# Collaborative scientific visualization in the browser with Figurl

Jeremy Magland and Jeff Soules, Center for Computational Mathematics, Flatiron Institute

Flatiron-Wide Autumn Meeting (FWAM), October 2023

Figurl

Follow along: https://github.com/magland

# Visualization in the scientific process

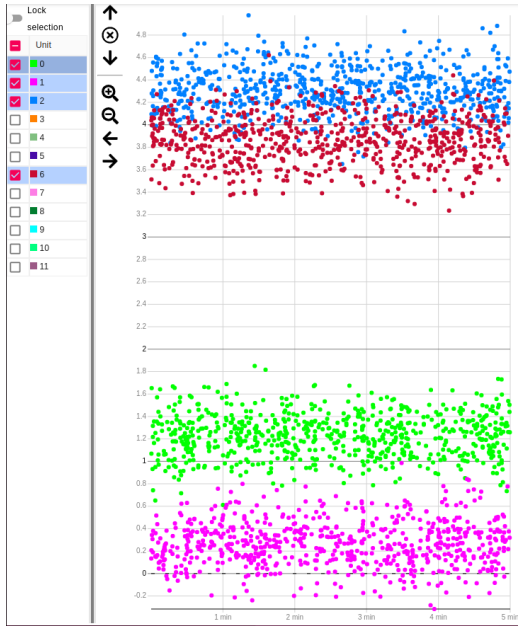
Visualization is critical at every stage of the data-centric scientific process

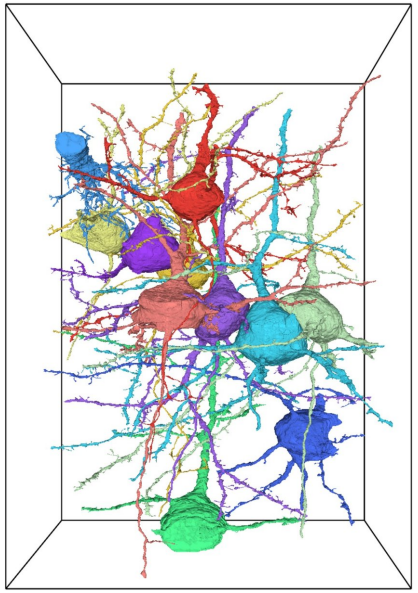
- Data exploration
- Quality control
- Curation
- Analysis and interpretation
- Communication

# Benefits of interactive visualizations

Compared with static plots, interactive visualizations enable:

- Exploration of large, complex datasets
- Exploration of parameter space
- Gain intuition about dataset
- Identification of outliers and artifacts (QC)
- Interactive / collaborative curation

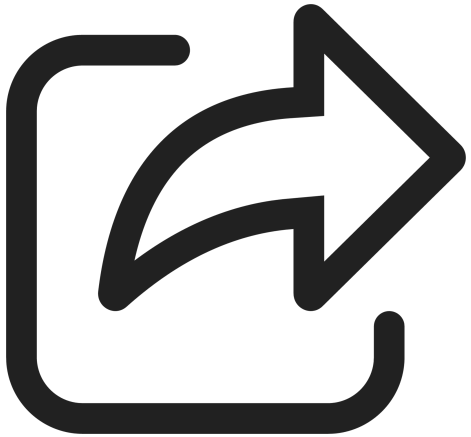


# Current limitations in sharing interactive visualizations

Interactive visualizations are not easily shared as they often require:

- Specific operating system / hardware
- Specialized software
- Transfer of large datasets
- Expertise in setting up / reproducing the visualization



# Why web-based software?

- Easy to use
- Easy to share
- Cross-platform
- Development cycle advantages (simplifies distribution, etc.)
- Integrates naturally with cloud resources
- Collaboration
- Reproducibility
- **Limitations:** no native access to local files/software, requires internet connection, limited access to previous versions, requires coding in JavaScript

FLATIRON INSTITUTE

# Existing browser-based visualization tools

Observable, Vega, Plotly, Bokeh, D3, Matplotlib, Google Charts, VisPy, Altair, Deck.gl, P5.js, Three.js, Babylon.js, A-Frame, Pixi.js, Recharts, NVD3, C3.js, Chart.js, Vega-Lite, ECharts, Highcharts, Leaflet

## ... and many more

These are powerful tools, but they typically need to be embedded in a framework to be usable and shareable.

