User-Instructions

Please take a moment to read the Kilosort 4 paper, which is the primary sorting algorithm used. Please also read Spike Interface Paper, which is the open source library the pipeline is built on.

These instructions assume basic familiarity with VS Code, jupyter notebooks, conda environments, and command line interface. If you're unfamiliar, try it anyways! :)

1. Start ec2 instance
    1. See Instructions
2. SSH into AWS machine
    1. See Instructions
    2. A Laptop will be configured to have easy access to AWS Machine
    3. student@z-lab-ec2
3. Open "codespace"
4. Open Notebooks -> z-sort notebook
5. data lives in the data directory
    1. data is necessarily formatted like this:

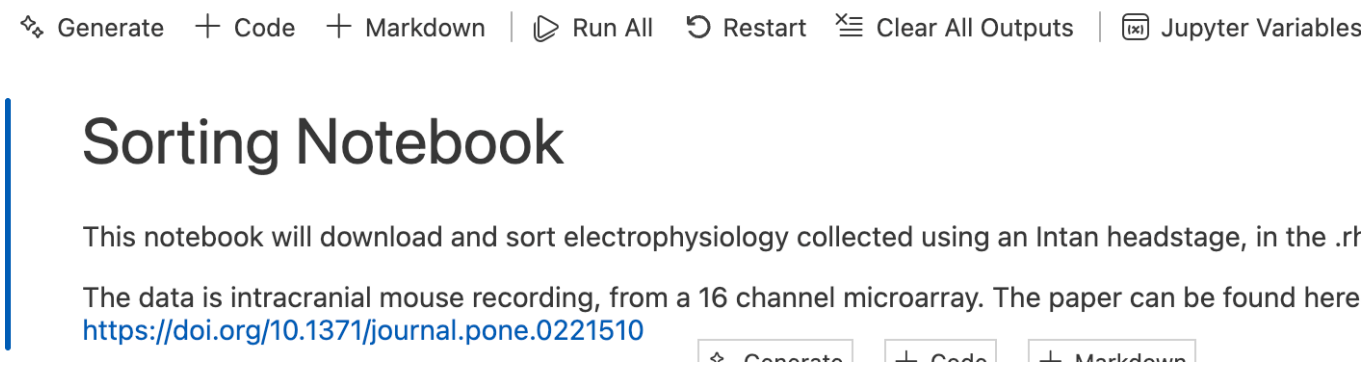
```
data                                # Contains patient data
    Patient-1                       # <C> Variable patient ID
        session-1                   # <C> Variable session ID
            raw                     # Always called raw
                intan_file.rdh      # Raw recording file
            sorted                  # Contents are generated by sorting
                figs                # Contains png summary
                sorter_folder       # Contains the sorted spikes
                analyzer_folder     # <P> Nessecary intermediate folder
                curated_analyzer.zarr  # Final product to export
        session-2                   # Other Patient-1 Sessions
            ...
    Patient-2                       # Next Patient
        ...

sorting_script                      # Sorting Code
    custom_probes                   # Contains nessecary probe geometry
    docs                            # Contains helpful docs
    notebooks                       # Contains sorting notebookas
        z-sort_notebook.ipynb     ---> # This is the main notebook
    src                             # Source code (edit with caution)

---
```

```
<C> - You will need to copy the name of this directory into the notebook
<P> - You will need to paste the relative path and paste into the notebook,
        which is done by opening the file navigation bar on the left of VS
        code, right clicking, and "copy relative path". This path is needed
        to launch the
```

Run jupyter notebook cells in order

-1. If you just did some sorting, make sure to end the active terminal by pressing control+z

0. In between sessions, you *MUST* restart the notebook, otherwise you will get phantom data 👻



1. Activate conda environment

```
# Getting Set Up

1. Open a terminal. Make sure "Sorter" environment is active by running these commands in the terminal

```
conda deactivate
conda activate sorter
```
```

```
(sorter) marco@ip-172-31-78-229:~/codespace/sorting_script$ conda deactivate
conda activate sorter
```

2. import lab script

```
# 2. Import scripts
from sorting_scripts import zsort
```

✓ 0.6s

---

3. Set Patient and Session Paths

```
# 3. Set Patient and Session Paths
patient = "Intan_RDH_2000"
session = "Session1"

path_dict = zsort.set_paths(patient, session)
path_dict
```

✓ 0.0s

```
1
Found Intan file: /data/Intan_RDH_2000/Session1/raw/Intan RHD file1.rhd

{'patient': 'Intan_RDH_2000',
 'session': 'Session1',
 'repo_root': PosixPath('/home/marco/codespace/sorting_script'),
 'data_root': PosixPath('/data'),
 'session_location': PosixPath('/data/Intan_RDH_2000/Session1'),
 'sorted_data': PosixPath('/data/Intan_RDH_2000/Session1/Intan_RDH_2000-Session1-sorted')
 'sorter_output_folder': PosixPath('/data/Intan_RDH_2000/Session1/Intan_RDH_2000-Session1
 'analyzer_folder': PosixPath('/data/Intan_RDH_2000/Session1/Intan_RDH_2000-Session1-sort
 'intan_file': PosixPath('/data/Intan_RDH_2000/Session1/raw/Intan RHD file1.rhd')}
```

---

4. Load File

```
# 4. Load File

import spikeinterface.full as si

recording_path = path_dict["intan_file"]

rec = si.read_intan(recording_path, stream_id = "0")
```

[3] ✓ 0.2s

5. Set Probe

```
# 5. Set Probe
probe = "neuronexus-A16x1_2mm_50_177_A16.json"
rec = zsort.set_probe(rec, path_dict, probe)
```

