the [parameters page](#) for details):

```

% JRCLUST parameters (default parameter set)
% For a description of these parameters, including default and legal values, see
↳ https://jrclust.readthedocs.io/en/latest/parameters/index.html

% USAGE PARAMETERS
outputDir = ''; % Directory in which to place output files (Will output to the same
↳ directory as this file if empty)

% PROBE PARAMETERS
probePad = [12, 12]; % (formerly vrSiteHW) Recording contact pad size (in  $\mu\text{m}$ ) (Height
↳ x width)
shankMap = []; % (formerly viShank_site) Shank ID of each site
siteLoc = []; % (formerly mrSiteXY) Site locations (in  $\mu\text{m}$ ) (x values in the first
↳ column, y values in the second column)
siteMap = []; % (formerly viSite2Chan) Map of channel index to site ID (The mapping
↳ siteMap(i) = j corresponds to the statement 'site i is stored as channel j in the
↳ recording')

% RECORDING FILE PARAMETERS
bitScaling = 0.30518; % (formerly uV_per_bit) ADC bit scaling factor (Conversion
↳ factor for ADC bit values to  $\mu\text{V}$ )
dataType = 'int16'; % (formerly vcDataType) Format of raw recordings
headerOffset = 0; % (formerly header_offset) Recording file header offset (in bytes)
nChans = 384; % Number of channels stored in recording file (Distinct from the number
↳ of AP sites)
rawRecordings = {' '}; % Path or paths to raw recordings to sort
recordingFormat = 'SpikeGLX'; % Format of raw recording
sampleRate = 30000; % (formerly sRateHz) Sampling rate (in Hz) of raw recording

% PREPROCESSING PARAMETERS
blankThresh = []; % (formerly blank_thresh) Threshold (in MADs) above which to reject
↳ samples exceeding channel median after filtering
filtOrder = 3; % Bandpass filter order
filterType = 'ndiff'; % (formerly vcFilter) Type of filter to use on raw data
freqLimBP = [300, 3000]; % (formerly freqLim) Frequency cutoffs for bandpass filter
ignoreChans = []; % (formerly viChanZero) Channel numbers to ignore manually
ignoreSites = []; % (formerly viSiteZero) Site IDs to ignore manually

% SPIKE DETECTION PARAMETERS
CARMode = 'mean'; % (formerly vcCommonRef) The meaning of 'average' in 'common
↳ average reference'
blankPeriod = 5; % (formerly blank_period_ms) Duration of blanking period (in ms)
↳ when the common mean exceeds blankThresh
evtDetectRad = 50; % (formerly maxDist_site_um) Maximum distance (in  $\mu\text{m}$ ) to search
↳ over for duplicate peaks
evtWindow = [-0.25, 0.75]; % (formerly spkLim_ms) Time range (in ms) of filtered
↳ spike waveforms, centered at the peak
nSiteDir = []; % (formerly maxSite) Number of neighboring sites to group in either
↳ direction (nSitesEvt is set to 1 + 2*nSiteDir - nSitesExcl)
nSitesExcl = []; % (formerly nSites_ref) Number of sites to exclude from the spike
↳ waveform group for feature extraction
qqFactor = 5; % Spike detection threshold factor (Thr = qqFactor*med(abs(x-med(x)))/0.
↳ 6745)
refracInt = 0.25; % (formerly spkRefrac_ms) Spike refractory period (in ms)

% FEATURE EXTRACTION PARAMETERS
clusterFeature = 'pca'; % (formerly vcFet) The feature to extract from your spike
↳ waveforms in order to cluster them

```

(continues on next page)

(continued from previous page)

```

evtGroupRad = 75; % (formerly maxDist_site_spk_um) Maximum distance (in  $\mu$ m) for
↳ extracting spike waveforms
nPcsPerSite = 1; % (formerly nPcPerChan) Number of principal components to compute
↳ per site

% CLUSTERING PARAMETERS
RDEtdetrendMode = 'global'; % (formerly vcDetrend_postclu) Detrending mode to apply to
↳ rho-delta values in order to determine cluster centers
autoMergeBy = 'pearson'; % (formerly autoMergeCriterion) Metric to use for
↳ automerging clusters based on average waveform
distCut = 2; % (formerly dc_percent) Percentile of pairwise distances between spikes
↳ on a site to use as a cutoff distance
evtMergeRad = 35; % (formerly maxDist_site_merge_um) Maximum distance (in  $\mu$ m) to
↳ consider for merging spike waveforms
log10DeltaCut = 0.6; % (formerly delta1_cut) Log10 of delta cutoff (Spikes with delta
↳ values below this cutoff will not be considered as cluster centers)
log10RhoCut = -2.5; % (formerly rho_cut) Log10 of rho cutoff (Spikes with rho values
↳ below this cutoff will not be considered as cluster centers)
maxUnitSim = 0.98; % (formerly maxWavCor) Threshold for merging two units having
↳ similar spike waveforms (Units with a similiarity score above this value will be
↳ merged)
minClusterSize = 30; % (formerly min_count) Minimum number of spikes per cluster
↳ (Automatically set to the maximum of this value and twice the number of features)
nClusterIntervals = 4; % (formerly nTime_clu) Number of intervals to divide the
↳ recording into around a spike (When clustering, take the 1/nClusterIntervals
↳ fraction of all spikes around a spiking event to compute distance)

% DISPLAY PARAMETERS
dispTimeLimits = [0, 0.2]; % (formerly tlim) Time range (in ms) to display
nSpikesFigProj = 500; % (formerly nShow_proj) Maximum number of spikes per cluster to
↳ display in the feature projection view
nSpikesFigWav = 30; % (formerly nSpk_show) Maximum number of spikes per cluster to
↳ display generally

% TRIAL PARAMETERS
psthTimeLimits = []; % (formerly tlim_psth) Time range (in s) over which to display
↳ PSTH
trialFile = ''; % (formerly vcFile_trial) Path to file containing trial data (Can be .
↳ mat or .csv, must contain timestamps of trials in units of s)

```

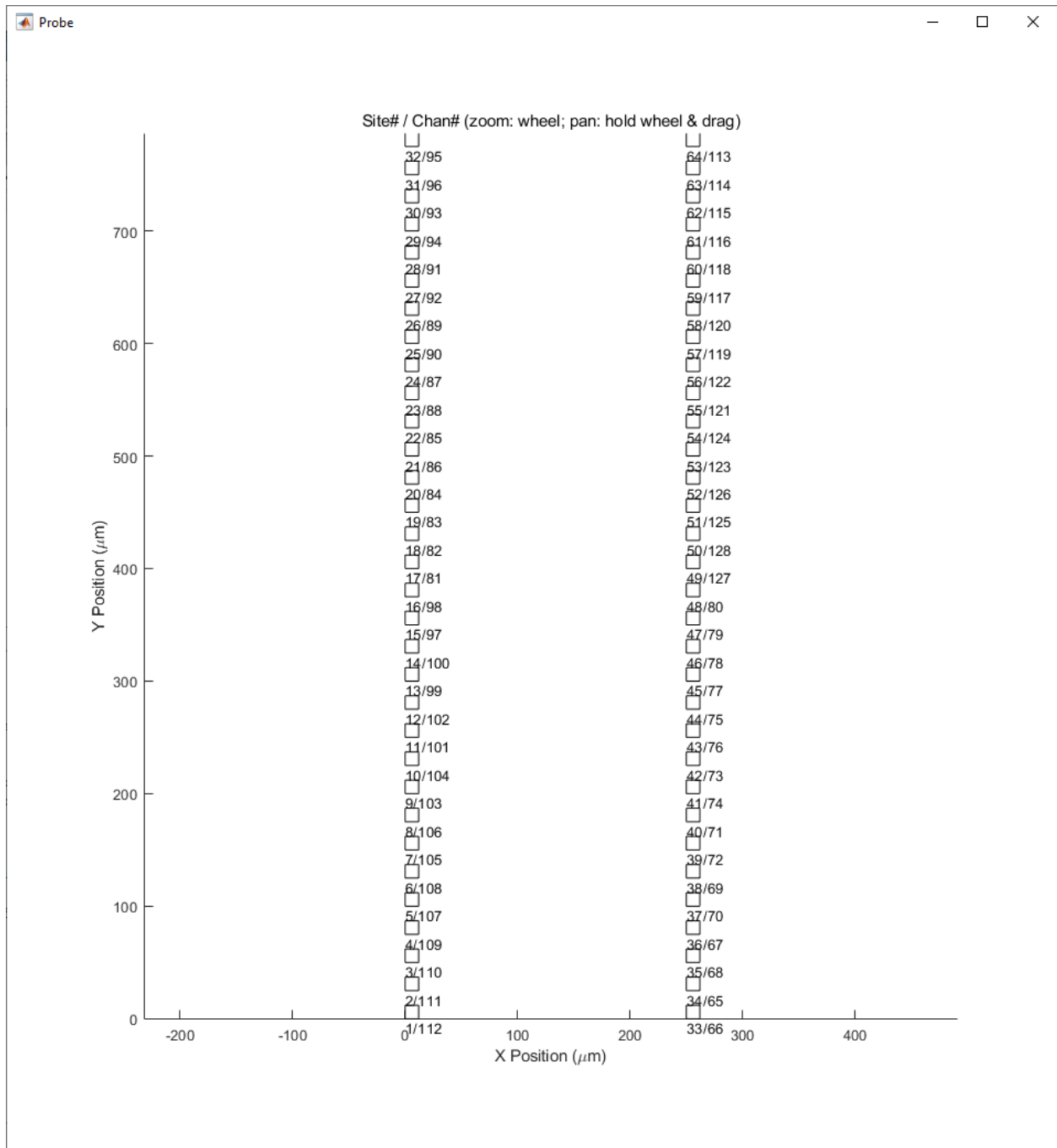
Look over your parameters and ensure they are satisfactory.

1.1.3 Displaying your probe

You might wish to plot your probe to ensure that it's modeled as expected. This can be done with

```
jrc probe /path/to/your/configfile.prm
```

You will get a plot similar to this:



In this example, we have a two-shank probe model, each shank having 32 active electrodes for a total of 64 sites. The shanks are spaced 250 μm apart, with the leftmost shank being considered the origin. (Your probe will almost certainly look different.)

The labels on the patch plots are **site number/channel number**. This is handy for visualizing the channel-site correspondence. You can use this graphic to verify that the sites you gave to JRCLUST are in their proper positions. You can zoom in and out with the scroll wheel to increase or decrease resolution as necessary.

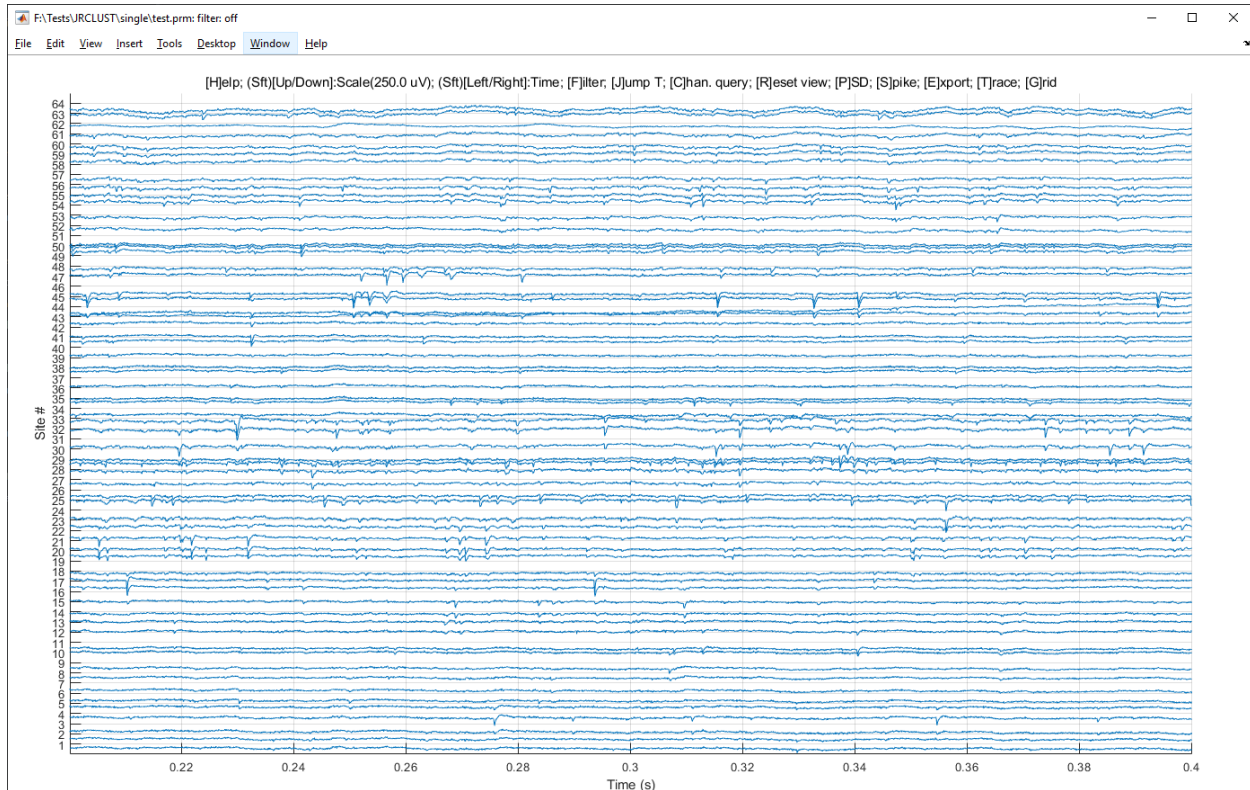
Close this plot and proceed to the next section.

1.1.4 Showing raw traces

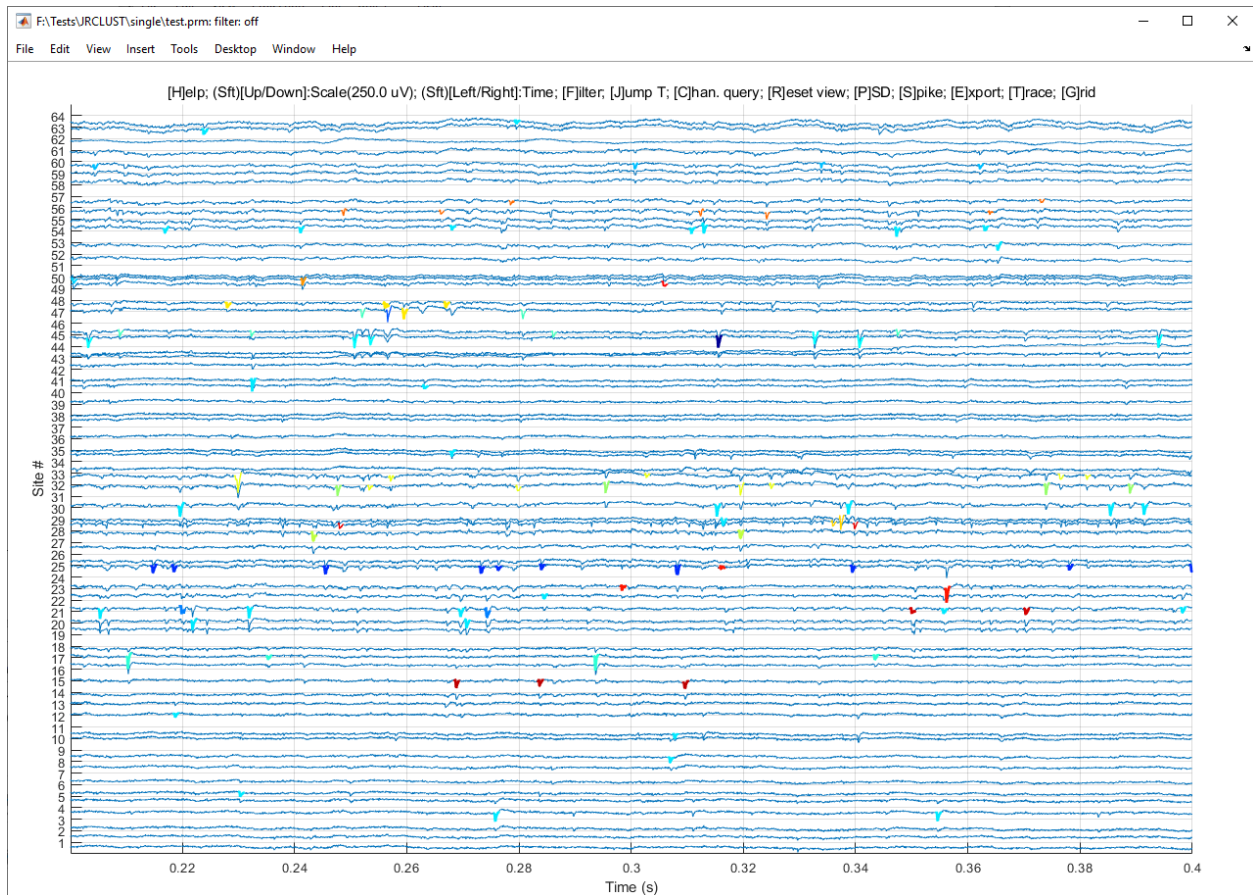
You will probably want to take a look at your raw traces. Do this with

```
jrc traces /path/to/your/configfile.prm
```

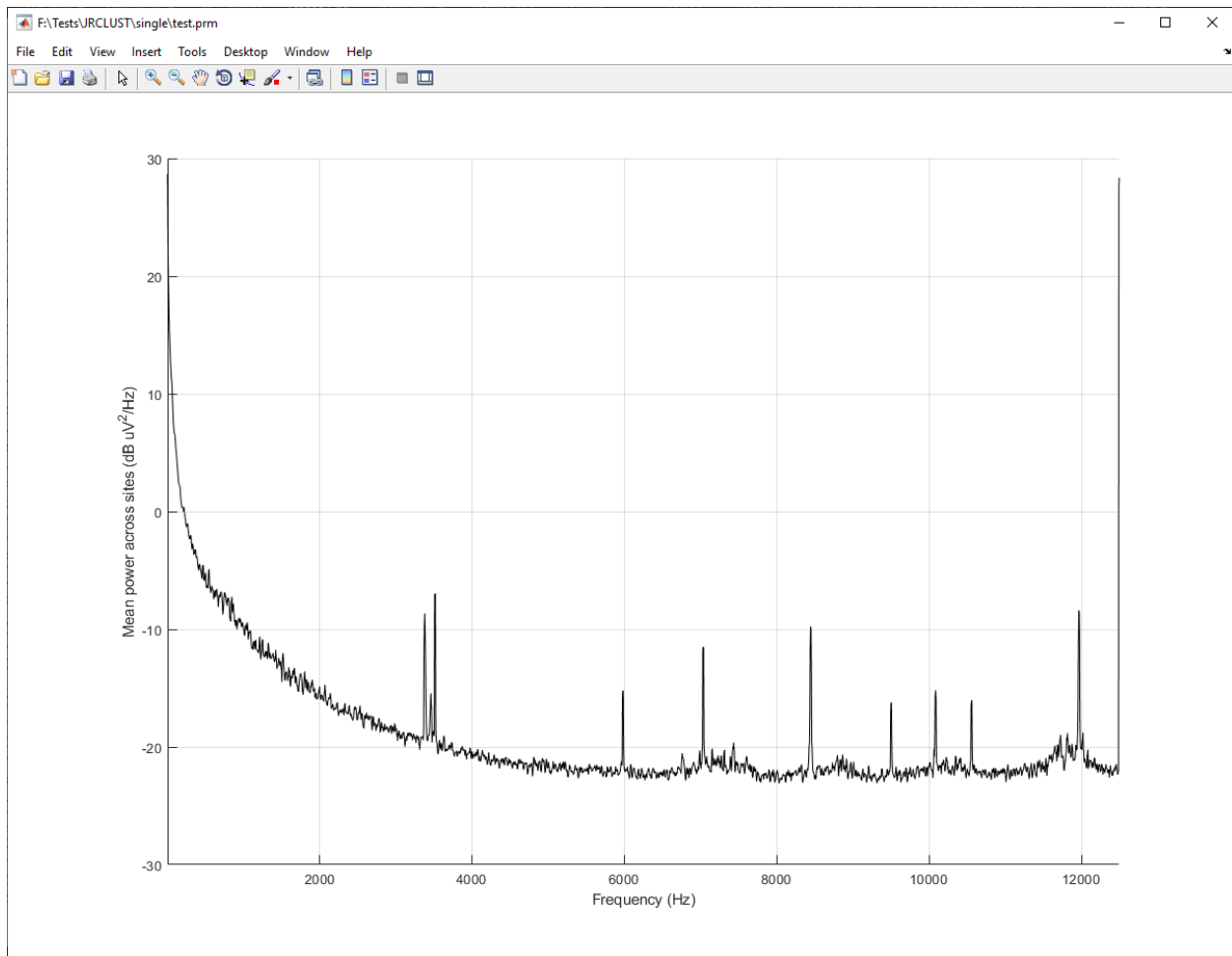
You will get an interactive plot that looks like this:



Here you can page through your raw data, adjusting your scale and applying various filters. Later, after you have detected and clustered your spikes, you can highlight spikes like so:



You can also plot the [power spectral density](#) by hitting the **P** key in this plot. You may select a single site or show the mean power vs. frequency as shown below:

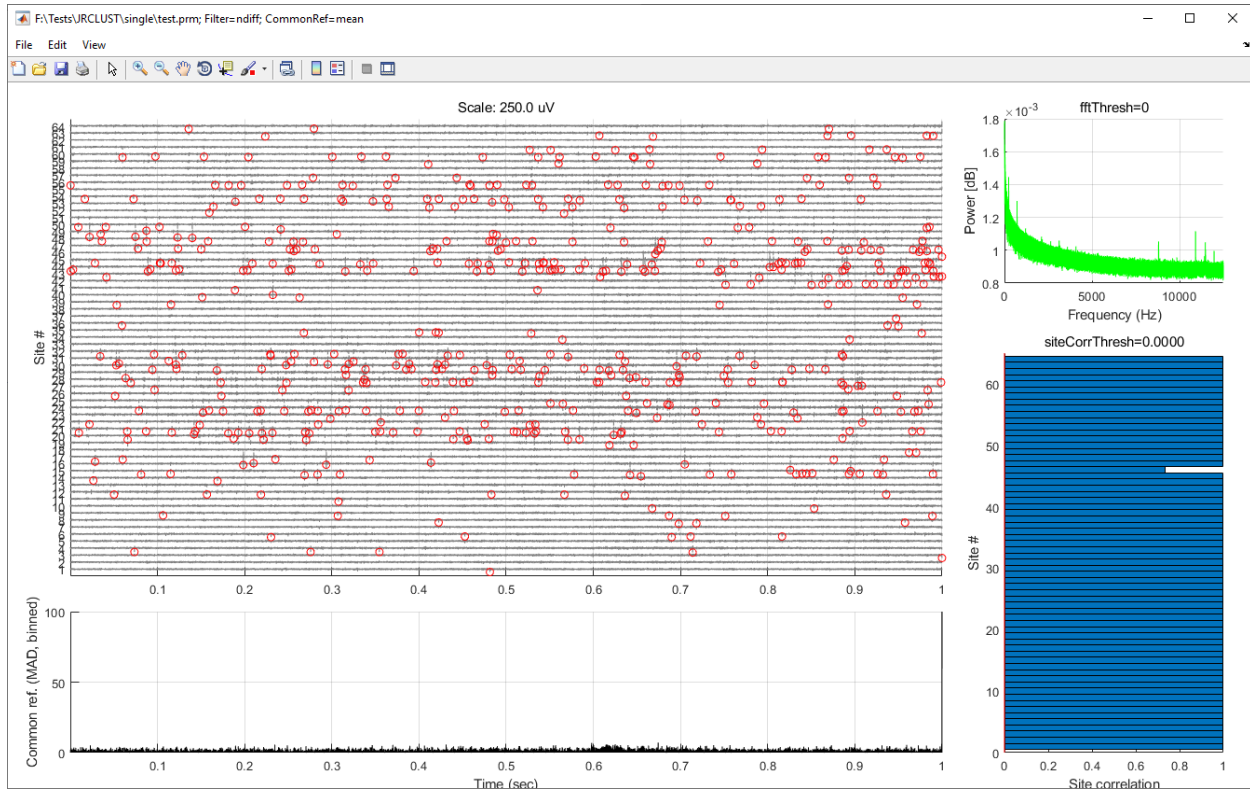


1.1.5 The preview GUI

A more advanced view on your raw traces is the preview GUI. Invoke it like so:

```
jrc preview /path/to/your/configfile.prm
```

This will take a look at your recording and do some preliminary spike detection with the parameters you've specified. (Spikes are circled in red below). You can view the maximum site-to-site correlation and set a threshold for bad sites (i.e., *sites to ignore*) if they come below that threshold. In the figure below, site 46 is poorly correlated with the other sites, so you might choose to ignore it. You can also view the common average across sites at each time step, expressed in units of MAD, to set a *threshold* for blanking out that period. There are many other parameters to set from the preview GUI. Take some time to explore these in the Edit menu, and see the effects they have by changing views in the View menu. (You can see the manual page for details.) Once you are satisfied with these parameters, you can select "Save to [your parameter file here]" from the File menu and start to detect spikes.



1.1.6 Detecting spikes

Now it's time to detect spikes in your file. This can be done with

```
jrc detect /path/to/your/configfile.prm
```

If you want to cluster your spikes immediately afterward, use

```
jrc detect-sort /path/to/your/configfile.prm
```

instead.

Depending on your choice of parameters, you should see something like the following output:

```
>> jrc detect test.prm
2019-08-30 14:04:09 Clearing GPU memory...
2019-08-30 14:04:09 GPU memory cleared (took 0.16 s)
2019-08-30 14:04:09 Detecting spikes in recordings...
2019-08-30 14:04:09 Processing file F:\Misc\Tests\JRCLUST\single\test.bin (1/1)...
2019-08-30 14:04:10 Processing load 1/1...
2019-08-30 14:04:10 Filtering samples...
2019-08-30 14:04:10 Finished filtering samples (took 0.10 s)
2019-08-30 14:04:11 Detecting spikes from each site...
2019-08-30 14:04:11 Detected 19 spikes on site 1
2019-08-30 14:04:11 Detected 17 spikes on site 2
2019-08-30 14:04:11 Detected 69 spikes on site 3
2019-08-30 14:04:11 Detected 127 spikes on site 4
2019-08-30 14:04:11 Detected 29 spikes on site 5
2019-08-30 14:04:11 Detected 22 spikes on site 6
```

(continues on next page)

(continued from previous page)

```
2019-08-30 14:04:11 Detected 51 spikes on site 7
2019-08-30 14:04:11 Detected 61 spikes on site 8
2019-08-30 14:04:11 Detected 52 spikes on site 9
2019-08-30 14:04:11 Detected 31 spikes on site 10
2019-08-30 14:04:11 Detected 113 spikes on site 11
2019-08-30 14:04:11 Detected 139 spikes on site 12
2019-08-30 14:04:11 Detected 61 spikes on site 13
2019-08-30 14:04:11 Detected 122 spikes on site 14
2019-08-30 14:04:11 Detected 274 spikes on site 15
2019-08-30 14:04:11 Detected 1156 spikes on site 16
2019-08-30 14:04:11 Detected 223 spikes on site 17
2019-08-30 14:04:11 Detected 83 spikes on site 18
2019-08-30 14:04:11 Detected 167 spikes on site 19
2019-08-30 14:04:11 Detected 362 spikes on site 20
2019-08-30 14:04:11 Detected 528 spikes on site 21
2019-08-30 14:04:11 Detected 248 spikes on site 22
2019-08-30 14:04:11 Detected 282 spikes on site 23
2019-08-30 14:04:11 Detected 331 spikes on site 24
2019-08-30 14:04:11 Detected 213 spikes on site 25
2019-08-30 14:04:11 Detected 207 spikes on site 26
2019-08-30 14:04:11 Detected 112 spikes on site 27
2019-08-30 14:04:11 Detected 426 spikes on site 28
2019-08-30 14:04:11 Detected 368 spikes on site 29
2019-08-30 14:04:11 Detected 227 spikes on site 30
2019-08-30 14:04:11 Detected 323 spikes on site 31
2019-08-30 14:04:11 Detected 404 spikes on site 32
2019-08-30 14:04:11 Detected 2047 spikes on site 33
2019-08-30 14:04:11 Detected 150 spikes on site 34
2019-08-30 14:04:11 Detected 69 spikes on site 35
2019-08-30 14:04:11 Detected 68 spikes on site 36
2019-08-30 14:04:11 Detected 121 spikes on site 37
2019-08-30 14:04:11 Detected 97 spikes on site 38
2019-08-30 14:04:11 Detected 102 spikes on site 39
2019-08-30 14:04:11 Detected 89 spikes on site 40
2019-08-30 14:04:11 Detected 114 spikes on site 41
2019-08-30 14:04:11 Detected 215 spikes on site 42
2019-08-30 14:04:11 Detected 251 spikes on site 43
2019-08-30 14:04:11 Detected 457 spikes on site 44
2019-08-30 14:04:11 Detected 645 spikes on site 45
2019-08-30 14:04:11 Detected 182 spikes on site 46
2019-08-30 14:04:11 Detected 278 spikes on site 47
2019-08-30 14:04:11 Detected 408 spikes on site 48
2019-08-30 14:04:11 Detected 337 spikes on site 49
2019-08-30 14:04:11 Detected 347 spikes on site 50
2019-08-30 14:04:11 Detected 168 spikes on site 51
2019-08-30 14:04:11 Detected 118 spikes on site 52
2019-08-30 14:04:11 Detected 196 spikes on site 53
2019-08-30 14:04:11 Detected 185 spikes on site 54
2019-08-30 14:04:11 Detected 105 spikes on site 55
2019-08-30 14:04:11 Detected 275 spikes on site 56
2019-08-30 14:04:11 Detected 121 spikes on site 57
2019-08-30 14:04:11 Detected 116 spikes on site 58
2019-08-30 14:04:11 Detected 250 spikes on site 59
2019-08-30 14:04:11 Detected 232 spikes on site 60
2019-08-30 14:04:11 Detected 62 spikes on site 61
2019-08-30 14:04:11 Detected 139 spikes on site 62
2019-08-30 14:04:11 Detected 51 spikes on site 63
```

(continues on next page)

(continued from previous page)

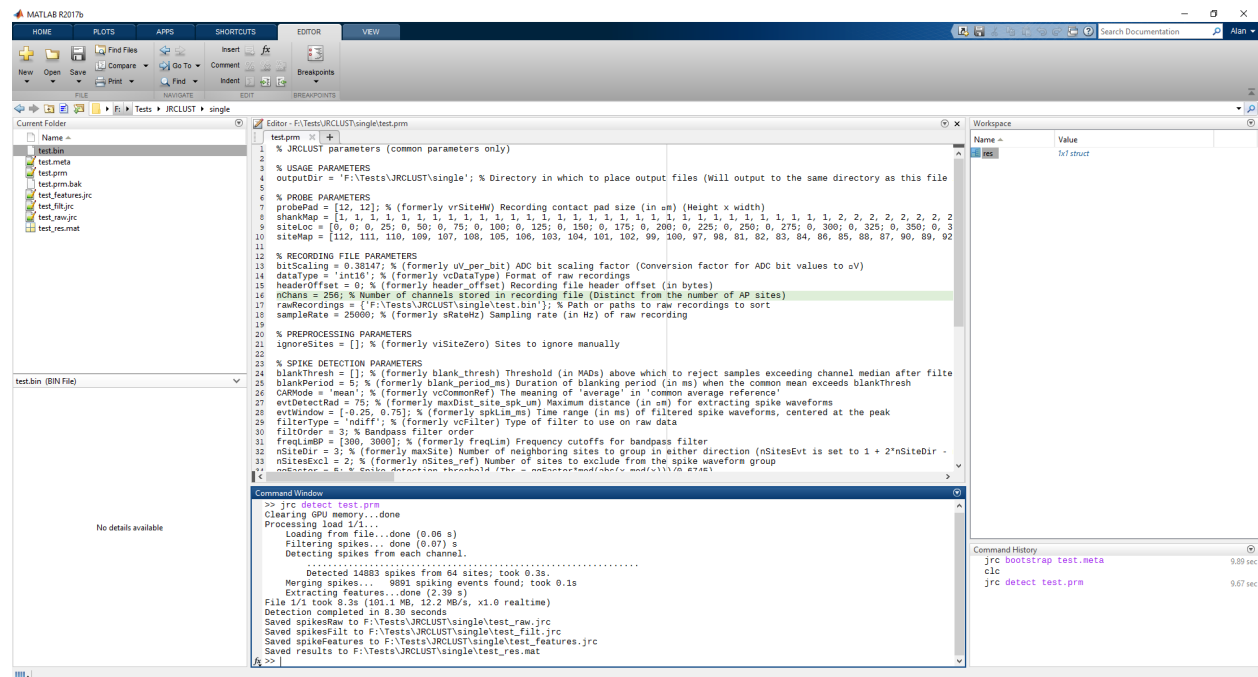
```

2019-08-30 14:04:11 Detected 41 spikes on site 64
2019-08-30 14:04:11 Finished detecting spikes (took 0.15 s)
2019-08-30 14:04:11 Merging duplicate spiking events...
2019-08-30 14:04:11 9891 spiking events found (took 0.10 s)
2019-08-30 14:04:11 Finished load 1/1 (took 1.46 s)
2019-08-30 14:04:11 Finished processing file F:\Misc\Tests\JRCLUST\single\test.bin (1/
→1) (took 2.04 s)
2019-08-30 14:04:11 Extracting features...
2019-08-30 14:04:12 Finished extracting features (took 0.31 s)
2019-08-30 14:04:12 Finished detecting (took 2.50 s)
2019-08-30 14:04:15 Saving results to F:\Misc\Tests\JRCLUST\single\test_res.mat...
2019-08-30 14:04:15 Results saved to F:\Misc\Tests\JRCLUST\single\test_res.mat (took
→0.02 s)

====DETECTION SUMMARY====
Detection completed in 2.50 s
Spike count: 9891
Spike counts per site: min 8 (site 2), max 2036 (site 33), median 85

```

Your detection results will be exported to the workspace in the form of a struct called `res` for inspection. See *Input and output files* for a description of the contents. For a detailed description of the detect step, see *Spike detection*.



1.1.7 Clustering spikes

Once you have detected your spikes, it's time to cluster them:

```
jrc sort /path/to/your/configfile.prm
```

You should see something like the following output:

```

>> jrc sort test.prm
2019-08-30 14:06:37 Loading F:\Misc\Tests\JRCLUST\single\test_res.mat...
2019-08-30 14:06:37 Finished loading F:\Misc\Tests\JRCLUST\single\test_res.mat (took
↳0.01 s)
2019-08-30 14:06:37 Loading F:\Misc\Tests\JRCLUST\single\test_raw.jrc...
2019-08-30 14:06:37 Finished loading F:\Misc\Tests\JRCLUST\single\test_raw.jrc (took
↳0.01 s)
2019-08-30 14:06:37 Loading F:\Misc\Tests\JRCLUST\single\test_filt.jrc...
2019-08-30 14:06:37 Finished loading F:\Misc\Tests\JRCLUST\single\test_filt.jrc (took
↳0.01 s)
2019-08-30 14:06:37 Loading F:\Misc\Tests\JRCLUST\single\test_features.jrc...
2019-08-30 14:06:37 Finished loading F:\Misc\Tests\JRCLUST\single\test_features.jrc
↳(took 0.00 s)
2019-08-30 14:06:37 Using spikes detected on 30-Aug-2019 14:04:12
2019-08-30 14:06:37 Clearing GPU memory...
2019-08-30 14:06:37 GPU memory cleared (took 0.16 s)
2019-08-30 14:06:37 Sorting detected spikes...
2019-08-30 14:06:37 Computing rho...
2019-08-30 14:06:38 Site 1: rho cutoff, 731869.75; average rho: 0.00016 (15 spikes)
2019-08-30 14:06:38 Site 2: rho cutoff, 1367942.13; average rho: 0.00036 (8 spikes)
2019-08-30 14:06:38 Site 3: rho cutoff, 371085.19; average rho: 0.00059 (44 spikes)
2019-08-30 14:06:38 Site 4: rho cutoff, 454989.97; average rho: 0.00098 (110 spikes)
2019-08-30 14:06:38 Site 5: rho cutoff, 818252.50; average rho: 0.00114 (21 spikes)
2019-08-30 14:06:38 Site 6: rho cutoff, 1096531.38; average rho: 0.00129 (8 spikes)
2019-08-30 14:06:38 Site 7: rho cutoff, 575589.00; average rho: 0.00151 (34 spikes)
2019-08-30 14:06:38 Site 8: rho cutoff, 481210.56; average rho: 0.00175 (29 spikes)
2019-08-30 14:06:38 Site 9: rho cutoff, 636892.19; average rho: 0.00193 (40 spikes)
2019-08-30 14:06:38 Site 10: rho cutoff, 2221163.75; average rho: 0.00211 (12 spikes)
2019-08-30 14:06:38 Site 11: rho cutoff, 692781.94; average rho: 0.00248 (91 spikes)
2019-08-30 14:06:38 Site 12: rho cutoff, 503940.38; average rho: 0.00293 (103 spikes)
2019-08-30 14:06:38 Site 13: rho cutoff, 745752.50; average rho: 0.00311 (39 spikes)
2019-08-30 14:06:38 Site 14: rho cutoff, 1094450.50; average rho: 0.00332 (43 spikes)
2019-08-30 14:06:38 Site 15: rho cutoff, 855698.94; average rho: 0.00402 (222 spikes)
2019-08-30 14:06:38 Site 16: rho cutoff, 308155.81; average rho: 0.00683 (1007 spikes)
2019-08-30 14:06:38 Site 17: rho cutoff, 829652.13; average rho: 0.00736 (161 spikes)
2019-08-30 14:06:38 Site 18: rho cutoff, 783599.13; average rho: 0.00758 (32 spikes)
2019-08-30 14:06:38 Site 19: rho cutoff, 698739.75; average rho: 0.00802 (85 spikes)
2019-08-30 14:06:38 Site 20: rho cutoff, 778769.88; average rho: 0.00854 (168 spikes)
2019-08-30 14:06:38 Site 21: rho cutoff, 894751.50; average rho: 0.00948 (369 spikes)
2019-08-30 14:06:38 Site 22: rho cutoff, 895366.63; average rho: 0.00986 (113 spikes)
2019-08-30 14:06:38 Site 23: rho cutoff, 811321.75; average rho: 0.01045 (141 spikes)
2019-08-30 14:06:38 Site 24: rho cutoff, 961806.00; average rho: 0.01101 (211 spikes)
2019-08-30 14:06:38 Site 25: rho cutoff, 917075.13; average rho: 0.01133 (74 spikes)
2019-08-30 14:06:38 Site 26: rho cutoff, 685997.25; average rho: 0.01184 (150 spikes)
2019-08-30 14:06:38 Site 27: rho cutoff, 1064097.88; average rho: 0.01209 (47 spikes)
2019-08-30 14:06:38 Site 28: rho cutoff, 539079.56; average rho: 0.01316 (250 spikes)
2019-08-30 14:06:38 Site 29: rho cutoff, 1070611.50; average rho: 0.01401 (261 spikes)
2019-08-30 14:06:38 Site 30: rho cutoff, 827772.50; average rho: 0.01425 (59 spikes)
2019-08-30 14:06:38 Site 31: rho cutoff, 1146530.88; average rho: 0.01492 (153 spikes)
2019-08-30 14:06:38 Site 32: rho cutoff, 970855.06; average rho: 0.01557 (241 spikes)
2019-08-30 14:06:38 Site 33: rho cutoff, 251356.16; average rho: 0.02157 (2036 spikes)
2019-08-30 14:06:38 Site 34: rho cutoff, 260651.09; average rho: 0.02213 (139 spikes)
2019-08-30 14:06:38 Site 35: rho cutoff, 714987.13; average rho: 0.02235 (45 spikes)
2019-08-30 14:06:38 Site 36: rho cutoff, 631074.38; average rho: 0.02262 (52 spikes)
2019-08-30 14:06:38 Site 37: rho cutoff, 453064.25; average rho: 0.02289 (60 spikes)
2019-08-30 14:06:38 Site 38: rho cutoff, 805470.44; average rho: 0.02323 (71 spikes)
2019-08-30 14:06:38 Site 39: rho cutoff, 660921.75; average rho: 0.02345 (45 spikes)