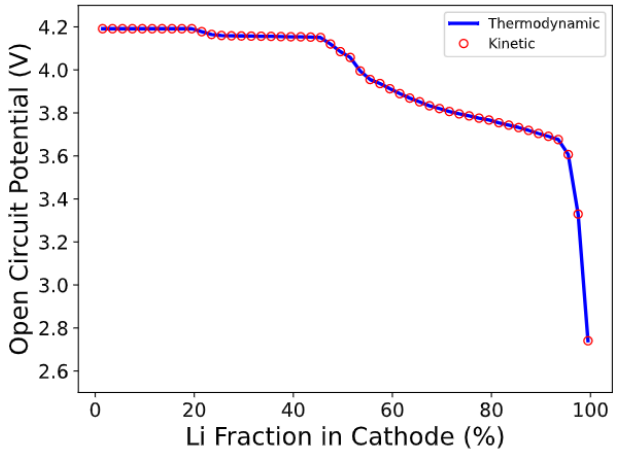
# Executable notebooks (browser-based solution)

- Jupyter, Google Colab, etc.
- **Pros**: reproducible, self-documenting, interactive
- Shareable on github, but only for static rendering of output cells
- Other systems exist for rendering in static pages, but they have limitations
- **Cons**: Requires a running backend, Cluttered by code, etc.

**Plot thermodynamic OCV, and compare to results from kinetic method**

In [10]:

```
plt.figure()
plt.plot(100 * X_Li_ca, E_cell_therm, color="b", linewidth=2.5)
plt.plot(
    100 * X_Li_ca,
    E_cell_kin,
    linewidth=0.0,
    marker="o",
    markerfacecolor="none",
    markeredgecolor="r",
)
plt.ylim([2.5, 4.3])
plt.xlabel("Li Fraction in Cathode (%)", fontsize=18)
plt.ylabel("Open Circuit Potential (V)", fontsize=18)
plt.legend(["Thermodynamic", "Kinetic"])

ax = plt.gca()

for tick in ax.xaxis.get_major_ticks():
    tick.label1.set_fontsize(14)
for tick in ax.yaxis.get_major_ticks():
    tick.label1.set_fontsize(14)
```




As one would expect, the two approaches give identical results. While both methods are incredibly fast, the thermodynamic method is roughly 30 times faster.

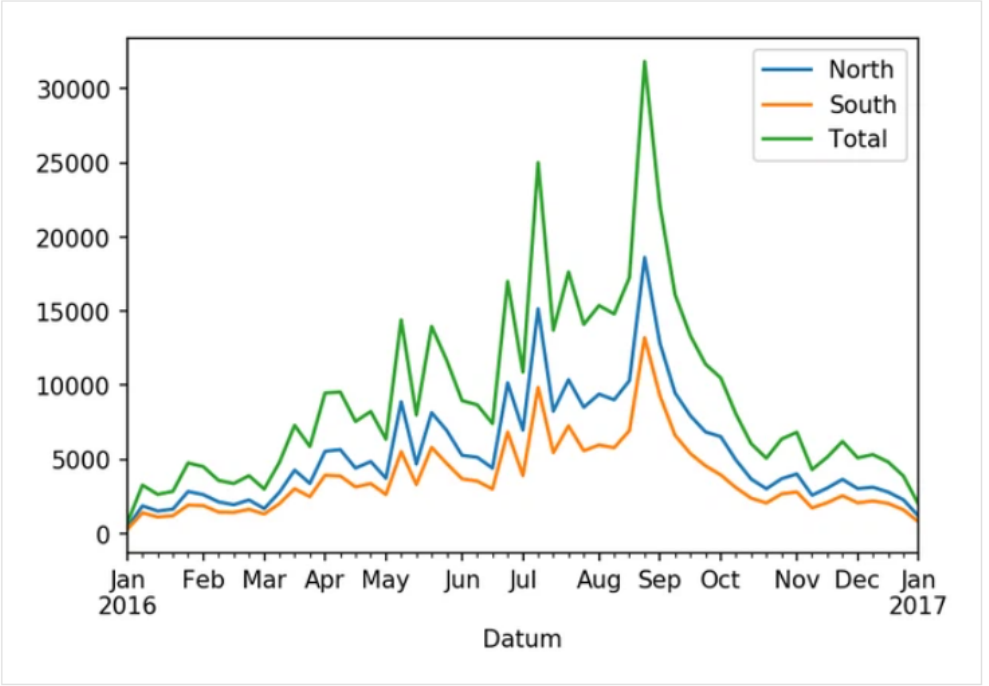
A large part of this is that the thermodynamic approach is an analytical approach (i.e. the answer is known from theory), while the kinetic approach relies on the root-finding fzero method to fit the correct voltage. Note also that the kinetic method, because of the use of Butler-Volmer kinetics, calculates the thermodynamic voltage, in order to calculate the overpotential $\eta = \Delta\Phi - \Delta\Phi_{eq}$.

However, it is at last important to note that, while slower, the kinetic method is of course more robust, and can be used to find results away from equilibrium. The thermodynamic method is only applicable at equilibrium (zero current).

# Binder: Making notebooks shareable

- Enable anyone to run code from public repositories.
- Allow authors to create interactive versions of their code.
- Scale to handle many users at once.