[4] ✓ 0.1s

```
no probe, attaching one
```

6. Sort

```
# 6. Sort

sort = zsort.sort(rec, path_dict)
```

[5] ✓ 1m 22.6s

```
[zsort] recording dtype is unsigned (uint16); converting to signed

write_binary_recording (no parallelization): 100% ████████████████████

kilosort.run_kilosort:
kilosort.run_kilosort: Computing preprocessing variables.
kilosort.run_kilosort: ----------------------------------------
kilosort.run_kilosort: N samples: 24000480
kilosort.run_kilosort: N seconds: 1200.024
kilosort.run_kilosort: N batches: 401
kilosort.run_kilosort: Preprocessing filters computed in 0.45s; total 0.45s
kilosort.run_kilosort:
kilosort.run_kilosort: Resource usage after preprocessing
kilosort.run_kilosort: ************************************************************
kilosort.run_kilosort: CPU usage:     7.10 %
kilosort.run_kilosort: Mem used:      5.70 %     |       3.55 GB
kilosort.run_kilosort: Mem avail:    58.55 / 62.10 GB
kilosort.run_kilosort: ------------------------------------------------------------
kilosort.run_kilosort: GPU usage:    `conda install pynvml` for GPU usage
kilosort.run_kilosort: GPU memory:    1.86 %     |      0.27   /    14.58 GB
kilosort.run_kilosort: Allocated:     0.06 %     |      0.01   /    14.58 GB
kilosort.run_kilosort: Max alloc:     0.44 %     |      0.06   /    14.58 GB
kilosort.run_kilosort: ************************************************************
kilosort.run_kilosort:
kilosort.run_kilosort: Computing drift correction.
```

7. Create analyzer

```
# 7. Analyze
zsort.analyze(rec, sort, path_dict)
```

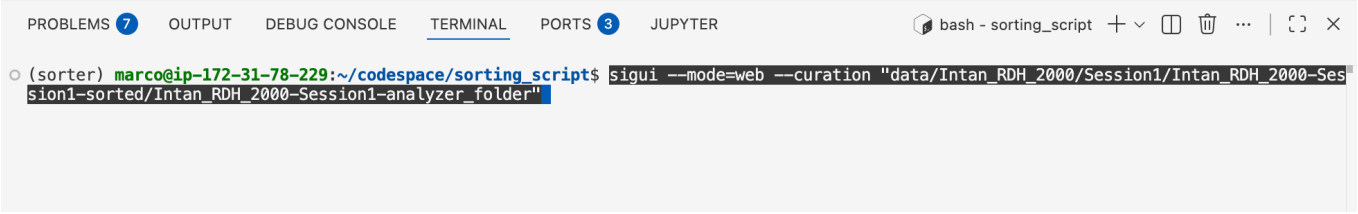
[6] ✓ 29.5s

```
No valid analyzer found, creating a new one
Reason: This folder does not exists /data/Intan_RDH_2000/Session1/Intan
[zsort] recording dtype is unsigned (uint16); converting to signed

estimate_sparsity (workers: 16 processes): 100%

/home/marco/miniconda3/envs/sorter/lib/python3.11/site-packages/spikein
  warnings.warn("The registered recording will not be persistent on disk

compute_waveforms (workers: 16 processes): 100%

noise_level (workers: 16 processes): 100%

Fitting PCA: 100%  32/32 [00:

Projecting waveforms: 100%  3

Compute : spike_amplitudes (workers: 16 processes): 100%

SortingAnalyzer: 16 channels - 32 units - 1 segments - binary_folder - s
Loaded 11 extensions: random_spikes, waveforms, templates, noise_levels
```