```

(continues on next page)

(continued from previous page)

```

2019-08-30 14:06:38 Site 40: rho cutoff, 685622.75; average rho: 0.02372 (39 spikes)
2019-08-30 14:06:38 Site 41: rho cutoff, 1017240.00; average rho: 0.02414 (79 spikes)
2019-08-30 14:06:38 Site 42: rho cutoff, 724510.81; average rho: 0.02464 (129 spikes)
2019-08-30 14:06:39 Site 43: rho cutoff, 1318095.13; average rho: 0.02502 (85 spikes)
2019-08-30 14:06:39 Site 44: rho cutoff, 810584.25; average rho: 0.02592 (312 spikes)
2019-08-30 14:06:39 Site 45: rho cutoff, 913448.50; average rho: 0.02711 (374 spikes)
2019-08-30 14:06:39 Site 46: rho cutoff, 928920.94; average rho: 0.02750 (67 spikes)
2019-08-30 14:06:39 Site 47: rho cutoff, 1020212.25; average rho: 0.02830 (236 spikes)
2019-08-30 14:06:39 Site 48: rho cutoff, 465102.56; average rho: 0.02917 (257 spikes)
2019-08-30 14:06:39 Site 49: rho cutoff, 804615.63; average rho: 0.02972 (188 spikes)
2019-08-30 14:06:39 Site 50: rho cutoff, 559785.13; average rho: 0.03033 (187 spikes)
2019-08-30 14:06:39 Site 51: rho cutoff, 557701.31; average rho: 0.03050 (16 spikes)
2019-08-30 14:06:39 Site 52: rho cutoff, 371283.31; average rho: 0.03069 (39 spikes)
2019-08-30 14:06:39 Site 53: rho cutoff, 542745.94; average rho: 0.03103 (94 spikes)
2019-08-30 14:06:39 Site 54: rho cutoff, 498734.75; average rho: 0.03148 (137 spikes)
2019-08-30 14:06:39 Site 55: rho cutoff, 1296482.13; average rho: 0.03166 (20 spikes)
2019-08-30 14:06:39 Site 56: rho cutoff, 377575.56; average rho: 0.03241 (241 spikes)
2019-08-30 14:06:39 Site 57: rho cutoff, 347963.09; average rho: 0.03268 (59 spikes)
2019-08-30 14:06:39 Site 58: rho cutoff, 849382.00; average rho: 0.03297 (58 spikes)
2019-08-30 14:06:39 Site 59: rho cutoff, 531617.50; average rho: 0.03337 (102 spikes)
2019-08-30 14:06:39 Site 60: rho cutoff, 354038.84; average rho: 0.03401 (177 spikes)
2019-08-30 14:06:39 Site 61: rho cutoff, 357702.53; average rho: 0.03422 (26 spikes)
2019-08-30 14:06:39 Site 62: rho cutoff, 881595.94; average rho: 0.03482 (123 spikes)
2019-08-30 14:06:39 Site 63: rho cutoff, 457689.47; average rho: 0.03507 (36 spikes)
2019-08-30 14:06:39 Site 64: rho cutoff, 699817.06; average rho: 0.03532 (21 spikes)
2019-08-30 14:06:39 Finished computing rho (took 1.27 s)
2019-08-30 14:06:39 Computing delta...
2019-08-30 14:06:39 Site 1: median delta: 1.27137 (15 spikes)
2019-08-30 14:06:39 Site 2: median delta: 1.66295 (8 spikes)
2019-08-30 14:06:39 Site 3: median delta: 1.00265 (44 spikes)
2019-08-30 14:06:39 Site 4: median delta: 0.85708 (110 spikes)
2019-08-30 14:06:39 Site 5: median delta: 1.58162 (21 spikes)
2019-08-30 14:06:39 Site 6: median delta: 1.52481 (8 spikes)
2019-08-30 14:06:39 Site 7: median delta: 1.24786 (34 spikes)
2019-08-30 14:06:39 Site 8: median delta: 1.22150 (29 spikes)
2019-08-30 14:06:39 Site 9: median delta: 1.08609 (40 spikes)
2019-08-30 14:06:39 Site 10: median delta: 1.10892 (12 spikes)
2019-08-30 14:06:39 Site 11: median delta: 0.87843 (91 spikes)
2019-08-30 14:06:39 Site 12: median delta: 0.86909 (103 spikes)
2019-08-30 14:06:39 Site 13: median delta: 1.10214 (39 spikes)
2019-08-30 14:06:39 Site 14: median delta: 1.15297 (43 spikes)
2019-08-30 14:06:39 Site 15: median delta: 0.73530 (222 spikes)
2019-08-30 14:06:39 Site 16: median delta: 0.55511 (1007 spikes)
2019-08-30 14:06:39 Site 17: median delta: 0.75445 (161 spikes)
2019-08-30 14:06:39 Site 18: median delta: 1.07030 (32 spikes)
2019-08-30 14:06:39 Site 19: median delta: 0.90345 (85 spikes)
2019-08-30 14:06:39 Site 20: median delta: 0.80473 (168 spikes)
2019-08-30 14:06:39 Site 21: median delta: 0.66820 (369 spikes)
2019-08-30 14:06:39 Site 22: median delta: 0.91149 (113 spikes)
2019-08-30 14:06:39 Site 23: median delta: 0.76094 (141 spikes)
2019-08-30 14:06:39 Site 24: median delta: 0.75902 (211 spikes)
2019-08-30 14:06:39 Site 25: median delta: 0.97118 (74 spikes)
2019-08-30 14:06:39 Site 26: median delta: 0.81842 (150 spikes)
2019-08-30 14:06:39 Site 27: median delta: 0.95115 (47 spikes)
2019-08-30 14:06:39 Site 28: median delta: 0.72237 (250 spikes)
2019-08-30 14:06:39 Site 29: median delta: 0.69283 (261 spikes)
2019-08-30 14:06:39 Site 30: median delta: 1.03695 (59 spikes)

```

(continues on next page)

(continued from previous page)

```

2019-08-30 14:06:39 Site 31: median delta: 0.75059 (153 spikes)
2019-08-30 14:06:39 Site 32: median delta: 0.74415 (241 spikes)
2019-08-30 14:06:39 Site 33: median delta: 0.47524 (2036 spikes)
2019-08-30 14:06:39 Site 34: median delta: 0.83504 (139 spikes)
2019-08-30 14:06:39 Site 35: median delta: 1.03859 (45 spikes)
2019-08-30 14:06:39 Site 36: median delta: 0.97808 (52 spikes)
2019-08-30 14:06:39 Site 37: median delta: 0.98207 (60 spikes)
2019-08-30 14:06:39 Site 38: median delta: 0.94462 (71 spikes)
2019-08-30 14:06:39 Site 39: median delta: 1.00673 (45 spikes)
2019-08-30 14:06:39 Site 40: median delta: 0.99713 (39 spikes)
2019-08-30 14:06:39 Site 41: median delta: 0.87949 (79 spikes)
2019-08-30 14:06:39 Site 42: median delta: 0.84302 (129 spikes)
2019-08-30 14:06:39 Site 43: median delta: 0.90183 (85 spikes)
2019-08-30 14:06:39 Site 44: median delta: 0.68221 (312 spikes)
2019-08-30 14:06:39 Site 45: median delta: 0.66591 (374 spikes)
2019-08-30 14:06:39 Site 46: median delta: 0.92985 (67 spikes)
2019-08-30 14:06:39 Site 47: median delta: 0.70058 (236 spikes)
2019-08-30 14:06:39 Site 48: median delta: 0.73736 (257 spikes)
2019-08-30 14:06:39 Site 49: median delta: 0.76567 (188 spikes)
2019-08-30 14:06:39 Site 50: median delta: 0.80459 (187 spikes)
2019-08-30 14:06:39 Site 51: median delta: 1.46500 (16 spikes)
2019-08-30 14:06:39 Site 52: median delta: 1.09898 (39 spikes)
2019-08-30 14:06:39 Site 53: median delta: 0.92572 (94 spikes)
2019-08-30 14:06:39 Site 54: median delta: 0.81506 (137 spikes)
2019-08-30 14:06:39 Site 55: median delta: 1.22181 (20 spikes)
2019-08-30 14:06:39 Site 56: median delta: 0.73346 (241 spikes)
2019-08-30 14:06:39 Site 57: median delta: 1.03142 (59 spikes)
2019-08-30 14:06:39 Site 58: median delta: 0.98693 (58 spikes)
2019-08-30 14:06:39 Site 59: median delta: 0.86694 (102 spikes)
2019-08-30 14:06:39 Site 60: median delta: 0.83345 (177 spikes)
2019-08-30 14:06:39 Site 61: median delta: 1.02984 (26 spikes)
2019-08-30 14:06:39 Site 62: median delta: 0.77846 (123 spikes)
2019-08-30 14:06:39 Site 63: median delta: 1.04947 (36 spikes)
2019-08-30 14:06:39 Site 64: median delta: 1.07382 (21 spikes)
2019-08-30 14:06:39 Finished computing delta (took 0.27 s)
2019-08-30 14:06:39 Assigning clusters (nClusters: 528)...
2019-08-30 14:06:39 9363/9891 spikes unassigned, 709 can be assigned
2019-08-30 14:06:39 8654/9891 spikes unassigned, 1015 can be assigned
2019-08-30 14:06:39 7639/9891 spikes unassigned, 1144 can be assigned
2019-08-30 14:06:39 6495/9891 spikes unassigned, 1065 can be assigned
2019-08-30 14:06:39 5430/9891 spikes unassigned, 977 can be assigned
2019-08-30 14:06:39 4453/9891 spikes unassigned, 866 can be assigned
2019-08-30 14:06:39 3587/9891 spikes unassigned, 740 can be assigned
2019-08-30 14:06:39 2847/9891 spikes unassigned, 620 can be assigned
2019-08-30 14:06:39 2227/9891 spikes unassigned, 519 can be assigned
2019-08-30 14:06:39 1708/9891 spikes unassigned, 438 can be assigned
2019-08-30 14:06:39 1270/9891 spikes unassigned, 333 can be assigned
2019-08-30 14:06:39 937/9891 spikes unassigned, 247 can be assigned
2019-08-30 14:06:39 690/9891 spikes unassigned, 218 can be assigned
2019-08-30 14:06:39 472/9891 spikes unassigned, 148 can be assigned
2019-08-30 14:06:39 324/9891 spikes unassigned, 103 can be assigned
2019-08-30 14:06:39 221/9891 spikes unassigned, 58 can be assigned
2019-08-30 14:06:39 163/9891 spikes unassigned, 34 can be assigned
2019-08-30 14:06:39 129/9891 spikes unassigned, 27 can be assigned
2019-08-30 14:06:39 102/9891 spikes unassigned, 28 can be assigned
2019-08-30 14:06:39 74/9891 spikes unassigned, 21 can be assigned
2019-08-30 14:06:39 53/9891 spikes unassigned, 15 can be assigned

```

(continues on next page)

(continued from previous page)

```

2019-08-30 14:06:39 38/9891 spikes unassigned, 18 can be assigned
2019-08-30 14:06:39 20/9891 spikes unassigned, 12 can be assigned
2019-08-30 14:06:39 8/9891 spikes unassigned, 4 can be assigned
2019-08-30 14:06:39 4/9891 spikes unassigned, 1 can be assigned
2019-08-30 14:06:39 3/9891 spikes unassigned, 2 can be assigned
2019-08-30 14:06:39 1/9891 spikes unassigned, 1 can be assigned
2019-08-30 14:06:39 9826/9891 spikes unassigned, 272 can be assigned
2019-08-30 14:06:39 9554/9891 spikes unassigned, 589 can be assigned
2019-08-30 14:06:39 8965/9891 spikes unassigned, 867 can be assigned
2019-08-30 14:06:39 8098/9891 spikes unassigned, 958 can be assigned
2019-08-30 14:06:39 7140/9891 spikes unassigned, 1008 can be assigned
2019-08-30 14:06:39 6132/9891 spikes unassigned, 976 can be assigned
2019-08-30 14:06:39 5156/9891 spikes unassigned, 872 can be assigned
2019-08-30 14:06:39 4284/9891 spikes unassigned, 726 can be assigned
2019-08-30 14:06:39 3558/9891 spikes unassigned, 618 can be assigned
2019-08-30 14:06:39 2940/9891 spikes unassigned, 529 can be assigned
2019-08-30 14:06:39 2411/9891 spikes unassigned, 416 can be assigned
2019-08-30 14:06:39 1995/9891 spikes unassigned, 325 can be assigned
2019-08-30 14:06:39 1670/9891 spikes unassigned, 284 can be assigned
2019-08-30 14:06:39 1386/9891 spikes unassigned, 185 can be assigned
2019-08-30 14:06:39 1201/9891 spikes unassigned, 122 can be assigned
2019-08-30 14:06:39 1079/9891 spikes unassigned, 69 can be assigned
2019-08-30 14:06:39 1010/9891 spikes unassigned, 39 can be assigned
2019-08-30 14:06:39 971/9891 spikes unassigned, 27 can be assigned
2019-08-30 14:06:39 944/9891 spikes unassigned, 28 can be assigned
2019-08-30 14:06:39 916/9891 spikes unassigned, 22 can be assigned
2019-08-30 14:06:39 894/9891 spikes unassigned, 17 can be assigned
2019-08-30 14:06:39 877/9891 spikes unassigned, 19 can be assigned
2019-08-30 14:06:39 858/9891 spikes unassigned, 13 can be assigned
2019-08-30 14:06:39 845/9891 spikes unassigned, 4 can be assigned
2019-08-30 14:06:39 841/9891 spikes unassigned, 1 can be assigned
2019-08-30 14:06:39 840/9891 spikes unassigned, 3 can be assigned
2019-08-30 14:06:39 837/9891 spikes unassigned, 1 can be assigned
2019-08-30 14:06:39 836/9891 spikes unassigned, 1 can be assigned
2019-08-30 14:06:39 Finished initial assignment (65 clusters) (took 0.11 s)
2019-08-30 14:06:39 Computing cluster mean waveforms...
2019-08-30 14:06:39 Finished computing cluster mean waveforms (took 0.17 s)
2019-08-30 14:06:39 Computing waveform correlation...
2019-08-30 14:06:40 Finished computing waveform correlation (took 0.19 s)
2019-08-30 14:06:40 Computing cluster self-similarity...
2019-08-30 14:06:40 Finished computing cluster self-similarity (took 0.05 s)
2019-08-30 14:06:40 Computing cluster quality scores...
2019-08-30 14:06:40 Finished computing cluster quality scores (took 0.06 s)
2019-08-30 14:06:40 Merging clusters by similarity...
2019-08-30 14:06:40 No clusters to merge (took 0.00 s)
2019-08-30 14:06:40 Finished sorting (took 2.52 s)
2019-08-30 14:06:42 Saving results to F:\Misc\Tests\JRCLUST\single\test_res.mat...
2019-08-30 14:06:43 Results saved to F:\Misc\Tests\JRCLUST\single\test_res.mat (took
↳ 0.13 s)

====SORTING SUMMARY====
Sorting completed in 2.51 s
Clusters: 65 (no merges)
Spike count per cluster: min 31 (cluster 24), max 2036 (cluster 37), median 80
Site count per cluster: min 1 (cluster 1), max 1 (cluster 1), median 1

```

JRCLUST will tell you when your spikes were detected and perform *clustering* and postprocessing. As in the *detect*

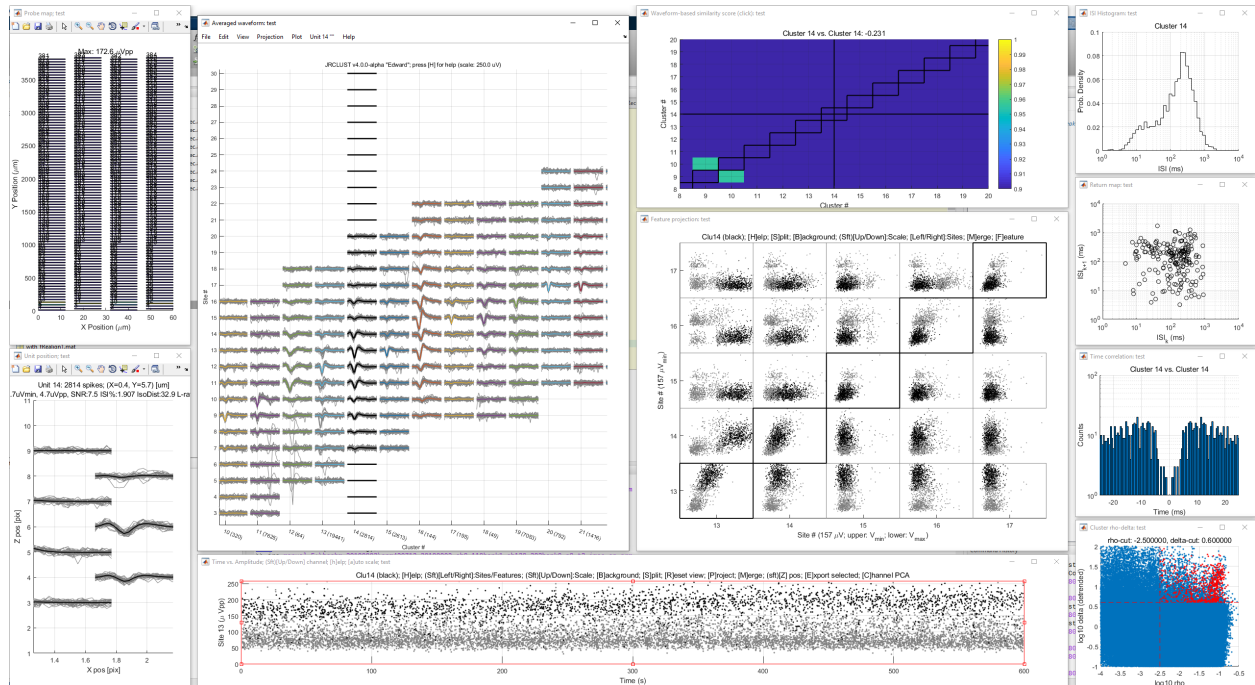
step, JRCLUST will export the results structure to the workspace for inspection.

1.1.8 Curating your clustering

You will now want to inspect the results of your clustering. Do this with

```
jrc manual /path/to/your/configfile.prm
```

You will be greeted with the following screen:



Each of these figures contains a different view onto the data. Here you will be able to annotate, delete, merge, or split clusters. For a description of what each figure does and how to perform these operations, see the [Cluster curation](#) section. When you are satisfied with your clustering, you may save and exit.

1.2 Migrating from JRCLUST v3

1.2.1 What's different

- Variable and parameter names have changed.
- The input and output file names have changed.
- A probe file is no longer required, but may be used.
- `vcFile` and `csFile_merge` are now a single parameter, `rawRecordings`.
- Some commands are *missing*.
- Logging and curation history are stored in the `DensityPeakClustering` object `hClust`, instead of in a separate file where they might get out of sync.
- You must specify your parameter file for every invocation of JRCLUST.

- JRCLUST will overwrite old config files and save a backup copy to ‘filename.prm.bak’.

Note: Old sortings must be manually imported with `jrc importv3 mysession_jrc.mat`. This will output ‘*mysession_res.mat*’ and update your config file. (A backup will be made of your config file at ‘mysession.prm.bak’.) Additionally, your ‘_spkraw.jrc’, ‘_spkwav.jrc’, and ‘_spkfet.jrc’ will be renamed to ‘_raw.jrc’, ‘_filt.jrc’, and ‘_features.jrc’, respectively. Nothing will be altered except for this renaming.

1.3 Input and output files

Replace “mysession”, “myprobe”, and “myrecording” as appropriate.

1.3.1 Input files

Filename	Content	Format
mysession.prm	<i>Config file</i>	Plain text (MATLAB eval-uable)
myprobe.prb	<i>Probe file</i>	Plain text (MATLAB eval-uable)
myrecording.bin	<i>Raw recording files</i>	Binary (16-bit signed integer)
myrecording.meta	Meta file for your recording (see <i>Raw recording files</i>)	Binary file
mysession_trial.mat	<i>Trial file</i>	MATLAB data or CSV

Config file

A MATLAB eval-uable file containing your parameters. See, e.g., *Bootstrapping* and *JRCLUST parameters*.