For example, the following figure is an analysis of an open dataset related to bicycle usage in Zurich, Switzerland:




**Total number of bikes per week traveling (by direction) past Mythenquai, Zurich. Data from City of Zurich; Notebook from Tim Head.**

You can recreate and explore this result simply by clicking the text below. You'll be taken to a live Jupyter Notebook running in the cloud. Go ahead, click it. We'll wait.

LAUNCH BINDER

FLATIRON INSTITUTE

# Binder: How it works

- **Git Repository**: Start with code/notebooks in a git repo (e.g., GitHub)
- **Dependencies**: (requirements.txt or environment.yml)
- **Docker Image**: Converts repo into Docker image
- **Launch Instance**: Starts instance of that image
- **Interactive Sessions**: Jupyer notebooks
- **Ephemeral**: Changes during session are ephemeral; lost when session ends.

FLATIRON INSTITUTE

# Binder: Limitations

- Requires running backend.
  - Depending on number of users/views, can be expensive
  - Infrastructure maintenance
- User needs to wait for the backend to start
  - Matplotlib demo: https://matplotlib.org/ (relatively fast example 30-90 sec)
  - If image is large or needs to be built, can take several minutes
- Binder links can break over time
  - The first Google hit for a live binder is broken (pandas installation error)
  - Could rely on external content or services that change over time

FLATIRON INSTITUTE

# Binder: Can be very slow to load

Here's a powerful example but very slow to load because image needs to be built at startup. (From my experience these often ultimately fail to load.)

https://pythoninchemistry.org/sim_and_scat/classical_methods/van_der_waals

Thebelab | Interact

## The van der Waals interaction

Previously, you were given the opportunity to observe particles that interact only through the van der Waals interaction. In order to achieve this it is necessary to have a **potential model**, a mathematical function, that is capable of modeling this interaction. It is known that the van der Waals interaction encompasses **two** forces; the first is the long-range attractive London dispersion and the second is the short-range Pauli exclusion principle. Therefore, the potential model must be able to model both of these aspects.

One mathematical function that is commonly applied to model the van der Waals interaction is the **Lennard-Jones** potential model [1]. This considers the attractive London dispersion forces as follows,

$$E_{\text{attractive}}(r) = -4\varepsilon\left(\frac{\sigma}{r}\right)^{6},$$

where $\sigma$ is the distance at which the potential energy between the two particles is zero, $-\varepsilon$ is the potential energy at the equilibrium separation, and $r$ is the distance between the two atoms. The Pauli exclusion principle is repulsive and only substantial over very short distances. It is commonly modelled with the following function,

$$E_{\text{repulsive}}(r) = 4\varepsilon\left(\frac{\sigma}{r}\right)^{12},$$

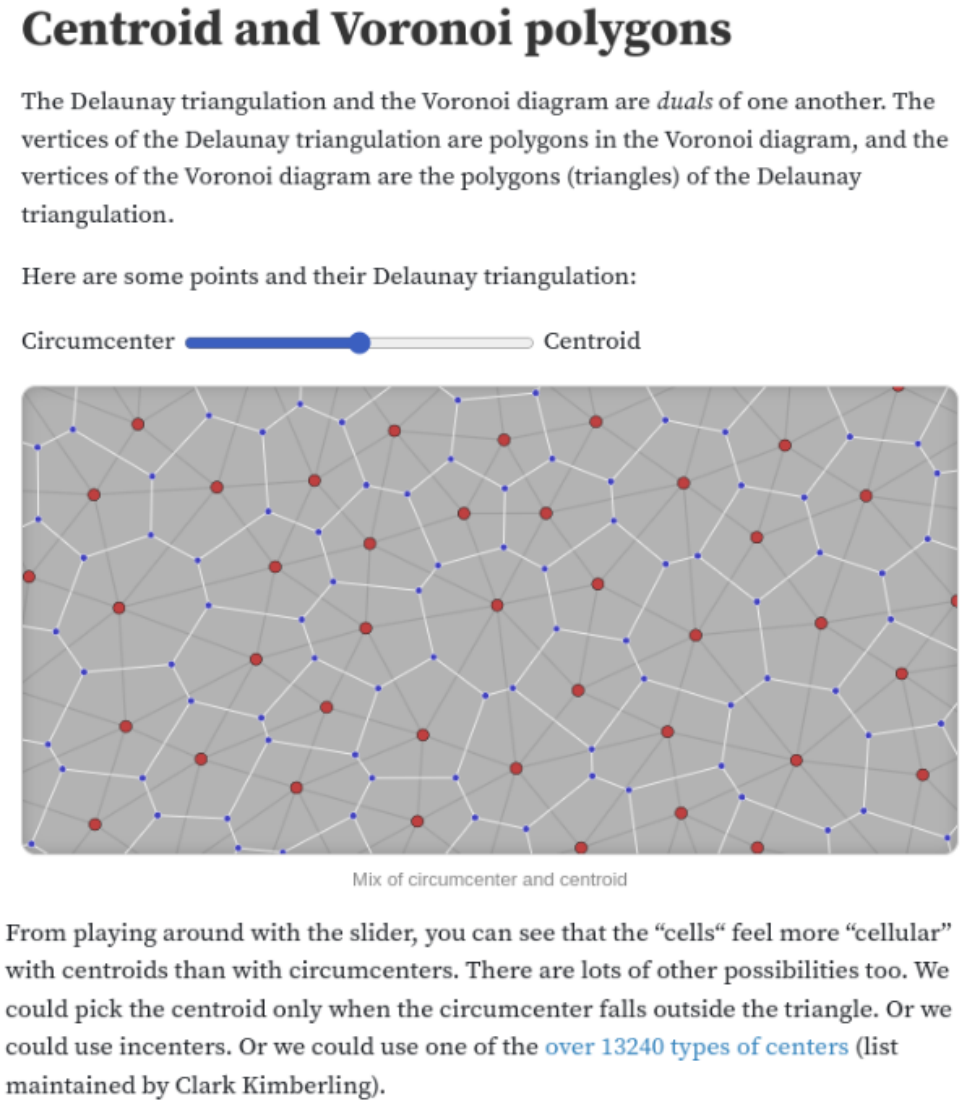
The Python code below defines each of the components of the Lennard-Jones potential and the total energy of the interaction. These are then all plotted on a single graph. The values of $\sigma$ and $\varepsilon$ are those associated with an argon-argon interaction, as defined by Rahman [2].

```
%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt

def attractive_energy(r, epsilon, sigma):
    """
    Attractive component of the Lennard-Jones
    interactionenergy.
```