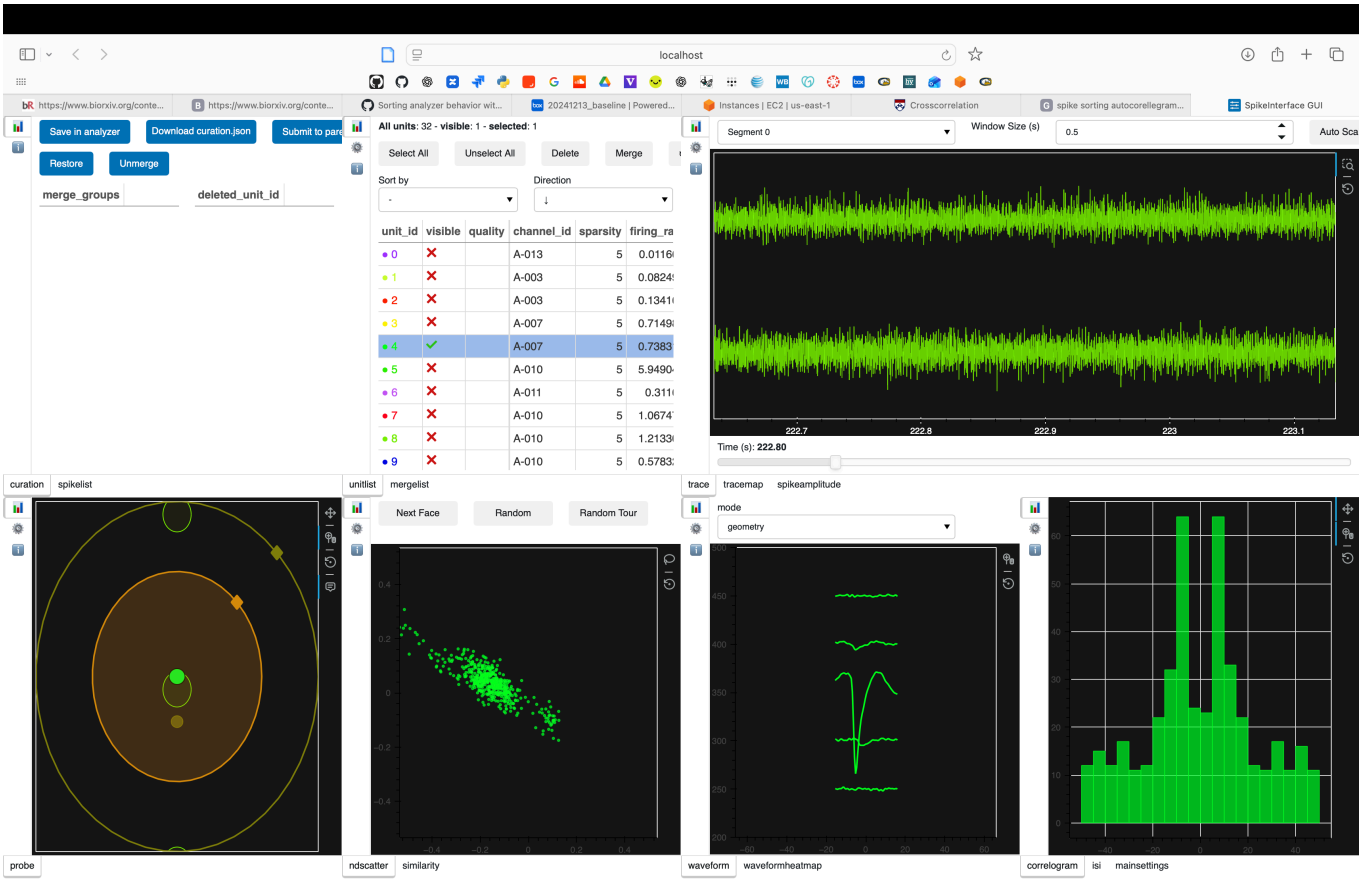
8. launch curator from terminal

Copy and paste this into the terminal

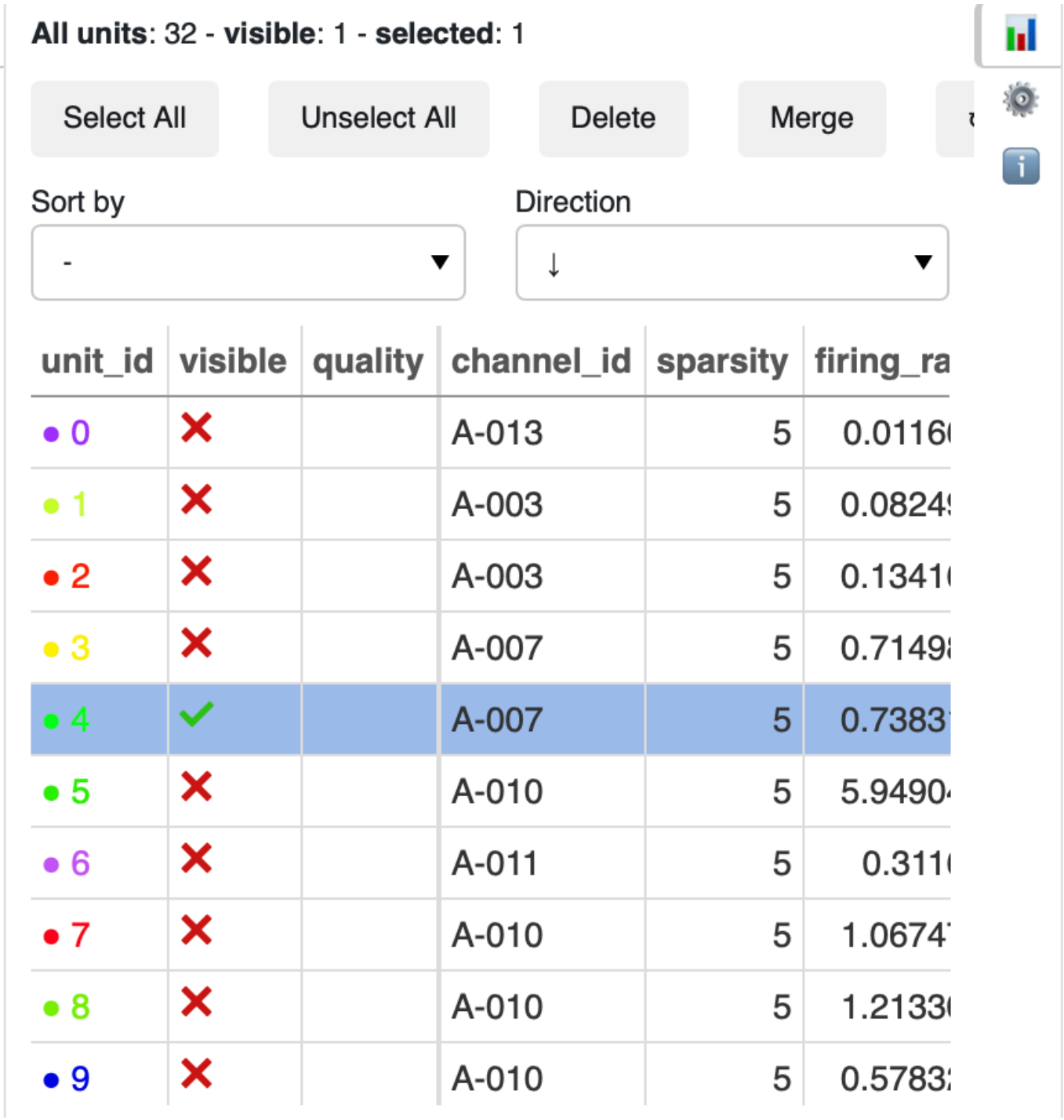
```
sigui --mode=web --curation "data/Intan_RDH_2000/Session1/Intan_RDH_2000-Session1-sorted/Intan_RDH_2000-Session1-analyzer_folder"
```