Probe file

A MATLAB eval-uable file containing parameters for your probe. (Probe files for various configurations are stored in JRCLUST/probes/.)

Note: Probe files are now **optional**. Probe file support is for backwards compatibility. Going forward, you can specify your probe parameters in your config file directly.

Your probe may be visualized with the `probe command`.

Example probe file

```
% Order of the probe sites in the recording file
channels = 1 + [40 103 39 104 41 102 38 105 42 101 37 106 43 100 36 107 44
↳ 99 35 108 45 98 34 109 46 97 33 110 47 96 32 111 48 127 63 112 49 126
↳ 62 113 50 125 61 114 51 124 60 115 52 116 59 123 53 117 58 122 54 118 57
↳ 121 55 119 56 120 8 71 7 72 9 70 6 73 10 74 5 69 11 75 4 68
↳ 12 76 3 67 13 77 2 66 14 78 1 65 15 79 0 64 16 80 31 95 17
↳ 81 30 94 18 82 29 93 19 83 28 92 20 84 27 91 21 85 26 90 22
↳ 86 25 89 23 87 24 88];
```

(continues on next page)

(continued from previous page)

```

% Site location in micrometers (x and y)
geometry = zeros(128, 2); % Create a 128 x 2 matrix filled with zero
geometry(1:2:end,1) = 0; % Set the x-coordinates of the left-edge sites
geometry(2:2:end,1) = 28; % Set the x-coordinates of the right-edge sites
geometry(1:2:end,2) = 20*(0:63); % Set the y-coordinates of the left-edge sites
geometry(2:2:end,2) = geometry(1:2:end,2); % Set the y-coordinates of the right edge
↳sites by copying the left-edge site values

% Reference sites are being excluded
ref_sites = [1 18 33 50 65 82 97 114]; % Specify the site numbers to exclude
channels(ref_sites) = []; % Delete reference sites from the channels vector
geometry(ref_sites,:) = []; % Delete reference sites from the geometry matrix

% Recording contact pad size in micrometers. Height x width
pad = [12 12];

% Default prm
maxSite = 4.5; % Used to calculate the number of sites to group spikes (nSites_spk
↳= 1 + maxSite*2)
um_per_pix = 20; % Vertical site center-to-center spacing (used for display)

% Shank information
shank = ones(size(channels)); % All sites belong to shank 1 (single shank recording)

```

Tags (variable names)

- `channels`: (1 x `nSites`), where `nSites` is the number of sites to load. The i th entry in `channels` denotes the channel (the row in the $n_{\text{channels}} \times n_{\text{samples}}$ matrix) in the raw recording containing data from site i . In the example above, channel 41 in the raw recording corresponds to site 1. This corresponds to the [siteMap](#) parameter.
- `geometry`: (`nSites` x 2). The location of each site, in μm . The first column corresponds to the horizontal or x dimension, and the second column corresponds to the vertical or y dimension (parallel to the probe shank). This corresponds to the [siteLoc](#) parameter.
- `pad`: (1 x 2). Dimensions (height, width) of the rectangular recording site, in μm . This corresponds to the [probePad](#) parameter.
- `ref_sites` (optional): (1 x k), where k is the number of reference sites. Indices of reference or disconnected sites to ignore. Alternatively, you can leave these sites out of your [siteMap](#) entry.
- `shank` (optional for single-shank probes): dimension: (1 x `nSites`). Shank ID for each site. For example, `shank = [1, 1, 1, 1, 2, 2, 2, 2]` will assign sites 1-4 to shank 1 and sites 5-8 to shank 2. This corresponds to the [shankMap](#) parameter.
- `maxSite` (optional): scalar. Number of neighboring sites in each direction over which to extract spike waveforms. This corresponds to the [nSiteDir](#) parameter.
- `nSites_ref` (optional): scalar. Total number of reference sites to exclude for feature extraction. This corresponds to the [nSitesExcl](#) parameter.
- `um_per_pix` (optional): scalar. Micrometers per center-to-center vertical site spacing. This corresponds to the [umPerPix](#) parameter.

Raw recording files

JRCLUST primarily supports the [SpikeGLX](#) recording format, namely a flat binary file containing 16-bit signed integers with a corresponding [metadata](#) file. ADC samples from channels 1 to n sampled at time k are stored together in series. For example, let $s_{i,j}$ denote the sample from the i th channel at time step j . Then the samples are ordered like so:

$$s_{1,1}, s_{2,1}, \dots, s_{n,1}, s_{1,2}, s_{2,2}, \dots, s_{n,2}, \dots, s_{n,k}$$

For more information, see the [SpikeGLX documentation on GitHub](#).

JRCLUST will ask for a path to your meta file when *bootstrapping* and use this to infer a path to your raw recording. If you don't have a meta file, don't panic! You will be able to select your raw recording, at which point you may specify the relevant information before you begin sorting.

Trial file

This can be either a MAT-file or a CSV. If a MAT-file, it should contain an array called `times`. If a CSV file, it should contain a single column. In either case, the array or the column should contain trial start times (in seconds). These can be used to plot the peristimulus time histogram (PSTH).

1.3.2 Output files

Filename	Content	Format
<code>mysession_res.mat</code>	<i>Detection and clustering results</i>	MATLAB data
<code>mysession_filt.jrc</code>	<i>Filtered spike traces</i> from a subset of channels	Binary (16-bit signed integer)
<code>mysession_raw.jrc</code>	<i>Raw spike traces</i> from a subset of channels	Binary (16-bit signed integer)
<code>mysession_features.jrc</code>	<i>Spike features</i>	Binary (single-precision float)
<code>mysession_hist.jrc</code>	<i>Operation history</i> from user curation	Binary (32-bit signed integer)
<code>mysession.csv</code>	<i>Cluster data</i>	CSV

Detection and clustering results

The following data are stored in `mysession_res.mat`:

Variable name	Content	Data format
<code>centerSites</code>	Sites with peak spike amplitudes for each of <i>nPeaksFeatures</i>	<code>nSpikes x nPeaksFeatures: int32</code>
<code>clusterCenters</code>	Indices of cluster site centers as determined in <i>Spike clustering</i>	nClusters x 1: double
<code>clusterCentroids</code>	Feature-weighted center x-y positions on the probe for each cluster	<code>nClusters x 2:</code> double
<code>clusterNotes</code>	Text annotations for each cluster	<code>nClusters x 1:</code> cell array of char
<code>clusterSites</code>	Mode of <code>spikeSites</code> for spikes in each cluster	<code>nClusters x 1:</code> double
<code>curatedOn</code>	Timestamp of last saved curation	scalar double
<code>detectTime</code>	Time spent (in seconds) in detection step	scalar double
<code>detectedOn</code>	Timestamp of last spike detection	scalar double

Continued on next page

Table 1 – continued from previous page

Variable name	Content	Data format
featuresShape	Dimensions of <i>Spike features</i>	1 x 3: double
filtShape	Dimensions of <i>Filtered spike traces</i>	1 x 3: double
history	Key-value store of curation operations with commit messages (see <i>Operation history</i>)	containers.Map
initialClustering	Initial spike table	nSpikes x 1: int32
meanSiteThresh	Mean (over chunks) detection threshold per site	1 x nSites: single
meanWfGlobal	Mean (over spikes/cluster) waveform over <i>all</i> sites (all zeros outside siteNeighbors for Reasons)	nSamples x nSites x nClusters: single
meanWfGlobalRaw	Mean (over spikes/cluster) <i>raw</i> waveform over <i>all</i> sites (all zeros outside siteNeighbors for Reasons)	nSamplesRaw x nSites x nClusters: single
meanWfLocal	Mean (over spikes/cluster) waveform over sites specified in siteNeighbors	nSamples x 2 * nSiteDir + 1 x nClusters: single
meanWfLocalRaw	Mean (over spikes/cluster) <i>raw</i> waveform over sites specified in siteNeighbors	nSamplesRaw x 2 * nSiteDir + 1 x nClusters: single
meanWfRawHigh	Mean (over spikes/cluster) <i>raw</i> waveform on <i>high</i> positions on the probe	nSamplesRaw x 2 * nSiteDir + 1 x nClusters: single
meanWfRawLow	Mean (over spikes/cluster) <i>raw</i> waveform on <i>low</i> positions on the probe	nSamplesRaw x 2 * nSiteDir + 1 x nClusters: single
rawShape	Dimensions of <i>Raw spike traces</i>	1 x 3: double
siteThresh	Detection threshold per site per chunk (see <i>Chunking</i>)	nSites x nChunks: single
sortTime	Time spent (in seconds) in clustering step	scalar double
sortedOn	Timestamp of last spike clustering	scalar double
spikeAmps	Spike amplitudes in ADC sample unit	nSpikes x 1: int32
spikeClusters	The spike table: cluster assignment for each spike	nSpikes x 1: int32
spikePositions	Feature-weighted center x-y positions on the probe for each spike	nSpikes x 2: double
spikeSites	Site with the peak spike amplitude (center site)	nSpikes x 1: int32
spikeSites2	Site with the second peak spike amplitude	nSpikes x 1: int32
spikeSites3	Site with the third peak spike amplitude	nSpikes x 1: int32
spikesByCluster	Cell of the spike indices per cluster	nClusters x 1: cell array of double
spikesBySite	Cell of the spike indices per site	nSites x 1: cell array of int32

Continued on next page

Table 1 – continued from previous page

Variable name	Content	Data format
spikesBySite2	Cell of the spike indices per secondary site	$n_{\text{Sites}} \times 1$: cell array of int32
spikesBySite3	Cell of the spike indices per tertiary site	$n_{\text{Sites}} \times 1$: cell array of int32
spikeTimes	Spike timing in ADC sample unit	$n_{\text{Spikes}} \times 1$: int32
unitCount	Count of spikes in each cluster	$n_{\text{Clusters}} \times 1$: double
waveformSim	Waveform-based pairwise cluster <i>sim</i> scores	$n_{\text{Clusters}} \times n_{\text{Clusters}}$: double

Filtered spike traces

A binary file containing filtered spike traces, extracted in a spatiotemporal window around the spiking event. These are stored as an $n_{\text{samples}} \times n_{\text{sites}} \times n_{\text{spikes}}$ array of signed (16-bit) integers.

n_{samples} is determined from *evtWindow* and n_{sites} is determined from *nSiteDir*.

Raw spike traces

A binary file containing raw spike traces, extracted in a spatiotemporal window around the spiking event. These are stored as an $n_{\text{samples}} \times n_{\text{sites}} \times n_{\text{spikes}}$ array of signed (16-bit) integers.

n_{samples} is determined from *evtWindowRaw* and n_{sites} is determined from *nSiteDir*.

Spike features

A binary file containing spike features. For each spike, JRCLUST computes some number of features at one or more positions in a spatiotemporal window. Consequently, the spike features data is stored as an $n_{\text{features}} \times n_{\text{positions}} \times n_{\text{spikes}}$ array of single-precision (32-bit) floating point values.

Operation history

A binary file containing a copy of the spike table for each operation performed during *Cluster curation*. For each operation, JRCLUST writes an operation ID (a 32-bit signed integer), followed by the spike table, at the end of this file. The operation ID is kept in sync with a key-value map (`hClust.history`), allowing users to revert operations.

Cluster data

A CSV export of spike-cluster data. For each spike, a row is written containing: - spike time - cluster ID - center site ID

1.4 Getting started

You've got a recording file and you've got MATLAB. Now what? If you've used an older version of JRCLUST, you might want to check out the *migrating* page to see what's changed from the old JRCLUST. If you're completely new to JRCLUST, start with the *tutorial*.

1.5 System requirements

1.5.1 Software Requirements

JRCLUST 4 was built and tested with **MATLAB R2017b** on Microsoft **Windows 10**. If you are running another operating system or an earlier version of MATLAB, JRCLUST may not work as expected. Post an issue on our [issues page](#) with the error message and we'll get it fixed.

You will also need the [CUDA Toolkit](#). See [this link](#) to determine which CUDA Toolkit version you will need for your version of MATLAB. It's best to keep your GPU driver updated as well.

In addition, the following toolboxes are **required**:

- [Parallel Computing](#)
- [Image Processing](#)
- [Signal Processing](#)
- [Statistics and Machine Learning](#)

1.5.2 Hardware Requirements

An NVIDIA GPU with [compute capability](#) 3.5 or later is strongly recommended. In addition, we also recommend you have at least 32 GiB of memory.

1.6 Usage

In general, usage goes like

```
jrc COMMAND REQUIRED\_ARG1 REQUIRED\_ARG2 [OPTIONAL\_ARGS ...]
```

where COMMAND is described below.

1.6.1 Documentation and help

- `jrc help`: Display a help menu.
- `jrc version`: Display the version number.
- `jrc about`: Display program information.

1.6.2 Pipeline commands

- `jrc importv3 mysession_jrc.mat`: Import a JRCv3 `_jrc.mat` file to the new format. **Does not overwrite your old results.**
- `jrc bootstrap`: Create a [config file](#).
- `jrc detect myparams.prm`: Perform spike detection and feature extraction. (Replace `myparams` with a unique name of your choosing.) Output files:
 - `myparams_res.mat`: A MAT-file containing detection results
 - `myparams_filt.jrc` (filtered spike traces)

- `myparams_raw.jrc` (raw spike traces)
- `myparams_features.jrc` (computed features)
- `myparams_hist.jrc` (operation history)

See [Input and output files](#) for details.

- `jrc sort myparams.prm`: Cluster detected spike features. If JRCLUST can't find any detected spikes, it will also perform the detect step. Your `myparams_res.mat` file will be updated with a [clustering handle](#) object.
- `jrc detect-sort myparams.prm`: Detect and cluster spikes. The difference between `sort` and `detect-sort` is that `detect-sort` will overwrite any previously-detected spikes, whereas `sort` will only perform spike detection if JRCLUST can't find any previously-detected spikes.
- `jrc recluster myparams.prm`: Recluster spikes. You might want to do this if, e.g., you've updated the clustering threshold parameters but don't want to recompute rho and delta.
- `jrc manual myparams.prm`: Open the manual curation GUI.

1.6.3 Preview commands

- `jrc probe {myprobe.prb, myparams.prm}`: Plot probe layout. You may specify either a config file or a probe file directly.
- `jrc traces myparams.prm [File #]`: Display traces from your raw recordings. If you have multiple recordings, you may specify the index of the file to display from.
- `jrc activity myparams.prm`: Plot firing rate as a function of time and depth.

1.6.4 Missing commands

Many other commands were available with previous versions of JRCLUST. These have been disabled because, as far as we know, they were not widely used. If you're missing a command, post an issue on our [issues page](#) request and we will correct our error.

1.7 Getting help

Have questions? Check out our support channel on [Gitter](#). If you run into any bugs or have any feature requests, [create an issue](#) on our GitHub [issues page](#).

The JRCLUST spike-sorting pipeline

The JRCLUST pipeline can be broken down into the following steps:

1. config file creation (*bootstrapping*)
2. *spike detection/feature extraction*
3. *clustering*
4. *manual curation* of automatic results

2.1 Bootstrapping

Your config file encapsulates the choices in parameters that you make, as well as describing the relevant probe configuration. You can create a config file by invoking `jrc bootstrap` or `jrc bootstrap /path/to/metafile.meta` from the MATLAB command window.

If you don't specify a [SpikeGLX meta file \(.meta\)](#), you will be asked to select one. This will collect recording-specific information from your meta file and set default parameters. (The default parameters may be inspected on the [JRCLUST parameters](#) page.) The location of your raw recording will be inferred from your meta file, so be sure they are similarly named (this is the SpikeGLX default) and placed together in the same directory!

If you don't have a .meta file or your .meta file is missing some data, JRCLUST requests the following information:

- **Sampling rate (Hz):** read from `imSampRate` (or `niSampRate` for NI recordings) in .meta file
- **Number of channels in file:** read from `nSavedChans` in .meta file
- **μ V/bit:** computed from `imAiRangeMax`, `imAiRangeMin` (or `niAiRangeMax`, `niAiRangeMin`, and `niMNGain` for NI recordings) in .meta file
- **Header offset (bytes):** set to 0 for SpikeGLX recordings since no header is stored in the .bin file
- **Data type:** select from `int16`, `uint16`, `single`, or `double` (SpikeGLX files are saved as `int16`)

You will also be asked to confirm your config filename and path to your raw recording. If you have multiple recordings, full paths will be separated by commas.

2.2 Spike detection

Once you have a config file, you can detect spikes in your recording. Any of the following commands will **detect** spikes:

- `detect` will perform spike detection/feature extraction and save results to disk.
- `detect-sort` will do all of the above and additionally cluster the spikes.
- `full` is the same as `detect-sort`, but will also pull up the curation GUI after clustering is completed.

(See the *usage section* for how to invoke these commands.)

For each recording you specify in your config file, JRCLUST will:

1. *Denoise* and *filter* samples. Also perform *common-average referencing* on the filtered samples.
2. Compute a *detection threshold* from the filtered samples (if you have not already supplied a threshold).
3. *Detect peaks*, i.e., points exceeding the threshold which are also genuine turning points.
4. *Merge peaks* detected at multiple neighboring sites, searching over a spatiotemporal window. Where larger peaks are detected within this limit, weaker peaks are removed.
5. *Extract spatiotemporal windows* around spiking events, in both raw and filtered samples.
6. *Compute low-dimensional features* from the resulting waveforms.

A detailed description of the steps in the detection process is indexed below.

2.2.1 Spike detection

Chunking

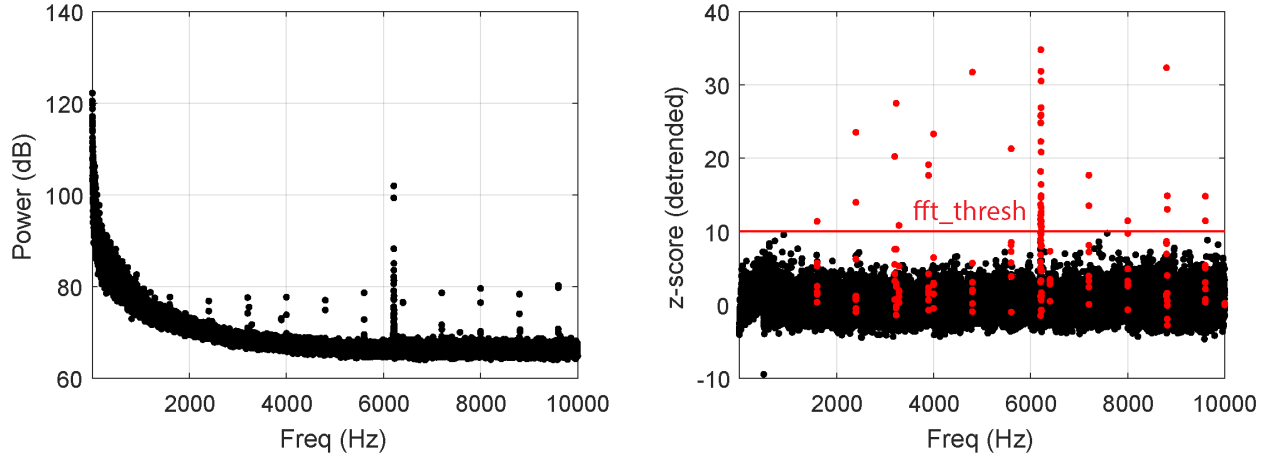
JRCLUST will chunk up large files and perform each of the following steps in sequence over each chunk. If you are using a GPU, the chunk size is determined from your available GPU memory. For consistency, **3/4** of the total GPU memory will be considered “available”. If you are not using a GPU, JRCLUST will try to determine the total amount of RAM available on your system and consider all of it to be available. JRCLUST will then take *a fraction* of that available memory and reserve it for processing. Large files are chunked to use as much of the available memory as possible.

Denoising

JRCLUST constructs an adaptive notch filter which cleans narrow-band noise peaks exceeding a threshold you supply *threshold you supply*. (Set this parameter (`fftThresh`) to 0 to skip this step.)

The time-domain signal is converted to the frequency domain via fast Fourier transform (FFT). After this, the average power across channels is computed. Power is (putatively) detrended using $\frac{1}{f}$ relationship in frequency vs. power. For each frequency bin, the power-frequency product is normalized by its *MAD*, and those frequencies whose MAD power-frequency products exceed this threshold are zeroed out. The cleaned frequency-domain signal is then transformed back to the time domain via inverse FFT.