# Observable (observablehq.com)

- **Interactive Exploration**: Users play with data and code
- **Collaboration**: Work with others in real-time
- **Reactivity**: Changes propagate automatically
- **Reproducibility**: Ensures results can be reproduced by others
- **Sharing and Publishing**: Disseminate findings easily

Example: centroid-and-voronoi-polygons



## Centroid and Voronoi polygons

The Delaunay triangulation and the Voronoi diagram are *duals* of one another. The vertices of the Delaunay triangulation are polygons in the Voronoi diagram, and the vertices of the Voronoi diagram are the polygons (triangles) of the Delaunay triangulation.

Here are some points and their Delaunay triangulation:

Circumcenter Centroid

Mix of circumcenter and centroid

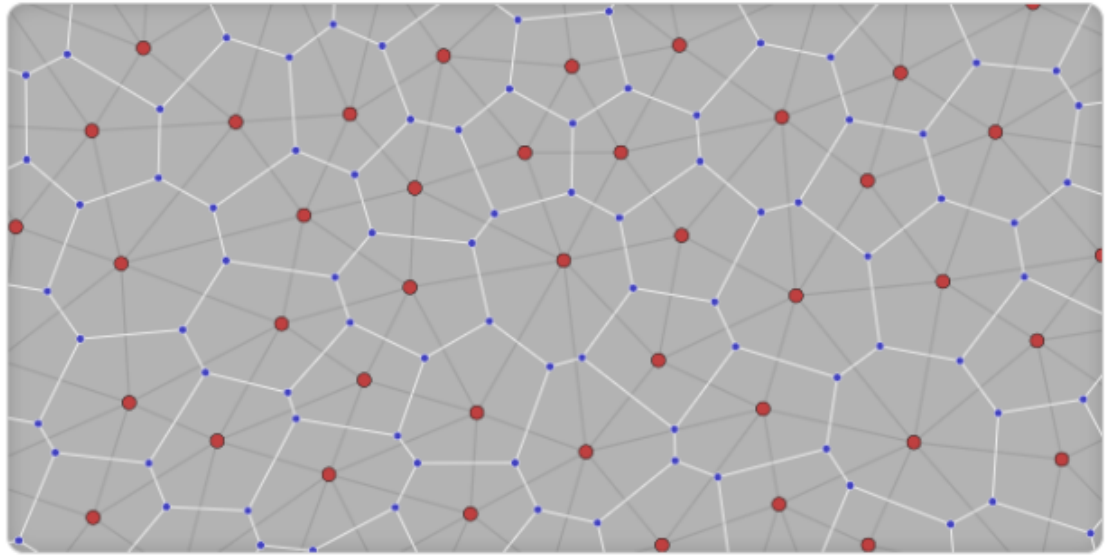
From playing around with the slider, you can see that the “cells“ feel more “cellular” with centroids than with circumcenters. There are lots of other possibilities too. We could pick the centroid only when the circumcenter falls outside the triangle. Or we could use incenters. Or we could use one of the over 13240 types of centers (list maintained by Clark Kimberling).