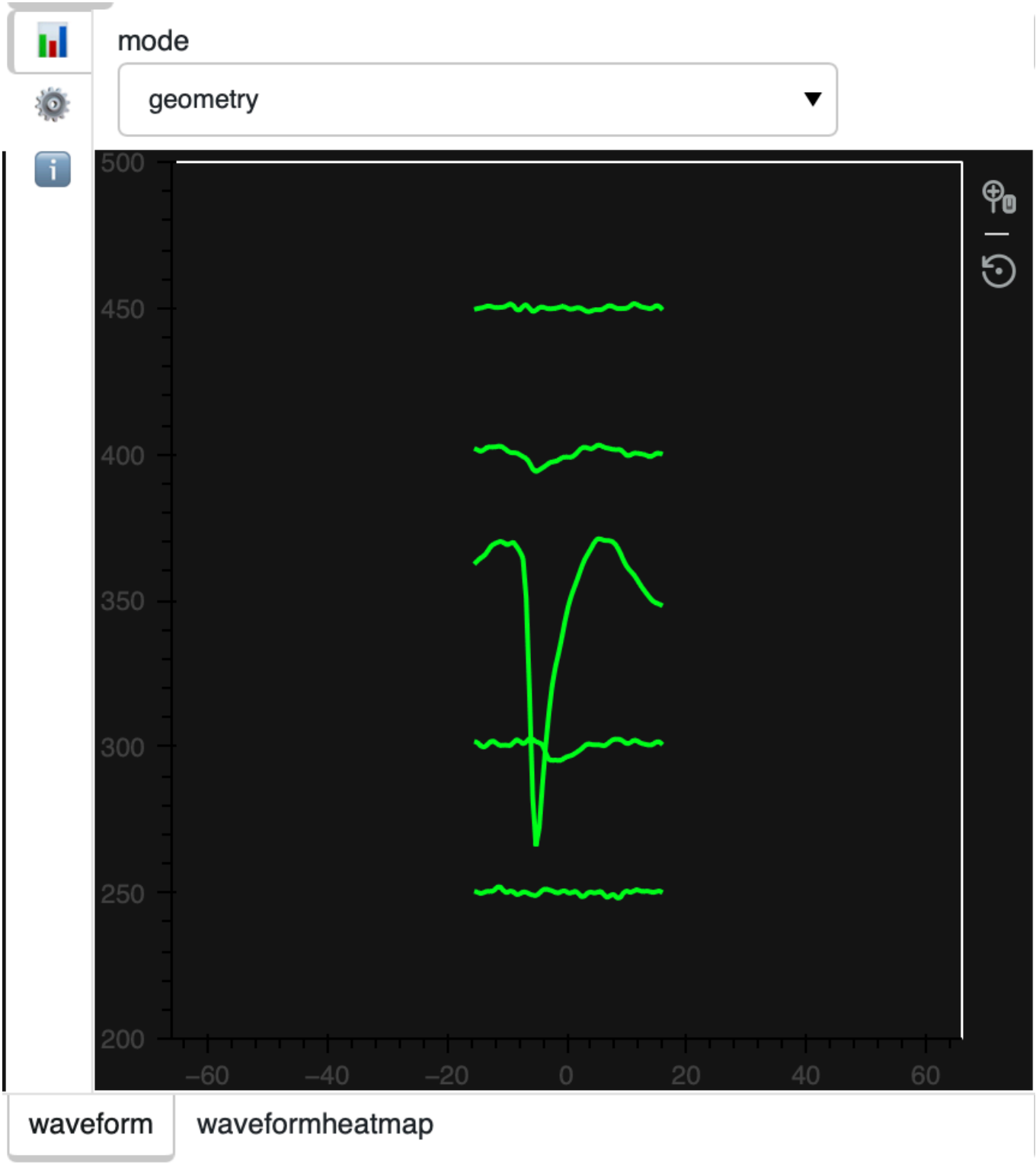
9. curate in the window that pops up in your browser