The figure below shows the original (left) and detrended (right) plots. Red indicates FFT coefficients that exceed threshold or points that are within 3 samples from the outlier points.



Filtering

This step filters the raw samples depending on the *filter type* you specify. Additionally, JRCLUST computes the common average reference and subtracts it from the filtered samples. (The meaning of “average” in “common average reference” is controlled by the *CAR mode* you specify.)

Computing detection thresholds

Assuming a Gaussian noise distribution on every site, we estimate the standard deviation $\sigma_{\text{noise}}^{(i)}$ of the background on site i by

$$\text{med}\sigma_{\text{noise}}^{(i)} = \left(\frac{|x^{(i)}|}{0.6745} \right)$$

where $x^{(i)}$ is the filtered signal on site i . (See [here](#) for justification.) A single *multiple* of $\sigma_{\text{noise}}^{(i)}$ that you specify serves as the detection threshold on each site. In other words, each site has a distinct threshold which is a multiple of an estimate of the standard deviation of its own noise distribution.

Detecting peaks

Samples on each site are compared to the threshold Thr_i for that site. Those samples with negative value but exceeding the threshold in magnitude are considered putative peaks. (If you *specify*, then JRCLUST can also detect peaks with positive value.) A putative peak must also exceed its immediate neighbors in magnitude, i.e., the sample at time t_j is a peak if and only if the following conditions hold:

$$\begin{aligned} \text{amp} \quad & |(t_j)| > \text{Thr} \\ & |(t_j)| > |(t_{j-1})| \\ & |(t_j)| > |(t_{j+1})| \end{aligned}$$

You may also impose *additional constraints* on peak detection.

Merging peaks

Peak samples are compared with neighbors in a spatiotemporal window around each peak to detect duplicates. If peaks are found within this window, then the following operations are performed:

1. If a neighboring peak has a larger amplitude, then discard the center peak.
2. If a neighboring peak has the same amplitude but occurs first, then discard the center peak.
3. If a neighboring peak has the same amplitude and occurs at the same time, then discard the peak occurring on the higher-indexed site.

Extract spatiotemporal windows around spiking events

For each spiking event, JRCLUST will save raw and filtered samples as $n_{\text{sites}} \times n_{\text{samples}}$ matrices. n_{sites} is `nSitesEvt` (determined from `nSiteDir`), and n_{samples} depends on `evtWindow` and `evtWindowRaw` for filtered and raw samples, respectively. Each of these matrices is stored in an $n_{\text{sites}} \times n_{\text{samples}} \times n_{\text{spikes}}$ array and are used for feature extraction, automatically merging clusters, and display in the curation GUI.

At this point, *if you specify*, JRCLUST may either subtract a local common average reference from the windows and realign the spike waveforms; or interpolate the spike waveforms between samples to determine if the true peak maxima are at subpixels, taking the interpolant if this is found to be the case.

Compute features from extracted waveforms

JRCLUST subtracts the local common-average reference from the spike waveforms extracted above and then computes features from these according to the *feature you specify*. For more detailed descriptions of your options in features, see below.

Additionally, *if you specify*, JRCLUST finds an alternate location for each spike within that spike's spatiotemporal window. JRCLUST will extract a window around this secondary peak in order to compute features. This is done in order to accommodate necessary alterations to the clustering algorithm due to resource constraints.

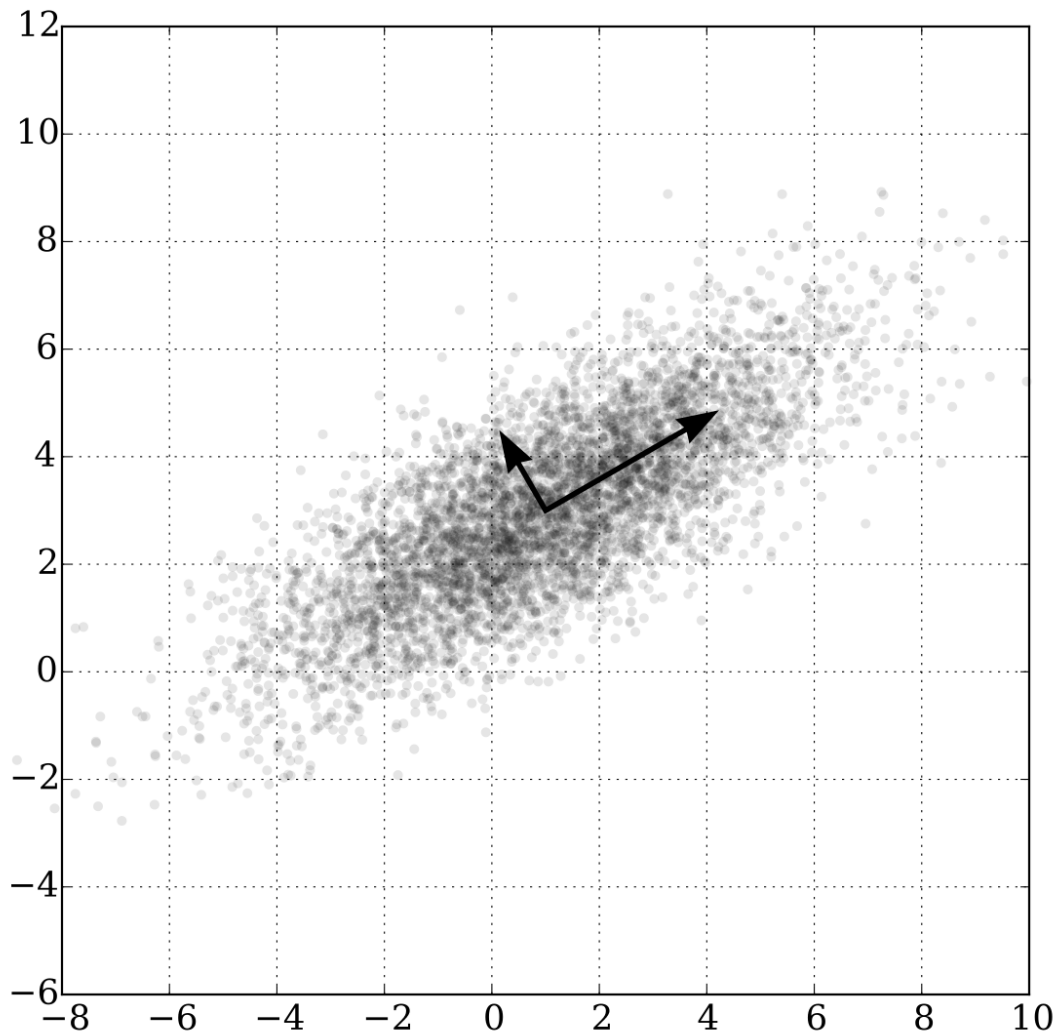
Features for clustering

The following features are available for clustering your spikes.

Principal components analysis

Principal components analysis, or PCA, is a method for reducing the individual observations (in our case, a spatiotemporal window around a spiking event) to just a few components. PCA constructs an orthogonal basis in such a way that the first basis vector explains as much of the variance in the data as possible, and each subsequent basis vector explains less variance, but more than any of the following basis vectors.

The diagram below illustrates the general idea:



Shown is a sample from a bivariate Gaussian distribution with the first two principal components superimposed on the data points (courtesy of Wikipedia). The larger vector describes the direction of most variance, while the smaller (orthogonal) vector describes less than the first. If we were to project each point onto the first principal component, we would have reduced the dimensionality of the points in this distribution from 2 to 1. This is a trivial example, but if we extend the 2-dimensional Gaussian to n dimensions we see how useful PCA can be in dimensionality reduction.

If you select 'pca' as your *clusterFeature*, then JRCLUST computes principal components from a random subsample of at most 10,000 of the *traces* of detected spikes in a given *chunk of data*. All spike traces on each site in the event radius are then projected onto the first (or first 2 or 3) principal vectors, giving us 1 (or 2, or 3) points per site per spike, instead of the entire waveform.

Additionally, *if you specify*, JRCLUST will interpolate the first principal component in order to maximize the projection of the spike traces onto that component.

Global PCA

Global PCA describes the same procedure as PCA, [above](#), but instead using a single common set of principal vectors computed from the first chunk, rather than computing a set of principal vectors for each chunk. You can use global PCA as your feature by setting your *clusterFeature* to 'gpca'.

Vmin

This feature projects the traces in the event window of each spike onto their minimum values.

Vminmax

This feature projects the traces in the event window of each spike onto their minimum **and** maximum values. This gives 2 features per site per spike.

Vpp

This feature projects the traces in the event window of each spike onto their minimum and maximum values, taking the distance between them as the feature.

2.3 Spike clustering

After the spiking events have been detected, they must be clustered by the features extracted from them. Any of the following commands will **sort** spikes:

- `sort` will cluster spiking events using spikes you have detected previously. If JRCLUST can't find a previous detection, it will also detect them for you.
- `detect-sort` will cluster spiking events after detecting them. If you have previously detected spikes, JRCLUST will overwrite them.
- `full` is the same as `detect-sort`, but will also pull up the curation GUI after clustering is completed.

(See the [usage section](#) for how to invoke these commands.)

JRCLUST will cluster the spike features using a variant of the clustering algorithm of [Rodriguez and Laio](#), also known as density-peak clustering or ρ - δ clustering. The general algorithm computes pairwise distances in the feature space \mathbb{R}^n , and, given a cutoff distance d , assigns to each point x_i in the feature space a *density*

$$\rho_i := \sum_j I(\|x_i - x_j\|_2 < d),$$

where I is the indicator function, giving 1 if the condition is true and 0 otherwise. In plain English, ρ_i is the number of points within a ball of radius d centered at x_i .

Once each point has been assigned a density, we then find the distance to the nearest neighbor of higher density,

$$\delta_i := \min_{\rho_i < \rho_j} \|x_i - x_j\|_2.$$

(If there is no j such that $\rho_i < \rho_j$, i.e., x_i is a maximally dense point, then δ_i is defined to be $\max_j \|x_i - x_j\|_2$.)

If the point x_i is both sufficiently dense (i.e., ρ_i is large enough) **and** sufficiently far from any other point with higher density, then x_i is deemed a *cluster center*, and all points with x_i as nearest neighbor of higher density will be assigned to the cluster centered around x_i .

We have left the terms *sufficiently dense* and *sufficiently far* somewhat vague, to say nothing of how the cutoff distance d is determined. These, along with the variations JRCLUST makes to accommodate the large volume of data, are elaborated [here](#).

2.3.1 Spike clustering

Adjustments to the algorithm

Because many millions of spikes may be detected in a single recording and the distance matrix grows as $O(n^2)$, we quickly run out of memory. Instead, we take advantage of the fact that two spikes which are far away from each other in space and in time are unlikely to be from the same neuron and thus should not belong to the same cluster. So instead of computing the distance between a spike and *every* other spike, we restrict our comparison of spikes on site K to spikes which either also peak on site K , or have a *secondary peak* on site K . Additionally, *if you specify*, we also restrict our comparison to spikes occurring nearby in time (you might want to do this if you observe drift in your recording). This has a tendency to oversplit clusters, but they can be *merged later*.

Determining d

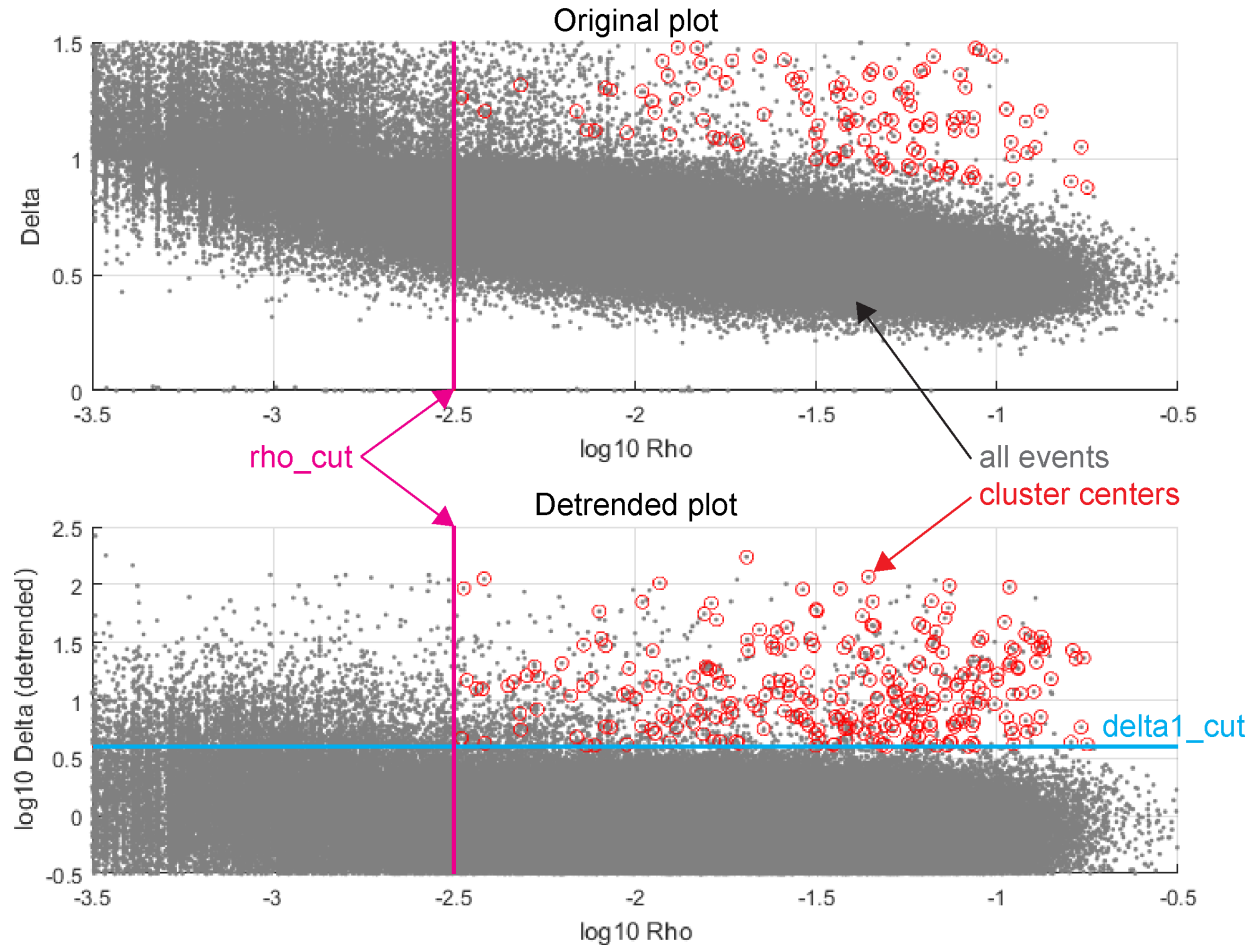
The best cutoff distance in feature space is generally not known *a priori*. In order to determine a good cutoff distance, we instead estimate a *percentile you specify* (the 2nd percentile by default), call it d_i , of the pairwise distances between spike features on each site i . We then take the median of the d_i as our global cutoff distance.

(This behavior can be overridden by setting the parameter *useGlobalDistCut* to 0 in your config file.)

Assigning clusters: “sufficiently dense” and “sufficiently far”

Once every spike has its ρ and δ scores, along with the nearest neighbor of higher density corresponding to the δ score, we then determine cluster centers.

You must *specify* the *thresholds* to determine a cluster center. In particular, a spike i must have $\log_{10}(\rho_i) > \log_{10}\text{RhoCut}$ (“sufficiently dense”) and $\log_{10}(\delta_i) > \log_{10}\text{DeltaCut}$ (“sufficiently far”). If *you specify*, this comparison may be made against a detrended plot, as in the images below.



Once cluster centers are assigned, it is a straightforward operation to assign spikes to clusters: for each spike which has not been assigned a cluster (initially, this is all spikes which are not cluster centers), assign it to the same cluster as its nearest neighbor of higher density. Most spikes will remain unassigned at first, but on each iteration of this procedure, the number of spikes which are unassigned will shrink until all spikes have been assigned to clusters.

Note: Because we compared primary peaks with secondary peaks, a spike may have as its nearest neighbor a spike on a nearby site.

Merging possibly oversplit clusters

After the initial clustering using, units with similar mean raw waveforms are merged based on their similarity, as computed using the Pearson correlation coefficient of the waveforms. Since probe drift changes the spike waveforms, averaged waveforms are computed at three depth ranges per unit using their center-of-mass positions. Additionally, a time shift is applied between each pair to find the maximum cross-correlation value. The similarity score is taken as the maximum score from the resulting 3×3 correlation matrix for each unit pair. Any cluster pair whose maximum similarity exceeds a threshold you specify is merged.

This process is repeated some number of times and the resulting clustering is output for inspection.

2.4 Cluster curation

Manual verification and correction of the automatic clustering can be done using the GUI curation tool.

Any of the following commands will **curate** spike clusters:

- `manual` will pull up the curation GUI. If JRCLUST can't find a previous clustering, it will ask if you want to cluster your spikes.
- `detect-sort` will do all of the above and additionally cluster the spikes.
- `full` will detect and cluster spikes, pulling up the curation GUI after clustering is completed. If you have previously detected spikes, JRCLUST will overwrite them.

(See the [usage section](#) for how to invoke these commands.)

In addition to deleting, splitting, or merging clusters, you may also annotate them as noise, MUA (multi-unit activity), or provide arbitrary notes on clusters. For a good primer on why you might want to do these things, take a look at [Phy's documentation](#). A detailed description of the different views onto the data is indexed below.

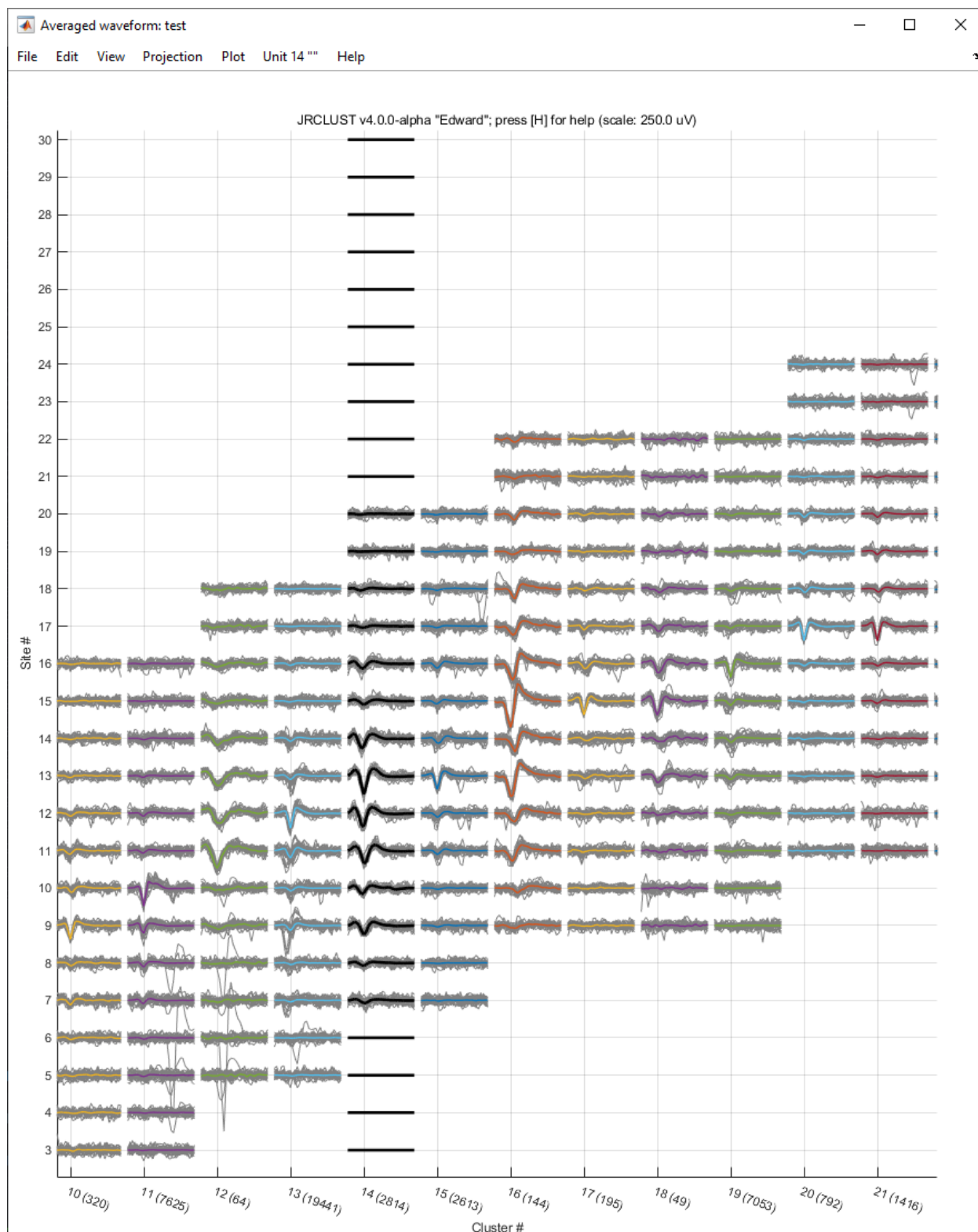
2.4.1 Cluster curation

As a user, you will want to inspect your clusters as well as make corrections to the automatic [detection](#) and [clustering](#) done by JRCLUST. This is done via the manual curation GUI, which is invoked like so:

```
jrc manual /path/to/your/configfile.prm
```

This GUI allows you to annotate (i.e., add a note to), delete, or split clusters, as well as merge two clusters. To accomplish this, a number of figures are displayed. The figures to show are specified in the [figList](#) parameter, but it is recommended to leave this parameter alone. With any of the interactive figures, you may press the **h** key to display a small help dialog.