# Observable: Limitations

- Uses modified JavaScript (main limitation)
- Performance issues for large datasets
- Visualizations must be embedded in notebooks

Circumcenter Centroid

```
viewof slider = {
  const form = html`<form><label>Circumcenter <input name=mix type=range
min=0 max=1 step=0.01> Centroid</label></form>`;
  form.value = form.mix.valueAsNumber;
  form.mix.oninput = () => form.value = form.mix.valueAsNumber;
  return form;
}
```



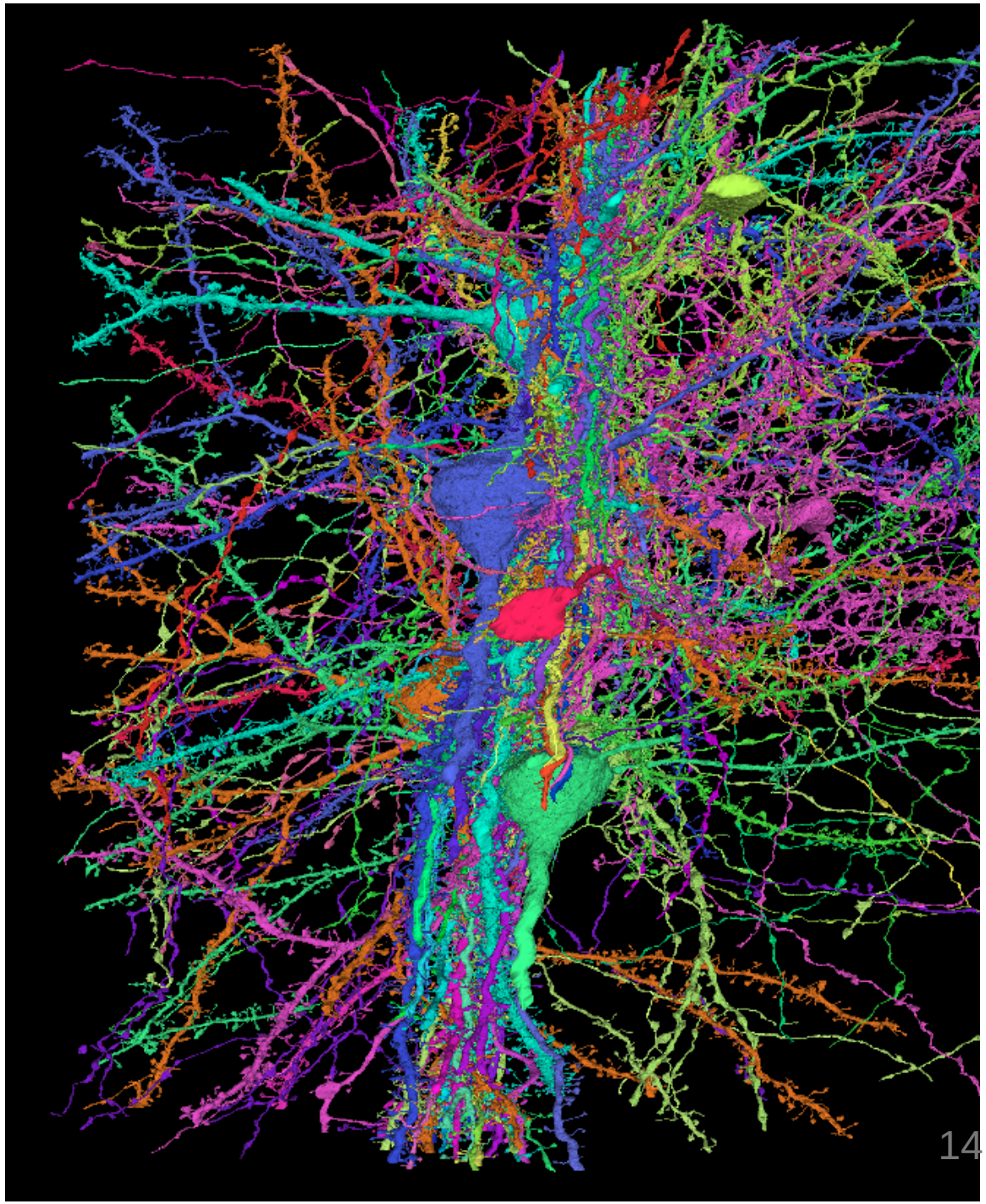
Mix of circumcenter and centroid

```
html`<figure><svg viewBox="100 100 800 400">
<g stroke-opacity="0.1">${triangleSvg}</g>
${polygonSvg(slider)}
${redPointsSvg}
${bluePointsSvg(slider)}
</svg>
<figcaption>Mix of circumcenter and centroid</figcaption></figure>`
```