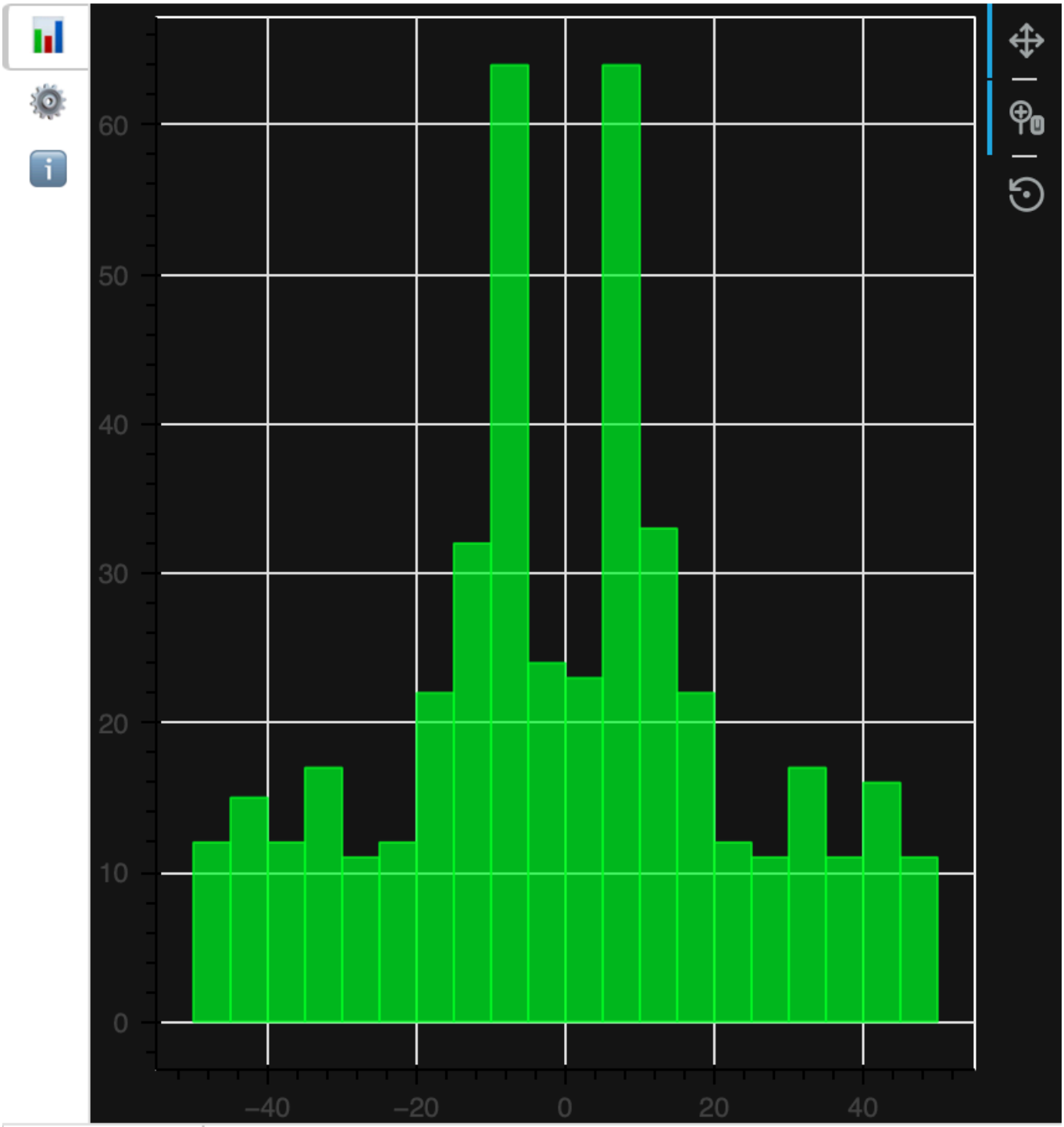
Should popup in browser



These are the units detected by the sorting algorithm

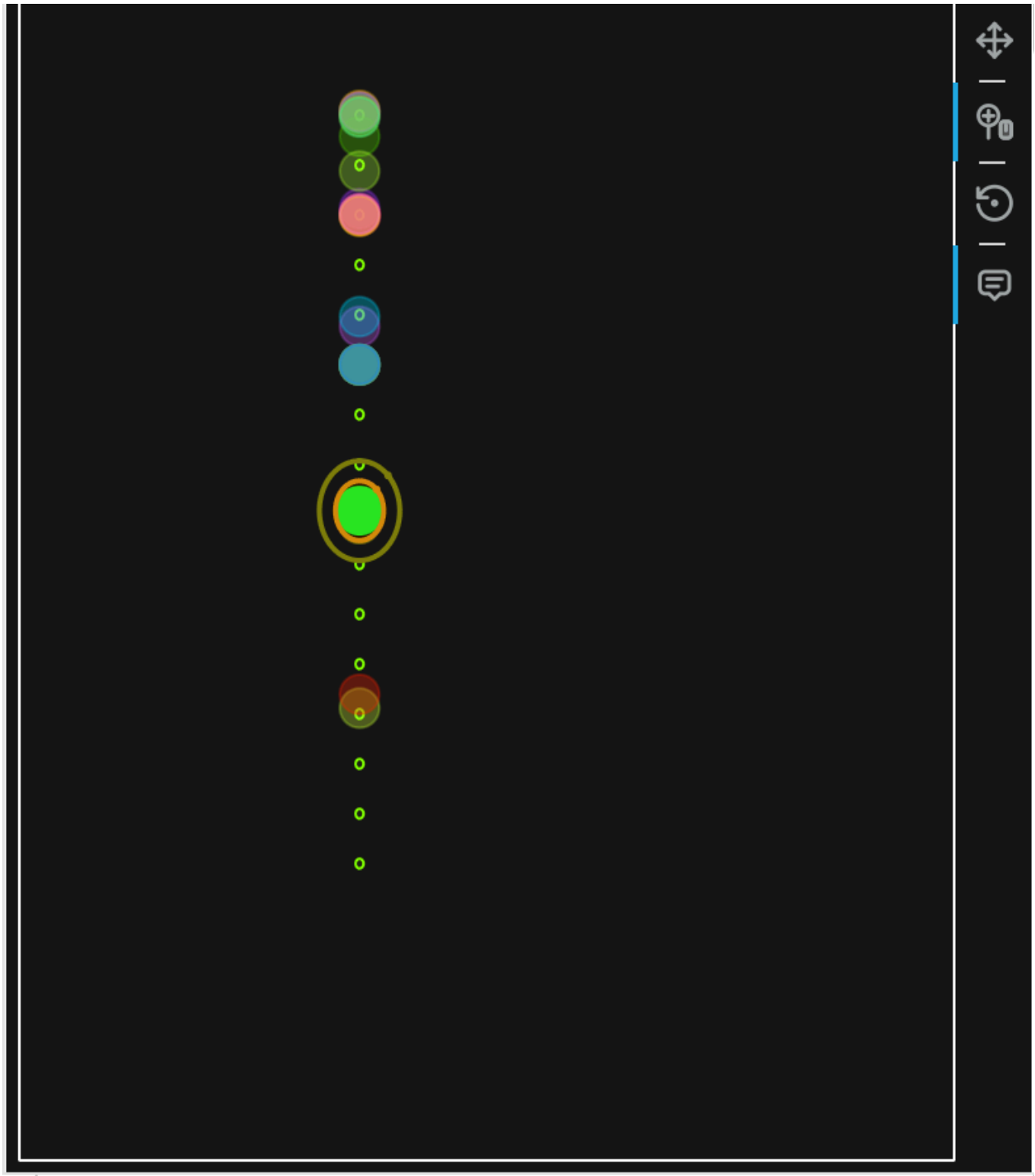


This is the trace of the selected spike, with one trace per nearby electrode



correlogram isi mainsettings

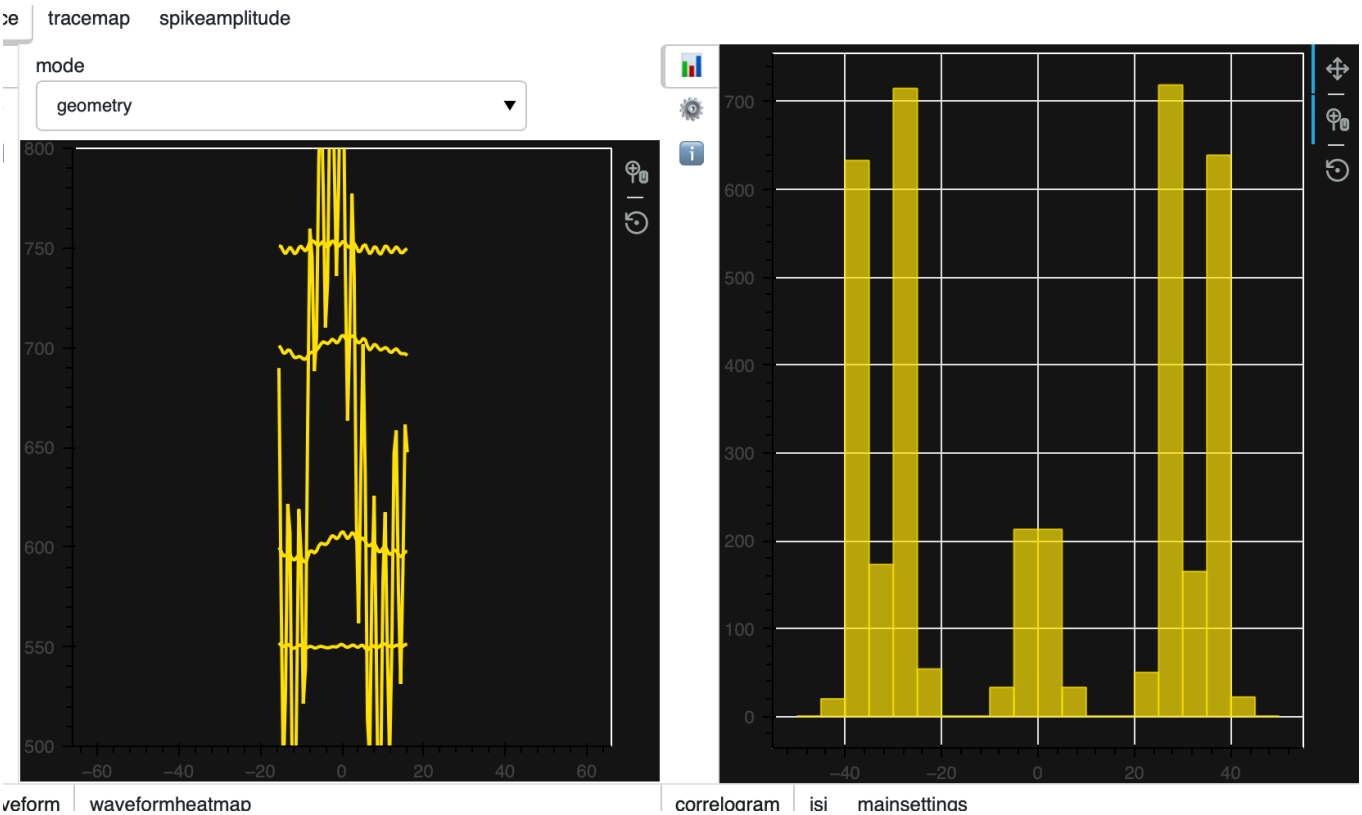
This is the auto-correlogram, which shows a histogram of the intervals between detected spikes. Because of a neurons refractory period, a single unit should show a significant dip around 0ms.