The waveform view (FigWav)

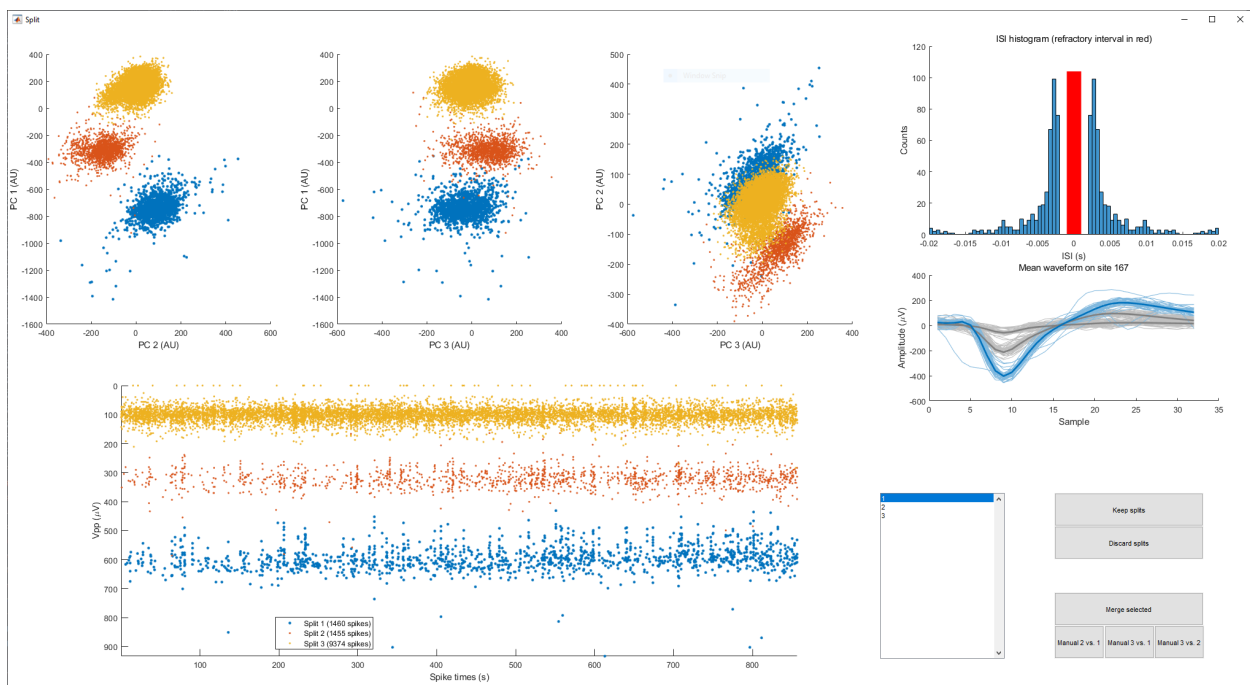


The waveform view is the main view on your data. Each cluster is ordered by its center site and the mean unit waveform is plotted for each site in a spatial neighborhood around the center. In the background, plotted in gray, are

some randomly-sampled individual waveforms. (You can toggle these off and on by pressing the **w** key and resample them with the **a** key.) You can select clusters by clicking their waveforms. The selected cluster will be highlighted in black. Right-clicking another cluster will cause that cluster to be highlighted in red and update the other figures with data from each cluster for comparison. (Select the next most similar cluster by pressing the **spacebar**). You may pan by holding the **Shift** key and clicking and dragging, or zoom in and out with the scroll wheel. The **left** and **right arrows** will select the previous and next clusters, respectively. The **up** and **down arrows** adjust the scale of the waveforms. Pressing **r** will zoom out and show all clusters, while pressing **z** will zoom in on the selected cluster. If you have a *trial file*, pressing **p** will plot the peristimulus time histogram of the selected cluster. To delete the selected cluster, press **d**. To split the selected cluster, press **s**. To merge the selected pair of clusters, press **m**.

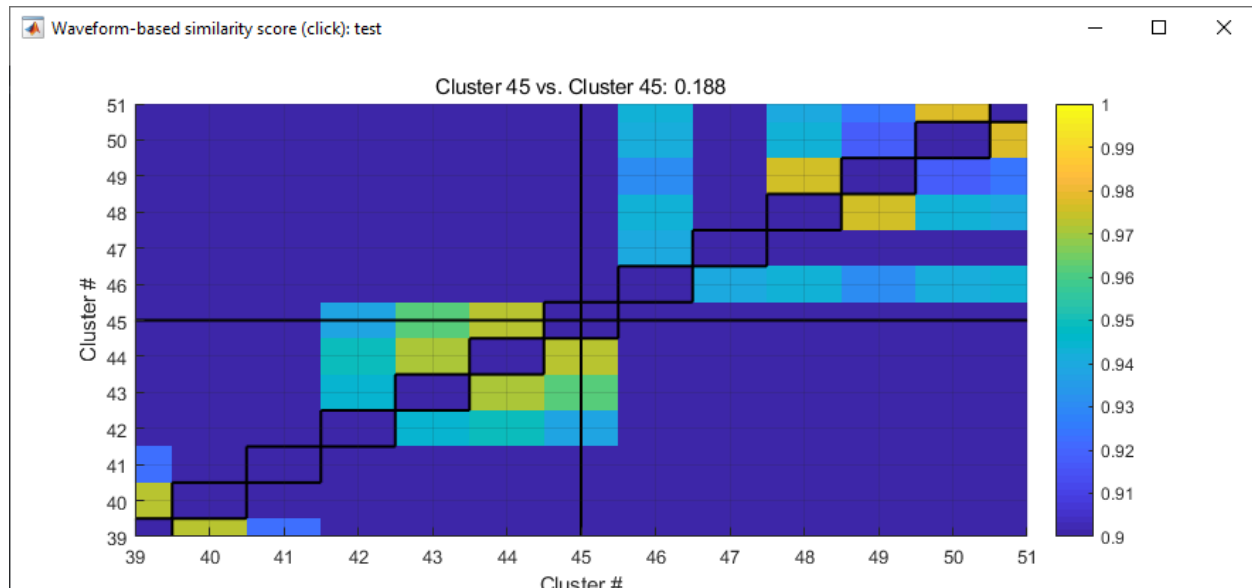
The waveform view also contains the menu system. Any time a reference is made to a menu entry in this GUI, that menu is here.

Splitting clusters (FigSplit)



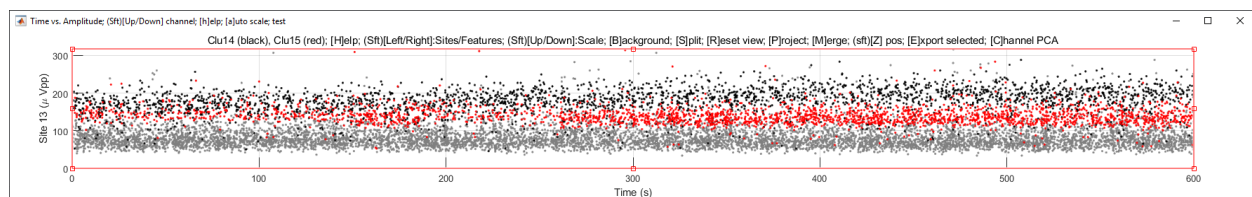
Two or more clusters may have been erroneously merged together by the algorithm. You can correct this by selecting the cluster in the waveform view and either pressing ‘s’ in the waveform view or selecting one of the *Split* options from the main menu. JRCLUST will then prompt you for the number of clusters you believe this cluster should be split into, and then perform hierarchical clustering on a set of features computed from the spikes in this cluster, using [Ward’s minimum variance method](#). You may view projections of spikes onto the 1st, 2nd, and 3rd principal components, a μV peak-to-peak vs. time plot, an ISI histogram, and mean waveforms for each split or combination of splits. You may also choose to manually split in any of the PC spaces or merge splits together. When you are satisfied, you can select “Keep splits” and your cluster will be split. If you change your mind you can select “Discard splits” and your cluster will be unaffected.

The similarity view (FigSim)



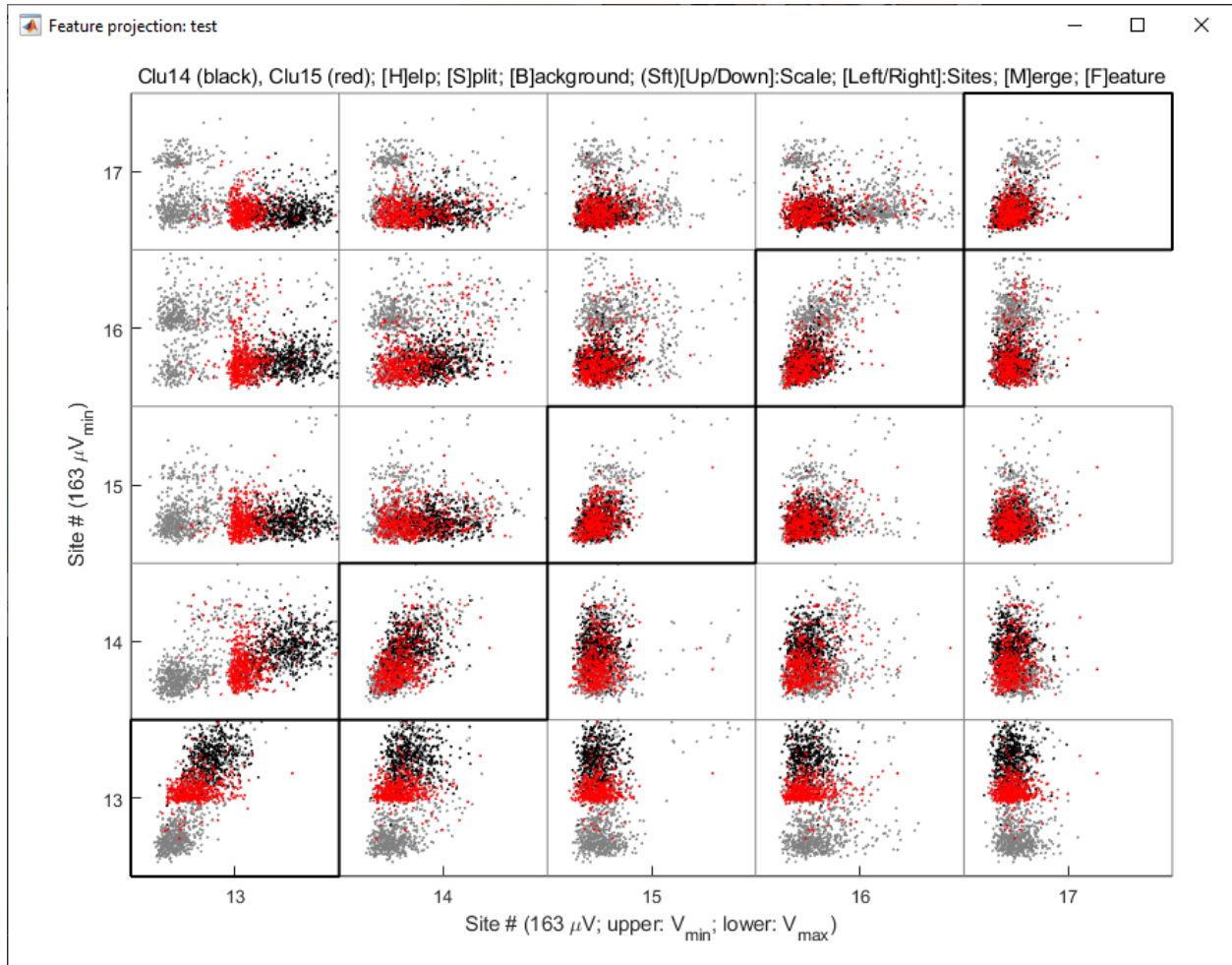
The similarity view tells you how similar each cluster is to every other. (Diagonal entries show self-similarity, lower-amplitude to higher-amplitude spikes.) Similarity is *computed at clustering* and every time the spike table changes. You may choose to automatically merge clusters whose similarity exceeds a given threshold (the default is 0.98) by selecting **Merge auto** from the Edit menu and supplying a different threshold. You may pan by holding the **Shift** key and clicking and dragging, or zoom in and out with the scroll wheel. The selected cluster is indicated by a crosshair at the center of a rectangle. To delete the selected cluster, press **d**. To split the selected cluster, press **s**. If two clusters are selected, then the crosshair will be red on the horizontal. To merge the selected pair of clusters, press **m**.

The feature-vs-time view (FigTime)



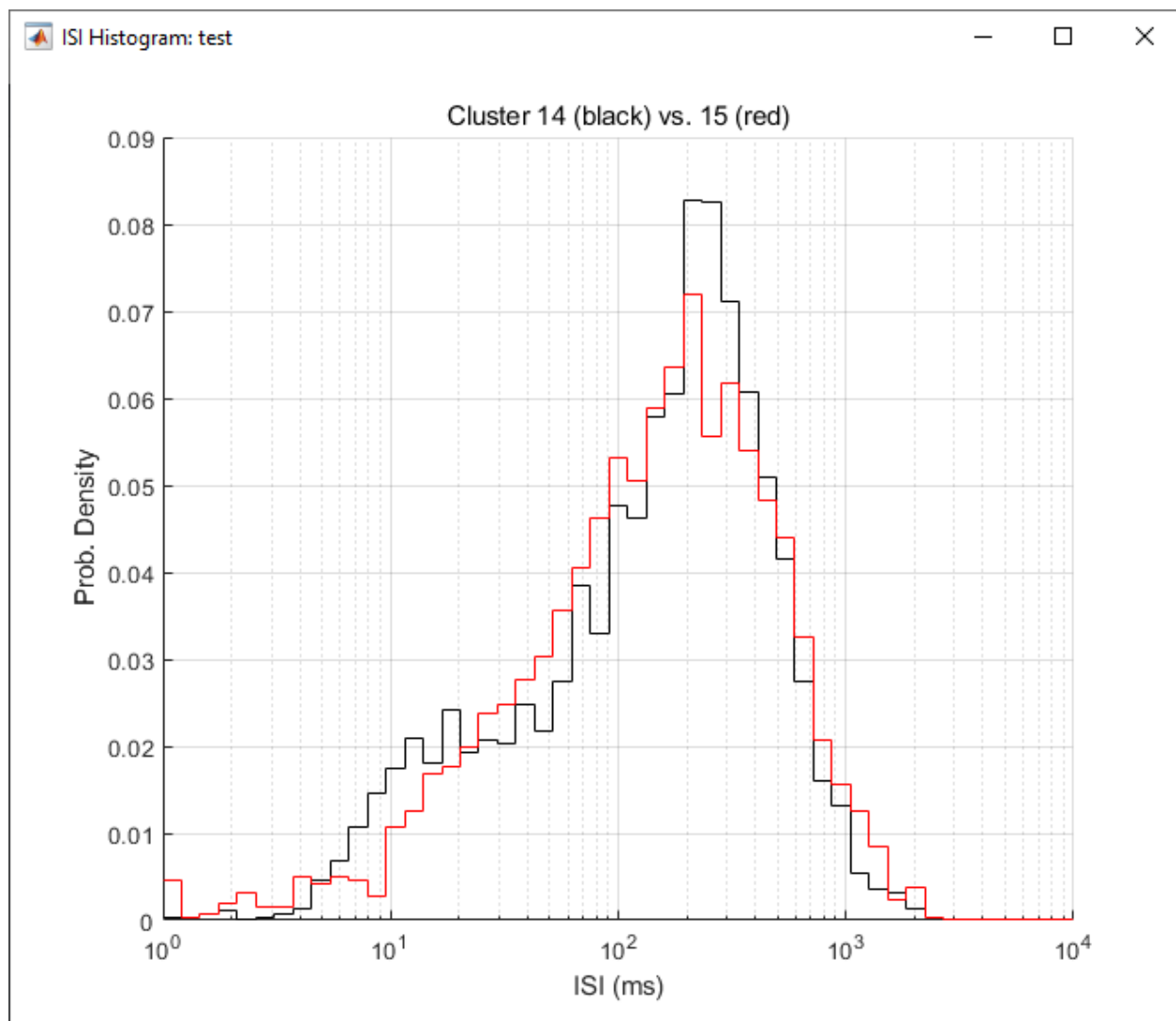
The feature-vs-time view displays a feature (by default the *peak-to-peak amplitude* (or Vpp), in μV) plotted against time (in seconds) on a given site. When selecting a unit, this feature is plotted on the center site of that unit, but you may change sites by pressing the **left** or **right arrows**, or adjust the scale with the **up** and **down arrows**. Press **r** to reset the view. The features shown for the selected cluster are in black, with background features in gray (toggle background features with the **b** key). If you select another cluster via the waveform or similarity views, features for the other cluster on that site will be shown in red. You can switch between the Vpp and PCA by pressing **f** in this view. If two clusters are selected, you may merge them with **m**. If one cluster is selected, you may split it by pressing **s** and drawing a polygon around the points in the new cluster to split off.