FLATIRON INSTITUTE

# Clickable hyperlink approach

- Example: Neuroglancer (from Google)
- Dissemination using URLs
- No backend server required (client-side computation)
- Open viewer

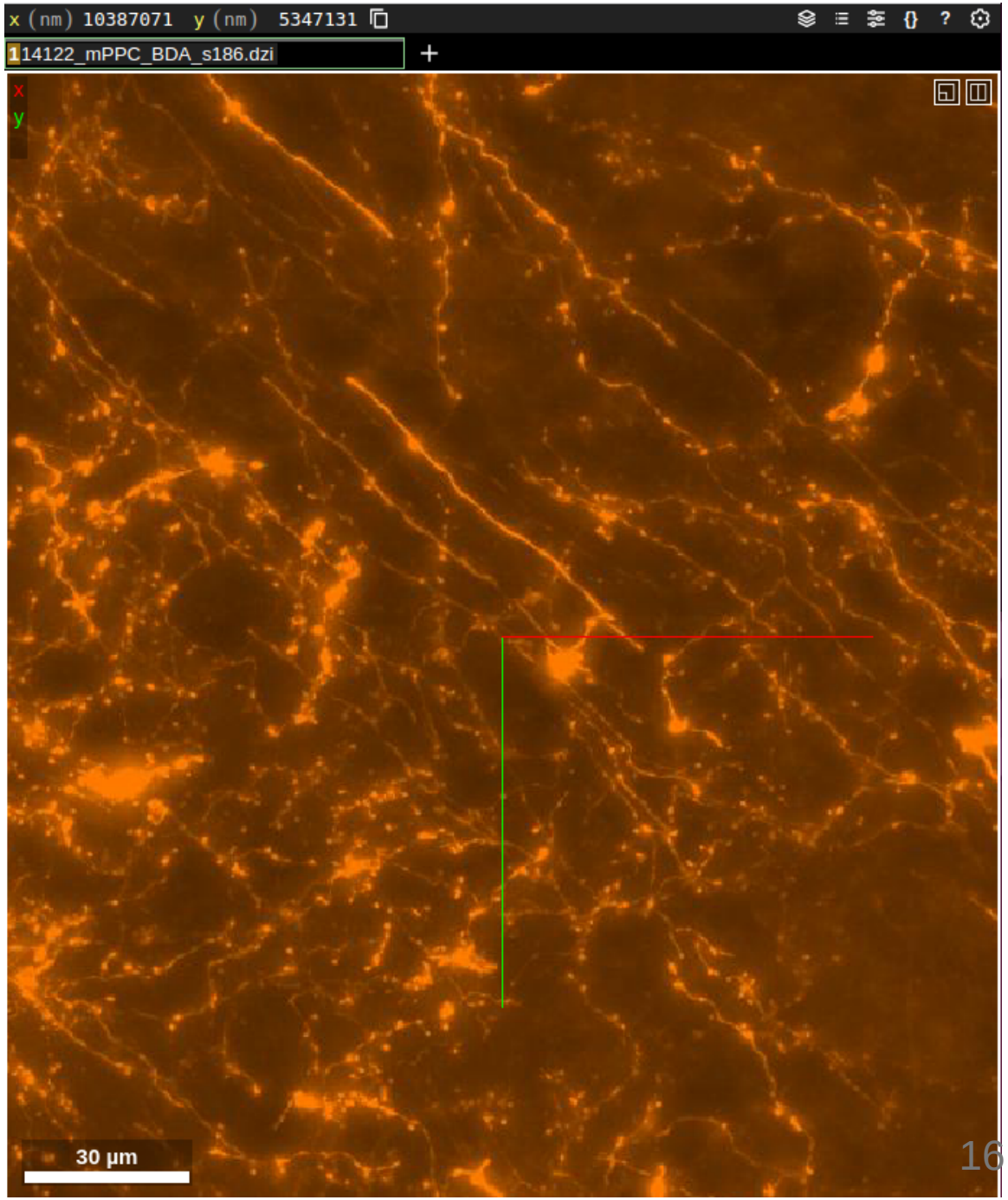


# Example Neuroglancer URL

```
https://neuroglancer-demo.appspot.com/#!{'layers':{'image':
{'type':'image'_'source':'precomputed://gs://neuroglancer-public-data/flyem_fib-
25/image'}_'ground-truth':
{'type':'segmentation'_'source':'precomputed://gs://neuroglancer-public-data/flyem_fib-
25/ground_truth'_'segments':['21894'_'22060'_'158571'_'24436'_'2515']}}_'navigation':
{'pose':{'position':{'voxelSize':[8_8_8]_'voxelCoordinates':
[2914.500732421875_3088.243408203125_4045]}}_'zoomFactor':30.097482839999932}_'perspecti
veOrientation':
[0.3143535554409027_0.8142156600952148_0.4843369424343109_-0.06040262430906296]_'perspe
ctiveZoom':443.63404517712684_'showSlices':false}
```