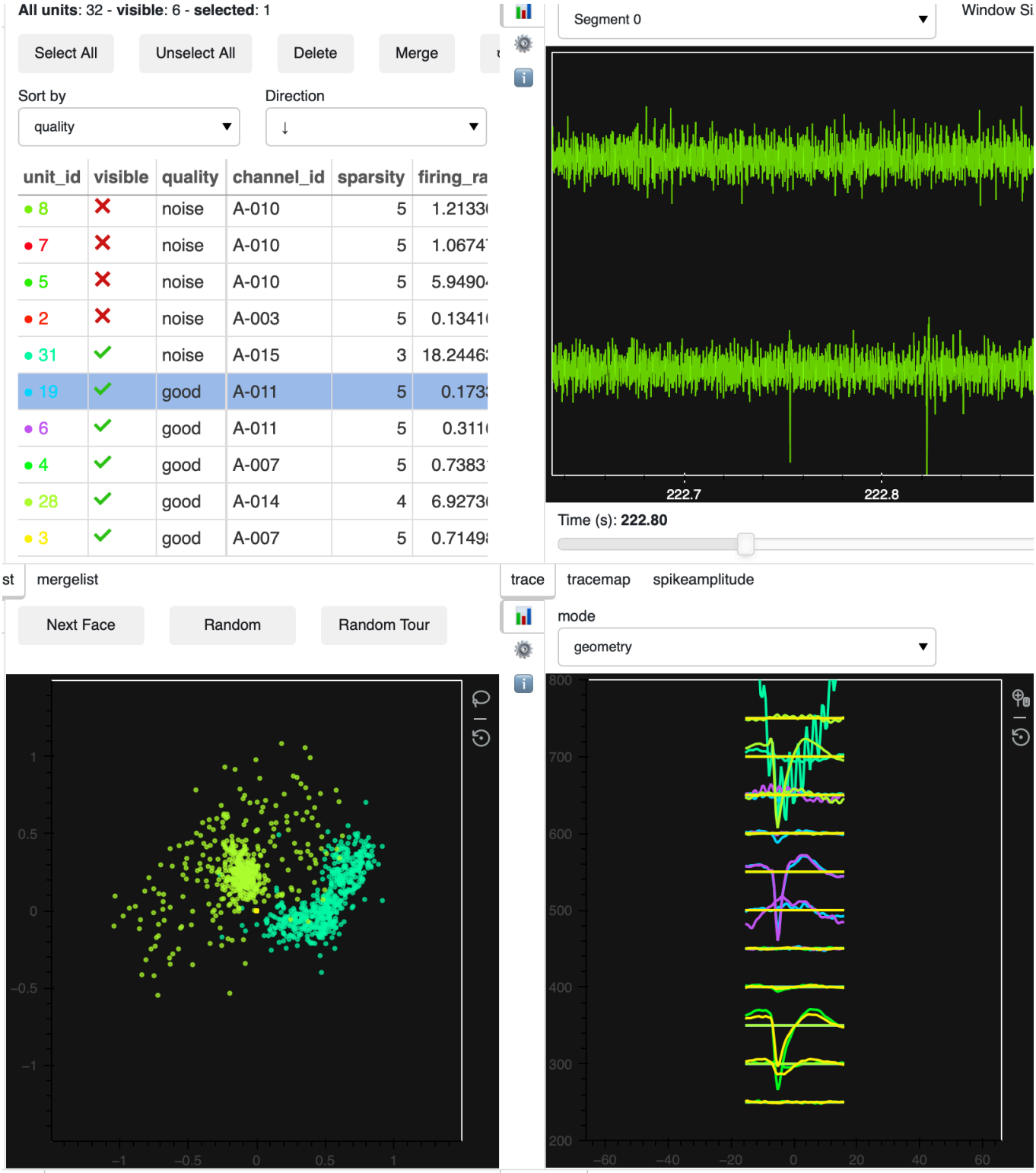
This shows the location of each electrode on the probe in 2D space. The small dots are the electrodes, the larger colored dots are putative spikes.

unit_id	visible	quality	channel_id	sparsity	firing_ra
0	✗		A-013	5	0.0116
1	✗		A-003	5	0.0824
2	✗		A-003	5	0.1341
3	✗		A-007	5	0.7149
4	✓		A-007	5	0.7383
5	✗	good	A-010	5	5.9490
6	✗	noise	A-011	5	0.311
7	✗	MUA	A-010	5	1.0674
8	✗		A-010	5	1.2133
9	✗		A-010	5	0.5783

Use the spike table to select spike quality, either good, noise, or MUA, which is multi unit activity.