The feature projection view (FigProj)



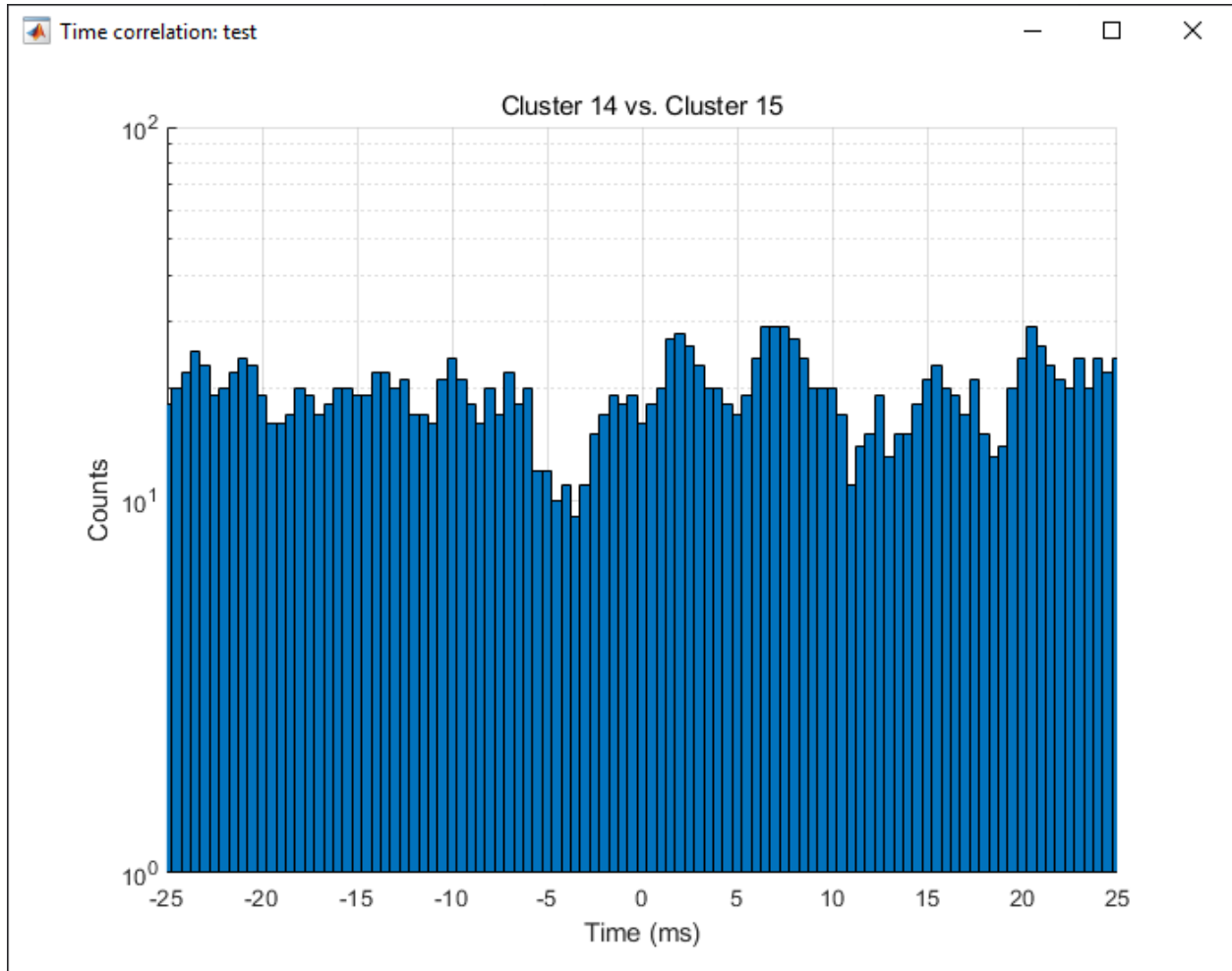
The feature projection view displays one feature vs. another on an adjacent group of sites. By default, these features are the minimum vs. maximum amplitudes on and below the diagonal, and minimum vs. minimum amplitudes above the diagonal. You may toggle the feature to display by pressing the **f** key, or by selecting one from the Projection menu. If showing the PCA feature, you may switch between PC2 vs. PC1, PC3 vs. PC1, or PC3 vs. PC2 with the **p** key. If two clusters are selected, you may merge them with **m**.

The ISI histogram view (FigHist)



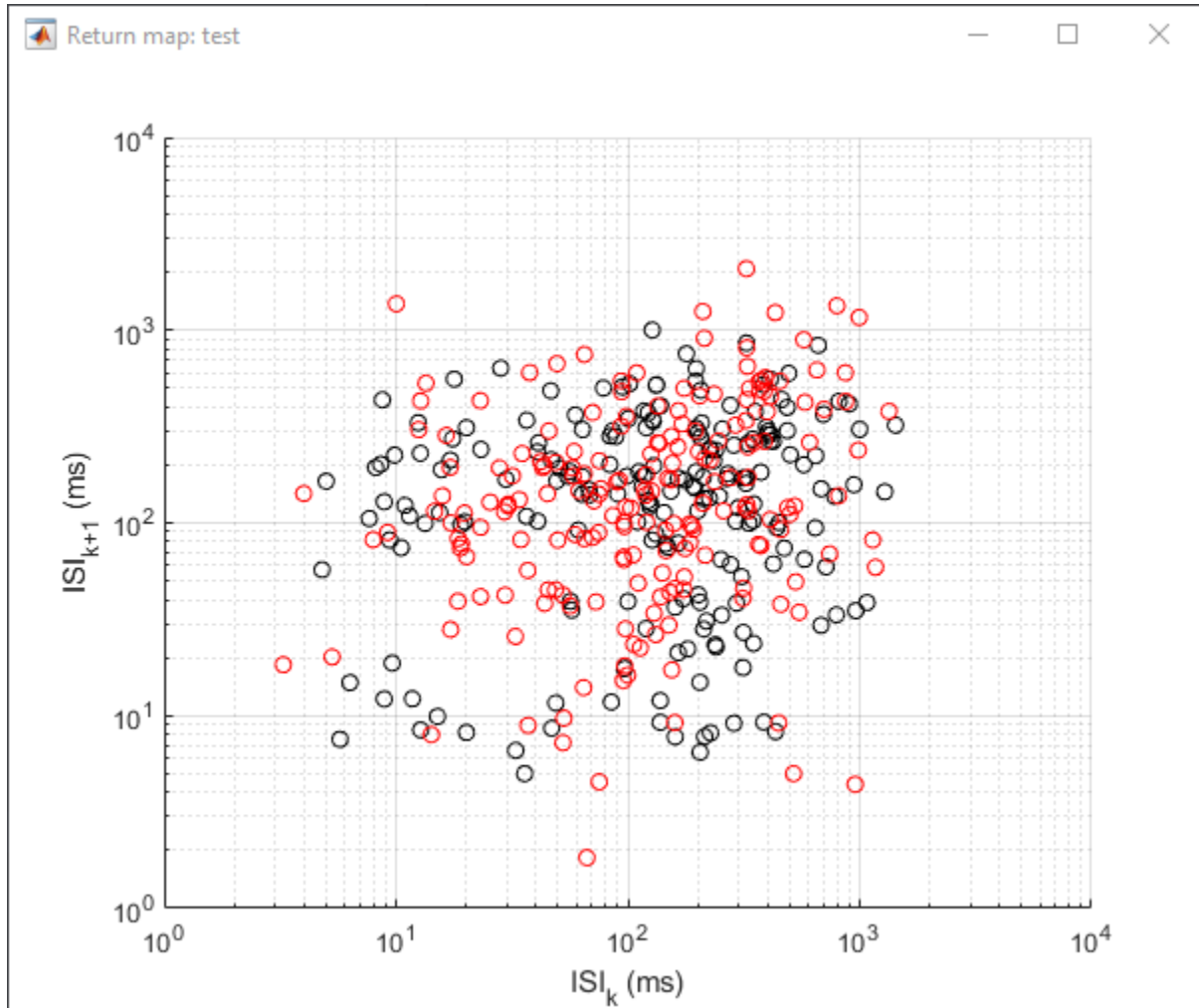
The ISI histogram shows a histogram of interspike intervals, i.e., intervals between firings (in ms) in the selected cluster.

The time correlation view (FigCorr)



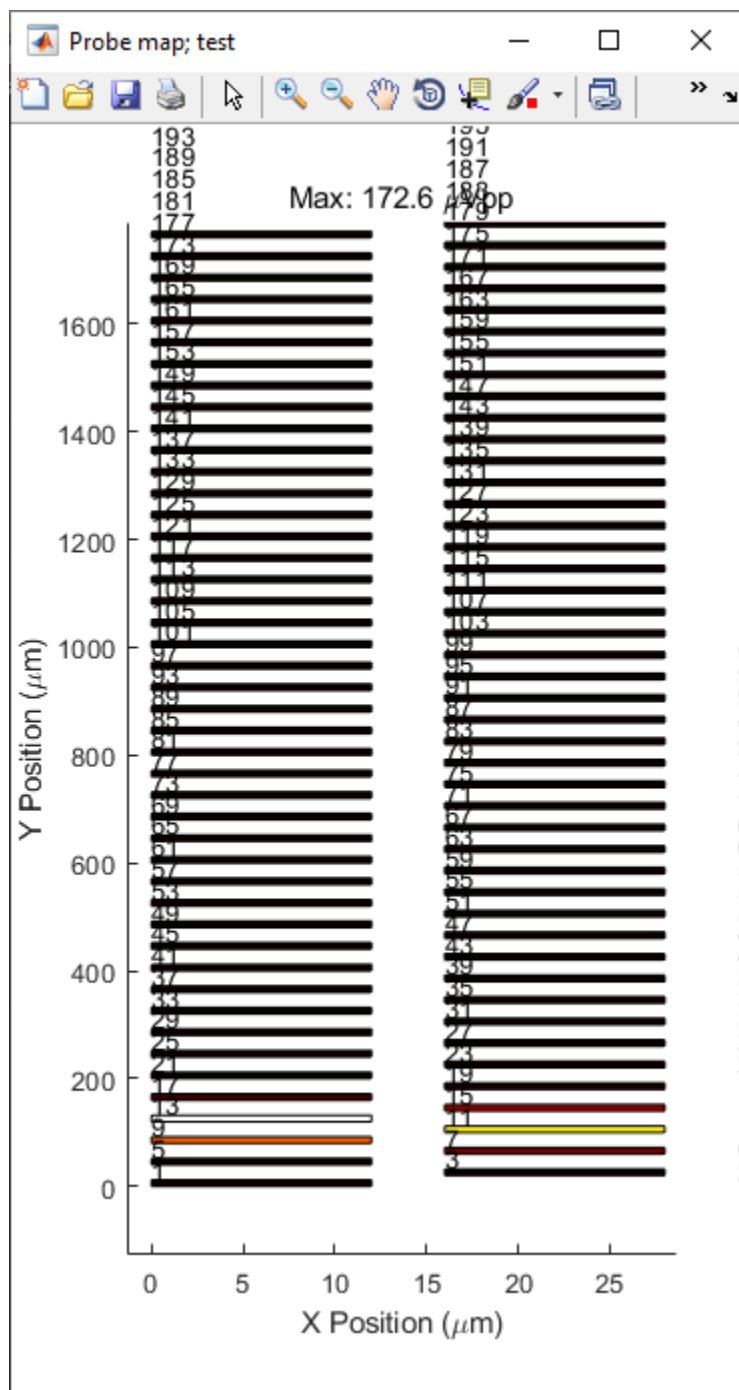
The time correlation view shows a count of spike firings at time lags of -25 ms to 25 ms, in 1/2 ms bins. If more than one cluster is selected, then the reference cluster is the primary selected cluster, and time lags are measured with respect to spikes in the reference cluster.

The return map view (FigISI)



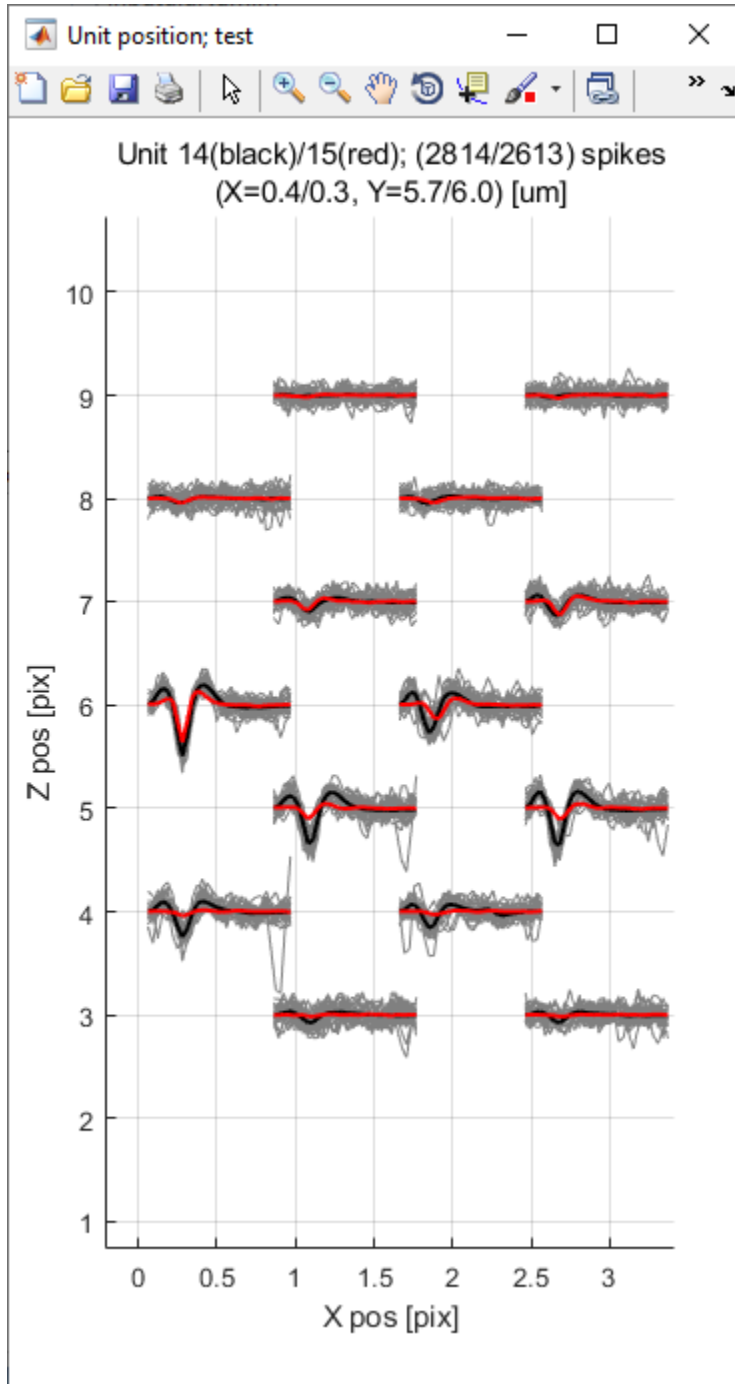
The return map view shows a sampling of interspike intervals (in milliseconds) from the selected cluster, plotted against the previous ISI. That is, if t_k denotes the length of the interval between spike k and spike $k + 1$, then this figure plots t_{k+1} vs. t_k for some subset of spikes in the selected cluster or clusters.

The probe map view (FigMap)



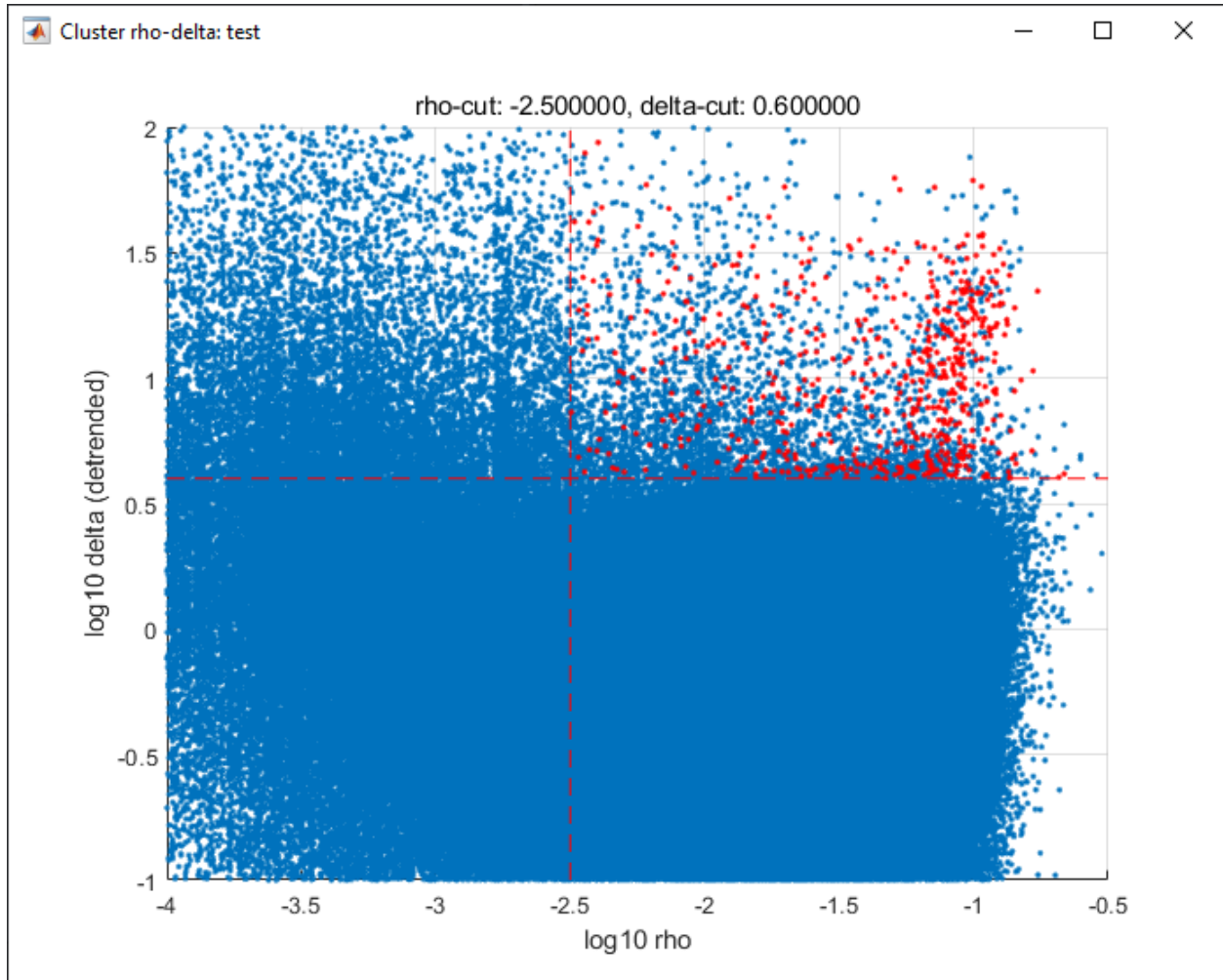
The probe map view plots a color-coded activity map on the probe site layout. The built-in `hot` color map is used to represent the Vpp of the average waveform of the selected cluster, so lighter colors indicate larger Vpp.

The probe position view (FigPos)



The probe position view shows the mean waveforms of the selected cluster or clusters on the probe. Whereas the waveform view shows the mean waveforms of each cluster stacked linearly, the position view shows where these waveforms are on the probe.

The rho-delta view (FigRD)



This figure shows the delta values plotted against the rho values for all spikes. Cluster centers are highlighted in red and the $\log_{10}\text{RhoCut}$ and $\log_{10}\text{DeltaCut}$ thresholds are plotted as dashed vertical and horizontal lines, respectively.

3.1 Common parameters

3.1.1 CARMode

(Formerly `vcCommonRef`)

The meaning of ‘average’ in ‘common average reference’.

One of the following:

- ‘mean’: Computes the mean across sites, excluding sites in *ignoreSites*.
- ‘median’: Computes the median across sites, excluding sites in *ignoreSites*.
- ‘none’: Skips CAR.

Default is ‘mean’.

3.1.2 RDDetrendMode

(Formerly `vcDetrend_postclu`)

Detrending mode to apply to rho-delta values in order to *determine cluster centers*.

One of the following:

- ‘global’: Performs the detrending for all sites together.
- ‘local’: Performs the detrending for each site separately.
- ‘logz’: Use z-scores instead of detrending.
- ‘regress’: Use a non-constant threshold to determine centers for each site.
- ‘none’: Skips detrending.

Default is ‘global’.

3.1.3 autoMergeBy

(Formerly autoMergeCriterion)

Metric to use for *automerging clusters* based on average waveform.

One of the following:

- ‘pearson’: Use the [Pearson correlation coefficient](#) to compute similarity.
- ‘dist’: Define similarity as $1 - \frac{\|tr_i - tr_j\|}{\max(\|tr_i\|, \|tr_j\|)}$, where tr_k is the mean waveform for cluster k .

Default is ‘pearson’.

3.1.4 bitScaling

(Formerly uV_per_bit)

ADC bit scaling factor (Conversion factor for ADC bit values to μV).

Default is 0.30518.

3.1.5 blankPeriod

(Formerly blank_period_ms)

Duration of blanking period (in ms) when the common mean exceeds *blankThresh*.

Default is 5.

3.1.6 blankThresh

(Formerly blank_thresh)

Threshold (in MADs) above which to reject samples exceeding channel median after filtering.

Default is empty.

3.1.7 clusterFeature

(Formerly vcFet)

The feature to extract from your spike waveforms in order to cluster them.

One of the following:

- ‘cov’: Covariance feature.
- ‘energy’: Standard deviation of spike waveform.
- ‘gpca’: *Global PCA*.
- ‘pca’: Projection of spike waveforms onto *principal components*.
- ‘vmin’: *Minimum spike amplitude*.
- ‘vminmax’: *Minimum and maximum spike amplitudes* (2 features/site).
- ‘vpp’: *Peak-to-peak spike amplitudes*.