# Neuroglancer: Limitations

- This is a specialized tool
- Requires data be prepared and uploaded to a cloud bucket (expertise and special configuration)

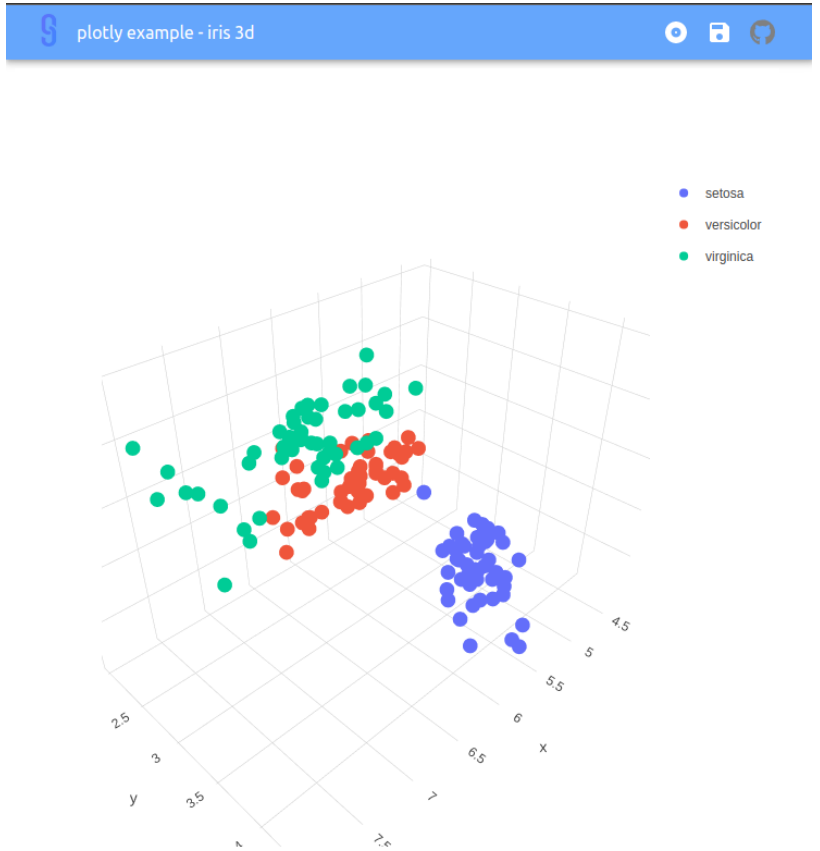


# Figurl: overview

- Simplifies sharing of interactive figures
  - Run Python script to generate shareable URL
  - Clickable hyperlink method
  - No backend server
  - Fast loading, not embedded in notebook
- Create custom visualization plugins
  - Static HTML bundles in the cloud
  - React/typescript (or other frameworks)
- Promotes scientific collaboration, communication, reproducibility

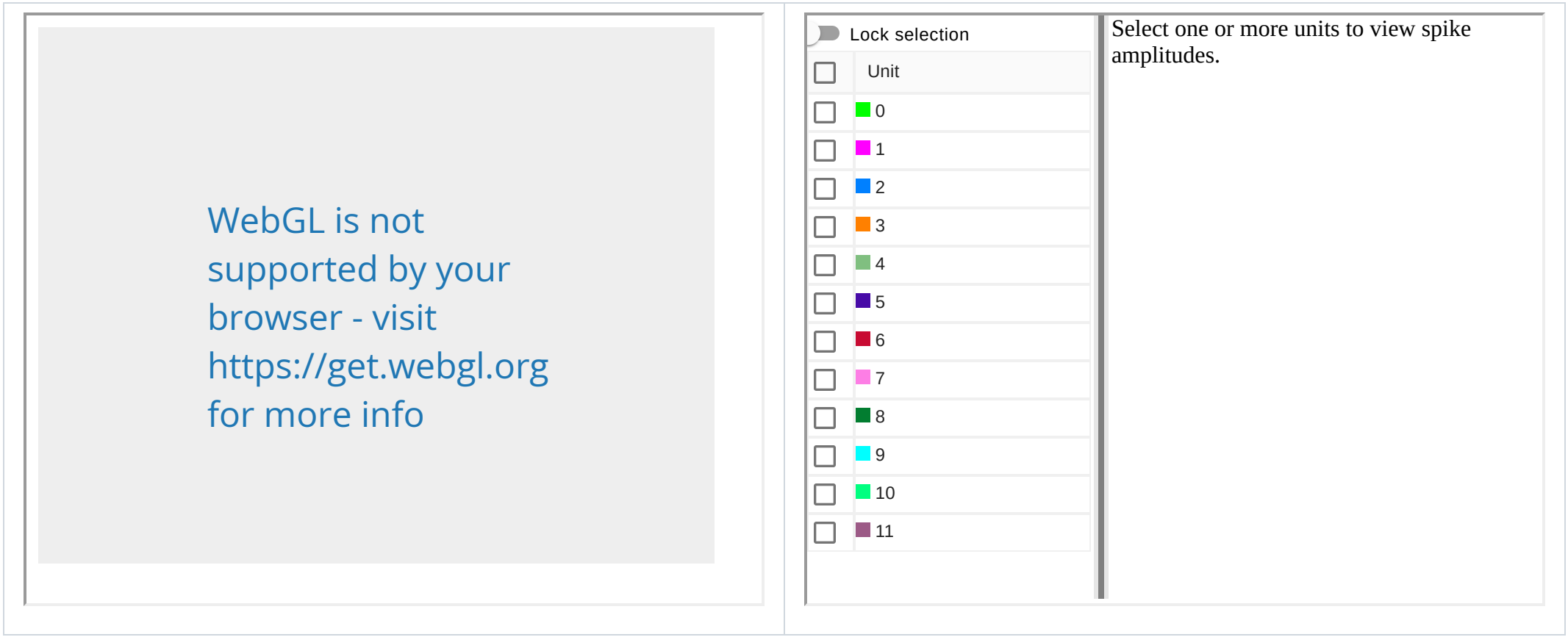
# Figurl: Plotly example

```
import plotly.express as px
import figurl as fig

# Load the iris dataset and create a Plotly figure
iris = px.data.iris()
ff = px.scatter_3d(iris, x='sepal_length',
                   y='sepal_width', z='petal_width',
                   color='species')

# Create and print the figURL
url = fig.Plotly(ff).url(label='plotly example - iris 3d')
print(url)
```