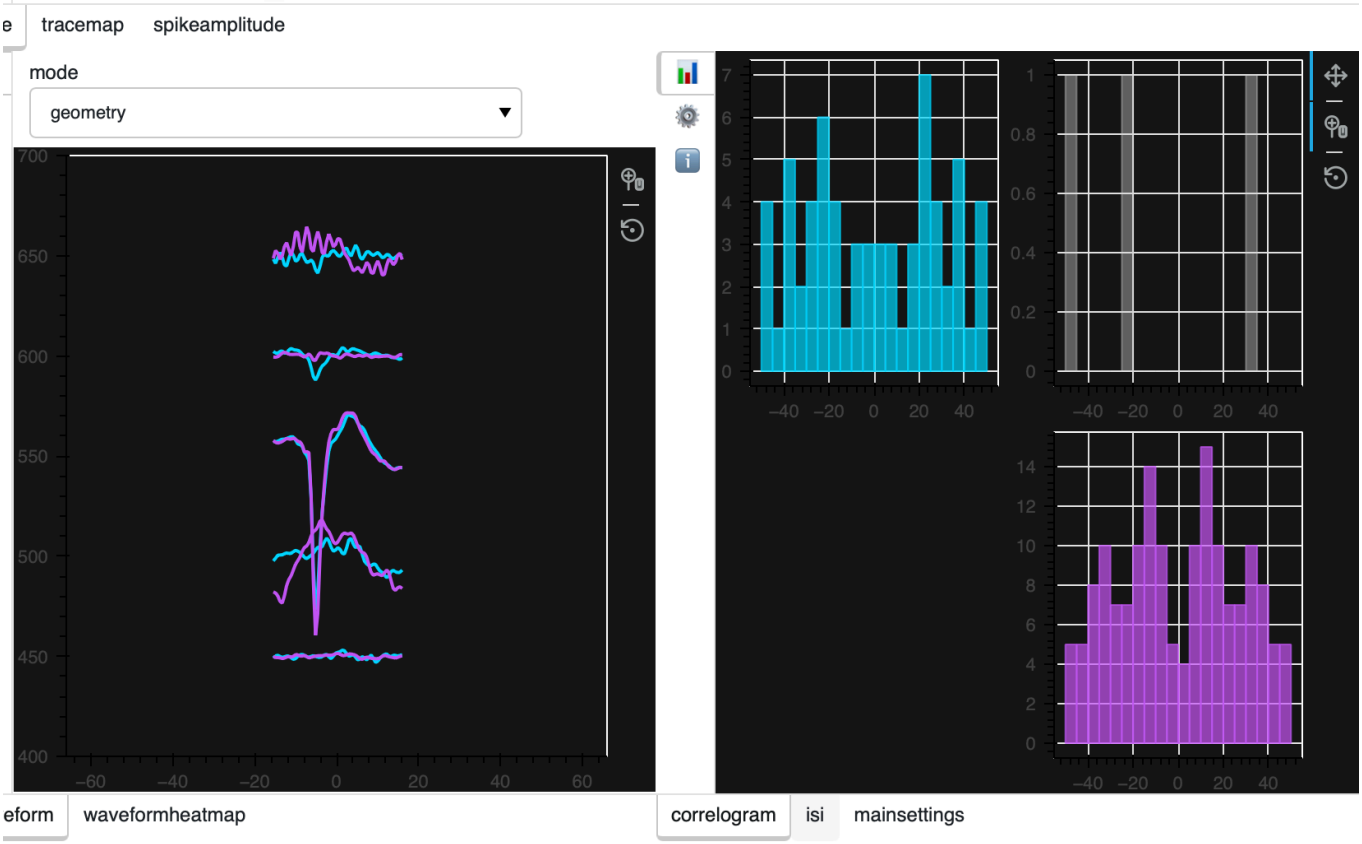


This trace is obvious noise.