Default is ‘pca’.

3.1.8 dataType

(Formerly `vcDataType`)

Format of raw recordings.

One of the following:

- 'int16'
- 'uint16'
- 'int32'
- 'uint32'
- 'single'
- 'double'

Default is 'int16'.

3.1.9 dispTimeLimits

(Formerly `tlim`)

Time range (in ms) to display.

Default is [0, 0.2].

3.1.10 distCut

(Formerly `dc_percent`)

Percentile of pairwise distances between spikes on a site to use as a cutoff distance.

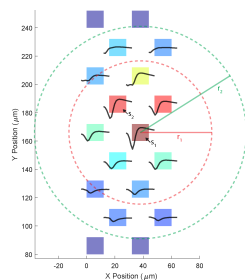
Default is 2.

3.1.11 evtDetectRad

(Formerly `maxDist_site_um`)

Maximum distance (in μm) to search over for *potential duplicates* (± 1 in the figure below). This distance is used to determine the number of sites to extract features if *nSiteDir* is empty.

Default is 50.

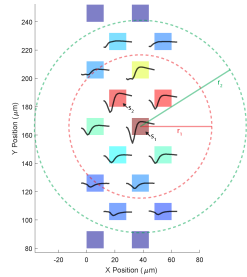


3.1.12 evtGroupRad

(Formerly `maxDist_site_spk_um`)

Maximum distance (in μm) for *extracting spike waveforms* (`r2` in the figure below).

Default is 75.



3.1.13 evtMergeRad

(Formerly `maxDist_site_merge_um`)

Maximum distance (in μm) between sites to consider *merging a pair of units*.

Default is 35.

3.1.14 evtWindow

(Formerly `spkLim_ms`)

Time range (in ms) of filtered spike waveforms, centered at the peak.

Must be an array with 2 elements, the first negative and the second positive. For example, if `evtWindow` is set to `[-0.5, 0.5]`, then 1/2 ms worth of samples are extracted before and after the spiking event.

Default is `[-0.25, 0.75]`.

3.1.15 filtOrder

Bandpass filter order.

Default is 3.

3.1.16 filterType

(Formerly `vcFilter`)

Type of filter to use on raw data.

One of the following:

- ‘ndiff’: Applies a differentiation filter, choosing a kernel depending on the order given in `nDiffOrder`.
- ‘sgdiff’: Applies a [Savitzky-Golay](#) filter depending on the order given in `nDiffOrder`.
- ‘bandpass’
- ‘fir1’

- ‘user’: Convolves your raw samples with a *kernel of your choosing*.
- ‘none’: Skips filtering (not recommended).

Default is ‘ndiff’.

3.1.17 freqLimBP

(Formerly `freqLim`)

Frequency cutoffs for bandpass filter.

Default is [300, 3000].

3.1.18 headerOffset

(Formerly `header_offset`)

Recording file header offset (in bytes).

JRCLUST will skip this many bytes at the beginning of your recording file.

Default is 0.

3.1.19 ignoreChans

(Formerly `viChanZero`)

Channel numbers to ignore manually. Should be an array of integers between 1 and `nChans`. These values will be merged into *ignoreSites*.

Default is empty.

3.1.20 ignoreSites

(Formerly `viSiteZero`)

Site IDs to ignore manually. Should be an array of integers between 1 and `nSites`.

Default is empty.

3.1.21 log10DeltaCut

(Formerly `delta1_cut`)

\log_{10} of delta cutoff (Spikes with delta values below this cutoff will not be considered as cluster centers).

Default is 0.6.

3.1.22 log10RhoCut

(Formerly `rho_cut`)

\log_{10} of rho cutoff (Spikes with rho values below this cutoff will not be considered as cluster centers).

Default is -2.5.

3.1.23 maxUnitSim

(Formerly maxWavCor)

Threshold for merging two units having similar spike waveforms (Units with a similarity score above this value will be merged).

See *autoMergeBy* for how “similarity” is defined.

Default is 0.98.

3.1.24 minClusterSize

(Formerly min_count)

Minimum number of spikes per cluster (Automatically set to the maximum of this value and twice the number of features).

Default is 30.

3.1.25 nChans

Number of channels stored in recording file (Distinct from the number of AP sites).

Default is 384.

3.1.26 nClusterIntervals

(Formerly nTime_clu)

Number of intervals to divide the recording into around a spike.

When clustering, take the $\frac{1}{nClusterIntervals}$ fraction of all spikes around a spiking event to compute distance.

For example, if `nClusterIntervals = 1`, all spikes will be used; if `nClusterIntervals = 2`, JRCLUST will take the half of all spikes which are closest in time to compute distances. Increasing this value will take fewer and fewer spikes to compare at the risk of oversplitting clusters (you might want to do this if you observe fast drift in your recording). However, automated merging based on the *waveform correlation* can merge most of the units initially split by drift.

Default is 4.

3.1.27 nPCsPerSite

(Formerly nPcPerChan)

Number of principal components to compute per site.

Default is 1.

3.1.28 nSiteDir

(Formerly maxSite)

Number of neighboring sites to group in either direction.

The total number of sites per spike group (`nSitesEvt`) is $1 + 2 * \text{“nSiteDir”}$. In other words, a spike group includes the site on which the spike occurs, along with `nSiteDir` sites in the horizontal direction and `nSiteDir` in the vertical direction.

If empty, the number of sites per spike group is determined from *evtGroupRad*.

Warning: This parameter may be deprecated in an upcoming release in favor of `evtGroupRad`.

Default is empty.

3.1.29 nSitesExcl

(Formerly `nSites_ref`)

Number of sites to exclude from the spike waveform group for feature extraction.

Default is empty.

3.1.30 nSpikesFigProj

(Formerly `nShow_proj`)

Maximum number of spikes per cluster to display in the feature projection view.

Default is 500.

3.1.31 nSpikesFigWav

(Formerly `nSpk_show`)

Maximum number of spikes per cluster to display generally.

Default is 30.

3.1.32 outputDir

Directory in which to place output files (Will output to the same directory as this file if empty).

Default is an empty string.

3.1.33 probePad

(Formerly `vrSiteHW`)

Recording contact pad size (in μm) (Height x width).

Default is empty.

3.1.34 psthTimeLimits

(Formerly `tlim_psth`)

Time range (in s) over which to display PSTH.

Default is empty.

3.1.35 qqFactor

Spike detection threshold.

Multiplier of the *estimate* $\sigma_{\text{noise}}^{(i)}$ of standard deviation of noise distribution on each site to compute the threshold for that site. In other words,

$$\text{Thr}_i := \text{qqFactor} \cdot \sigma_{\text{noise}}^{(i)}$$

is the spike detection threshold for site i .

Default is 5.

3.1.36 rawRecordings

Path or paths to raw recordings to sort.

Default is empty.

3.1.37 recordingFormat

Format of raw recordings.

One of the following:

- ‘SpikeGLX’: A flat [binary file](<https://github.com/billkarsh/SpikeGLX/blob/master/Markdown/UserManual.md#output-file-format>) with no header, stored in channels x samples order.
- ‘Intan’: Traditional [Intan file format](http://www.intantech.com/files/Intan_RHD2000_data_file_formats.pdf).

Default is ‘SpikeGLX’.

3.1.38 refracInt

(Formerly `spkRefrac_ms`)

Spike refractory period (in ms).

Default is 0.25.

3.1.39 sampleRate

(Formerly `sRateHz`)

Sampling rate (in Hz) of raw recording.

Default is 30000.

3.1.40 shankMap

(Formerly `viShank_site`)

Shank ID of each site.

Default is empty.

3.1.41 siteLoc

(Formerly `mrSiteXY`)

Site locations (in μm) (x values in the first column, y values in the second column).

Default is empty.

3.1.42 siteMap

(Formerly `viSite2Chan`)

Map of channel index to site ID (The mapping `siteMap(i) = j` corresponds to the statement ‘site i is stored as channel j in the recording’).

Default is empty.

3.1.43 trialFile

(Formerly `vcFile_trial`)

Path to file containing trial data (Can be `.mat` or `.csv`, must contain timestamps of trials in units of s).

Default is an empty string.

3.2 Advanced parameters

3.2.1 auxChan

(Formerly `iChan_aux`)

Auxiliary channel index.

Default is empty.

3.2.2 auxFile

(Formerly `vcFile_aux`)

Path to file containing auxiliary channel.

Default is an empty string.

3.2.3 auxLabel

(Formerly `vcLabel_aux`)

Label for auxiliary channel data.

Default is 'Aux channel'.

3.2.4 auxSampleRate

(Formerly `sRateHz_aux`)

Sample rate for auxiliary file.

Default is empty.

3.2.5 auxScale

(Formerly `vrScale_aux`)

Scale factor for aux data.

Default is 1.

3.2.6 batchMode

Suppress message boxes in favor of console messages.

Default is true.

3.2.7 colorMap

(Formerly `mrColor_proj`)

RGB color map for background, primary selected, and secondary selected spikes (The first three values are the R values, the next three are the G values, and the last three are the B values.).

Default is [0.83203, 0, 0.9375, 0.85547, 0.50781, 0.46484, 0.91797, 0.76563, 0.085938].

3.2.8 corrRange

(Formerly `corrLim`)

Correlation score range to distinguish by color map.

Default is [0.9, 1].

3.2.9 detectBipolar

(Formerly `fDetectBipolar`)

Detect positive as well as negative peaks.

Default is false.

3.2.10 dispFeature

(Formerly `vcFet_show`)

Feature to display in the feature projection plot.

One of the following:

- 'cov'
- 'pca'
- 'ppca'
- 'vpp'
- 'template' (only supported for template-based clusterings, e.g., Kilosort)

Default is 'vpp'.

3.2.11 dispFilter

(Formerly `vcFilter_show`)

Filter to apply in traces plot.

One of the following:

- 'ndiff'
- 'sgdiff'
- 'bandpass'
- 'fir1'
- 'user'
- 'none'

Default is 'none'.

3.2.12 driftMerge

(Formerly `fDrift_merge`)

Compute multiple waveforms at three drift locations based on the spike position if true.

Default is true.

3.2.13 evtManualThresh

(Formerly `spkThresh_uV`)

Manually-set spike detection threshold (in μV).

Default is empty.

3.2.14 evtWindowMergeFactor

(Formerly `spkLim_factor_merge`)

Ratio of samples to take when computing correlation.

Default is 1.

3.2.15 evtWindowRaw

(Formerly `spkLim_raw_ms`)

Time range (in ms) of raw spike waveforms, centered at the peak.

Must be an array with 2 elements, the first negative and the second positive. For example, if `evtWindowRaw` is set to `[-1, 1]`, then 1 ms worth of samples are extracted before and after the spiking event.

Default is `[-0.5, 1.5]`.

3.2.16 extractAfterDetect

Extract features only after detecting all spikes across all files if true. Otherwise, features will be computed on a per-chunk, per-file basis, which may not be what you want.

This is effectively set to true if you specify `clusterFeature = 'gpca'`.

Default is false.

3.2.17 fftThresh

(Formerly `fft_thresh`)

Threshold (in MADs of power-frequency product) above which to remove frequency outliers when *denoising*. Frequencies with power-frequency product above this threshold will be zeroed out as noise.

Setting to 0 disables this notch filtering. If you choose to enable, the recommended value is 10.

Default is 0.

3.2.18 figList

List of tags of figures to display in feature view.

One of the following:

- 'FigCorr'
- 'FigHist'
- 'FigISI'
- 'FigMap'
- 'FigPos'
- 'FigProj'
- 'FigRD'
- 'FigSim'

- 'FigTime'
- 'FigWav'

Default is ["FigCorr", "FigHist", "FigISI", "FigMap", "FigPos", "FigProj", "FigRD", "FigSim", "FigTime", "FigWav"].

3.2.19 frFilterShape

(Formerly `filter_shape_rate`)

Kernel shape for temporal averaging (Used in estimation of the firing rate of a given unit).

One of the following:

- 'triangle'
- 'rectangle'

Default is 'triangle'.

3.2.20 frPeriod

(Formerly `filter_sec_rate`)

Time period (in s) over which to determine firing rate (Used in estimation of the firing rate of a given unit).

Default is 2.

3.2.21 frSampleRate

(Formerly `sRateHz_rate`)

Resampling rate (in Hz) for estimating the firing rate (Used in estimation of the firing rate of a given unit).

Default is 1000.

3.2.22 freqLimNotch

Frequency ranges to exclude for notch filter.

Default is empty.

3.2.23 freqLimStop

Frequency range to exclude for band-stop filter.

Default is empty.

3.2.24 gainBoost

(Formerly `gain_boost`)

Scale factor to boost gain in raw recording (Used in filtering operation).

Default is 1.

3.2.25 gpuLoadFactor

GPU memory usage factor (Use $1/\text{gpuLoadFactor}$ amount of GPU memory).

Default is 5.

3.2.26 groupShank

(Formerly `fGroup_shank`)

Group all sites on the same shank if true.

Default is true.

3.2.27 gtFile

(Formerly `vcFile_gt`)

Path to file containing ground-truth data.

Default is an empty string.

3.2.28 interpPC

(Formerly `fInterp_fet`)

Interpolate 1st principal vector to maximize projection of spikes if true.

Default is true.

3.2.29 lfpSampleRate

(Formerly `sRateHz_lfp`)

Sampling rate for LFP channels.

Default is 2500.

3.2.30 loadTimeLimits

(Formerly `tlim_load`)

Time range (in s) of samples to load at once (All samples are loaded if empty).

Default is empty.

3.2.31 maxAmp

Amplitude scale (in μV).

Default is 250.

3.2.32 maxBytesLoad

(Formerly MAX_BYTES_LOAD)

Maximum number of bytes to load into memory.

Default is empty.

3.2.33 maxClustersSite

(Formerly maxCluPerSite)

Maximum number of cluster centers computed per site (Used if *RDDetrendMode* is 'local').

Default is 20.

3.2.34 maxSecLoad

(Formerly MAX_LOAD_SEC)

Maximum sample duration (in s) to load into memory (Overrides maxBytesLoad if nonempty).

Default is empty.

3.2.35 meanInterpFactor

(Formerly nInterp_merge)

Interpolation factor for mean unit waveforms (Set to 1 to disable).

Default is 1.

3.2.36 minNeighborsDetect

(Formerly nneigh_min_detect)

Minimum number of sample neighbors exceeding threshold for a sample to be considered a peak.

For example, consider a potential peak occurring at sample t_i . If minNeighborsDetect is set to 1, then **either** sample t_{i-1} or t_{i+1} must also exceed the detection threshold. If minNeighborsDetect is set to 2, then **both** sample t_{i-1} and t_{i+1} must also exceed the detection threshold. If minNeighborsDetect is set to 0, then samples t_{i-1} and t_{i+1} are not considered.

Must be one of 0, 1, or 2.

Default is 0.

3.2.37 minSitesWeightFeatures

(Formerly min_sites_mask)

Minimum number of sites to have if using weightFeatures (Ignored if weightFeatures is false).

Default is 5.

3.2.38 nClustersShowAux

(Formerly nClu_show_aux)

Number of clusters to show in the aux vs. firing rate correlation.

Default is 10.

3.2.39 nDiffOrder

(Formerly nDiff_filt)

Order for differentiator filter (Used if and only if *filterType* is 'sgdiff' or 'ndiff').

Default is 2.

3.2.40 nLoadsMaxPreview

(Formerly nLoads_max_preview)

Number of time segments to load in preview.

Default is 30.

3.2.41 nPassesMerge

(Formerly nRepeat_merge)

Number of times to repeat automatic waveform-based merging.

Default is 10.

3.2.42 nPeaksFeatures

(Formerly nFet_use)

Number of potential peaks to use when computing features.

Default is 2.

3.2.43 nSamplesPad

(Formerly nPad_filt)

Number of samples to overlap between chunks in large files.

Default is 100.

3.2.44 nSecsLoadPreview

(Formerly sec_per_load_preview)

Number of seconds to load in preview.

Default is 1.

3.2.45 nSegmentsTraces

(Formerly `nTime_traces`)

Number of time segments to display in traces view (A value of 1 shows one continuous time segment).

Default is 1.

3.2.46 nSitesFigProj

Number of sites to show in feature projection view.

Default is 5.

3.2.47 nSkip

(Formerly `nSkip_show`)

Show every `nSkip` samples when plotting traces.

Default is 1.

3.2.48 nSpikesFigISI

Maximum number of spikes to show in ISI view.

Default is 200.

3.2.49 nThreadsGPU

(Formerly `nThreads`)

Number of GPU threads to use for clustering.

Default is 128.

3.2.50 outlierThresh

(Formerly `thresh_mad_clu`)

Threshold (in MADs) to remove outlier spikes for each cluster.

Default is 7.5.

3.2.51 pcPair

Pair of PCs to display.

Default is [1, 2].

3.2.52 projTimeLimits

(Formerly `tLimFigProj`)

Time range (in s) to display in feature projection view.

Default is empty.

3.2.53 psthTimeBin

(Formerly `tbin_psth`)

Time bin (in s) for PSTH view.

Default is 0.01.

3.2.54 psthXTick

(Formerly `xtick_psth`)

PSTH time tick mark spacing.

Default is 0.2.

3.2.55 ramToGPUFactor

(Formerly `nLoads_gpu`)

Ratio of RAM to GPU memory.

Default is 8.

3.2.56 randomSeed

Seed for the random number generator.

Default is 0.

3.2.57 realignTraces

Realign spike traces after subtracting local CAR (Realign if 1, perform subpixel interpolation if 2).

Default is 0.

3.2.58 showRaw

(Formerly `fWav_raw_show`)

Show raw traces in waveform view if true.

Default is false.

3.2.59 showSpikeCount

(Formerly `fText`)

Show spike count per unit in waveform plot.

Default is true.

3.2.60 siteCorrThresh

(Formerly `thresh_corr_bad_site`)

Threshold to reject bad sites based on maximum correlation with neighboring sites (Set to 0 to disable).

Default is 0.

3.2.61 spikeThreshMax

(Formerly `spkThresh_max_uV`)

Maximum absolute amplitude (in μV) permitted for spikes.

Default is empty.

3.2.62 tallSkinny

(Formerly `fTranspose_bin`)

Recording will be interpreted as `nChannels` x `nSamples` if true.

Default is true.

3.2.63 threshFile

(Formerly `vcFile_thresh`)

Path to `.mat` file storing the spike detection threshold (Created by preview GUI).

Default is an empty string.

3.2.64 umPerPix

(Formerly `um_per_pix`)

Vertical site center-to-center spacing.

Default is 20.

3.2.65 useElliptic

(Formerly `fEllip`)

Use elliptic (bandpass) filter if true (Uses Butterworth filter if false).

Default is true.

3.2.66 useGPU

(Formerly fGpu)

Use GPU where appropriate.

Default is true.

3.2.67 useGlobalDistCut

(Formerly fDc_global)

Use a global distance cutoff for all sites if true.

Default is false.

3.2.68 useParfor

(Formerly fParfor)

Use parfor where appropriate.

Default is true.

3.2.69 userFiltKernel

(Formerly vnFilter_user)

User-specified filter kernel (Ignored unless *filterType* is 'user').

Your filtered samples will be the output of a convolution of your raw samples with this kernel. You must specify this if and only if your *filterType* is 'user'.

Default is empty.

3.2.70 verbose

(Formerly fVerbose)

Be chatty when processing.

Default is true.

3.2.71 weightFeatures

(Formerly fSpatialMask_clu)

Weight display features by distance from site if true.

Default is false.

4.1 The JRC handle

This class encapsulates a “session”. A session can consist of previewing, detecting, sorting, curating, and various other operations around a single data set.

4.2 Package `jrclust`

4.3 The Clustering interface

This section is mainly about extending the JRCLUST framework to handle different clustering algorithms or the results of other spike sorting packages. The first thing to do is to design a class that implements the `Clustering` interface.

The `Clustering` interface describes a clustering of spikes. It can be found in the `jrclust.interfaces.Clustering` class (i.e., in `+jrclust/@Clustering/Clustering.m`).

4.3.1 Clustering properties

`hCfg`

A `Config` object.

4.4 DensityPeakClustering

Coming soon.