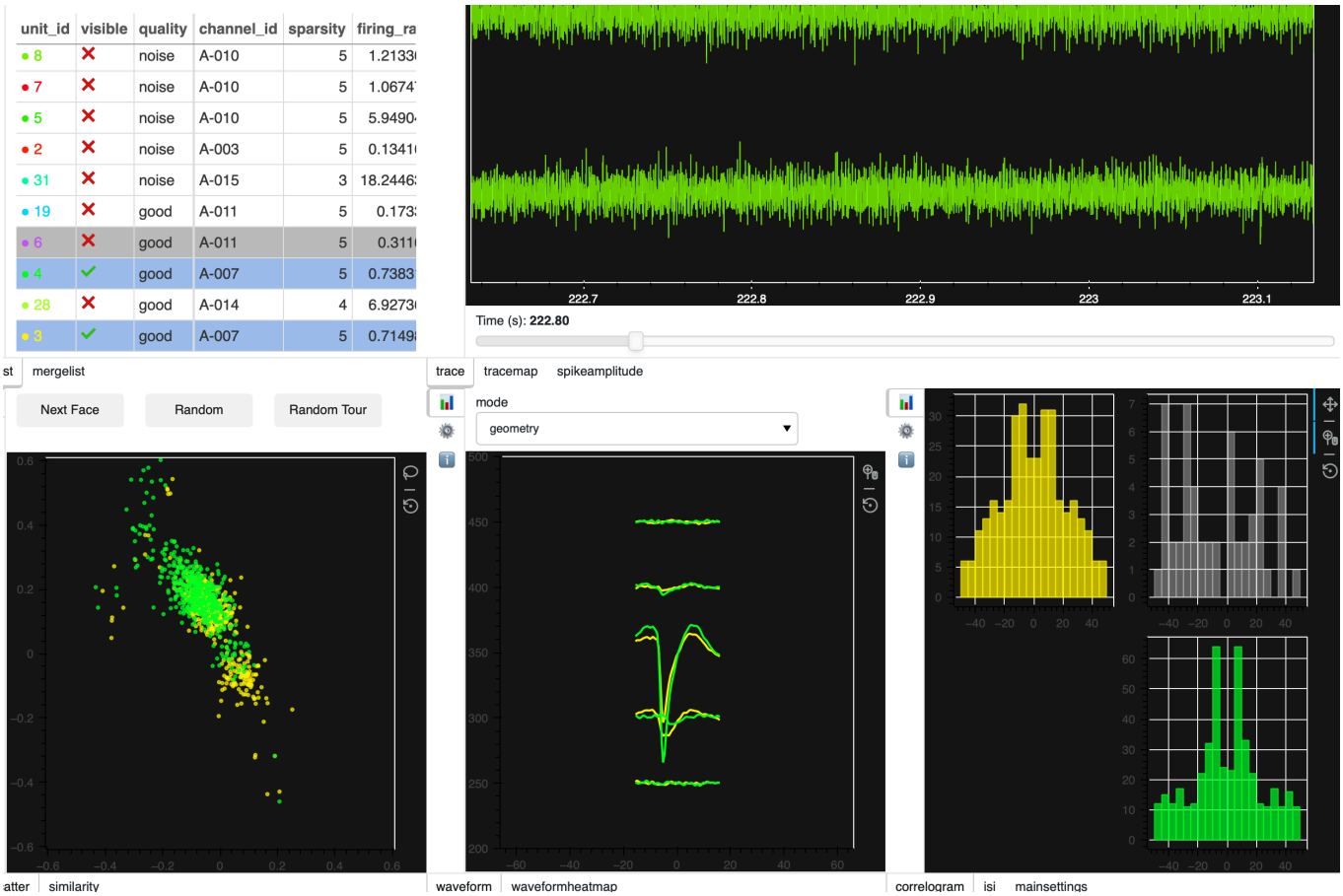


Once all of the spike qualities have been input, you can sort the table so the "good" units are near each other, to seem them all super imposed. If the spikes line up, select those two using "control + click", so see them on top of each other.

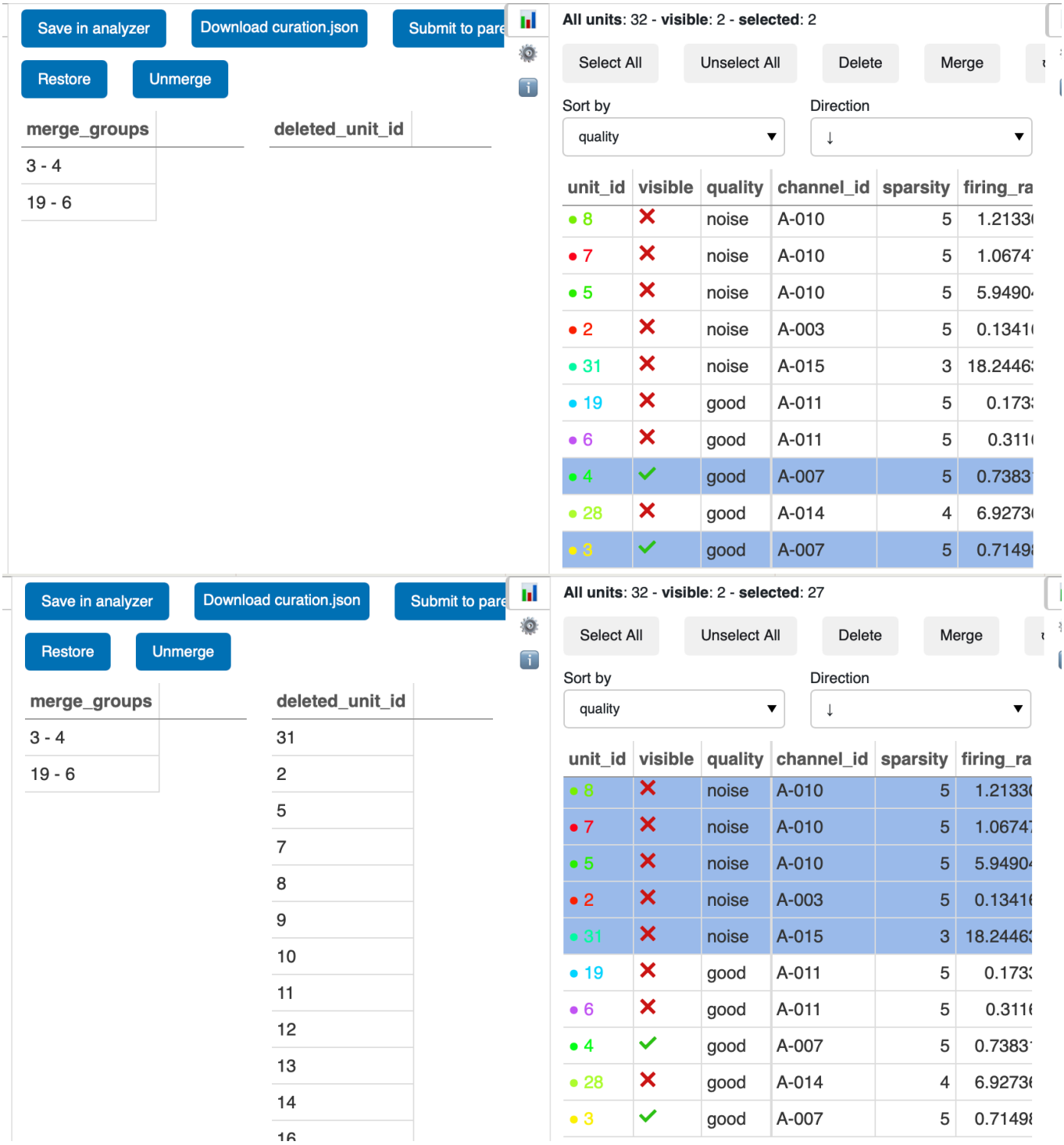
Sort by quality, and select all to see them superimposed on one another. Based on this view, 19 & 6 may be a single unit, and 3 & 28 may be a single unit. Lets drill into those.



Taking a closer look, we can see the traces stacked on top of each other, each autocorrelegram, and a cross correlelegram. The cross correlograom shows that are no spikes within 20 ms of one another between the two units, suggesting they should be merged



toggle the units in question to "visible", select both, and press merge. The new merge group will appear



Noise units can be deleted. They are not gone forever, they just leave a cleaner curation object.

Once everything is completed to your satisfaction, select "save in analyzer". This will create a hidden curation file.

10. Save Curated Data

Return to the notebook, and run the final cell so create a curated dataset. If you look in "data<patient<session<sorted" you will see a file ending in `*.zarr`. This is the final product of your sorting efforts, it contains the curated data.

11. You're done! 🎉 Be sure to end the process in the terminal with Control+Z, and restart the notebook kernel