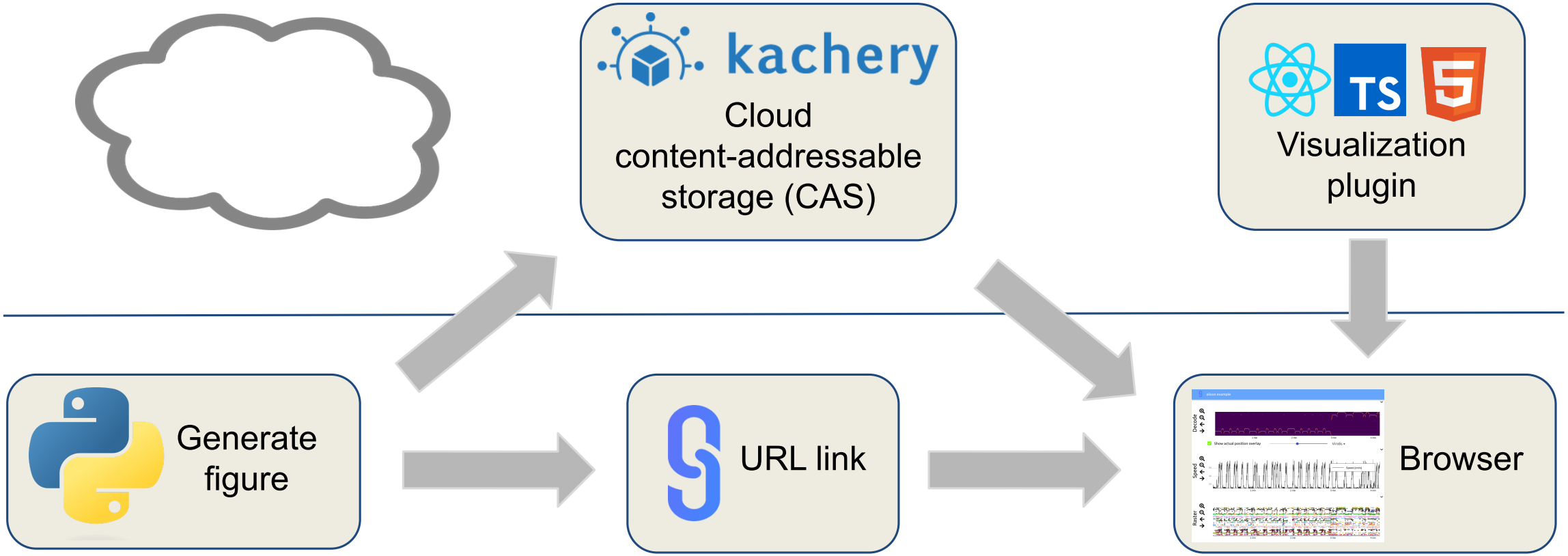



https://figurl.org/f?v=gs://figurl/plotly-1&d=sha1://5c6ec276ce9a3b20b208aaff911b037ce4052e51&label=plotly example - iris 3d

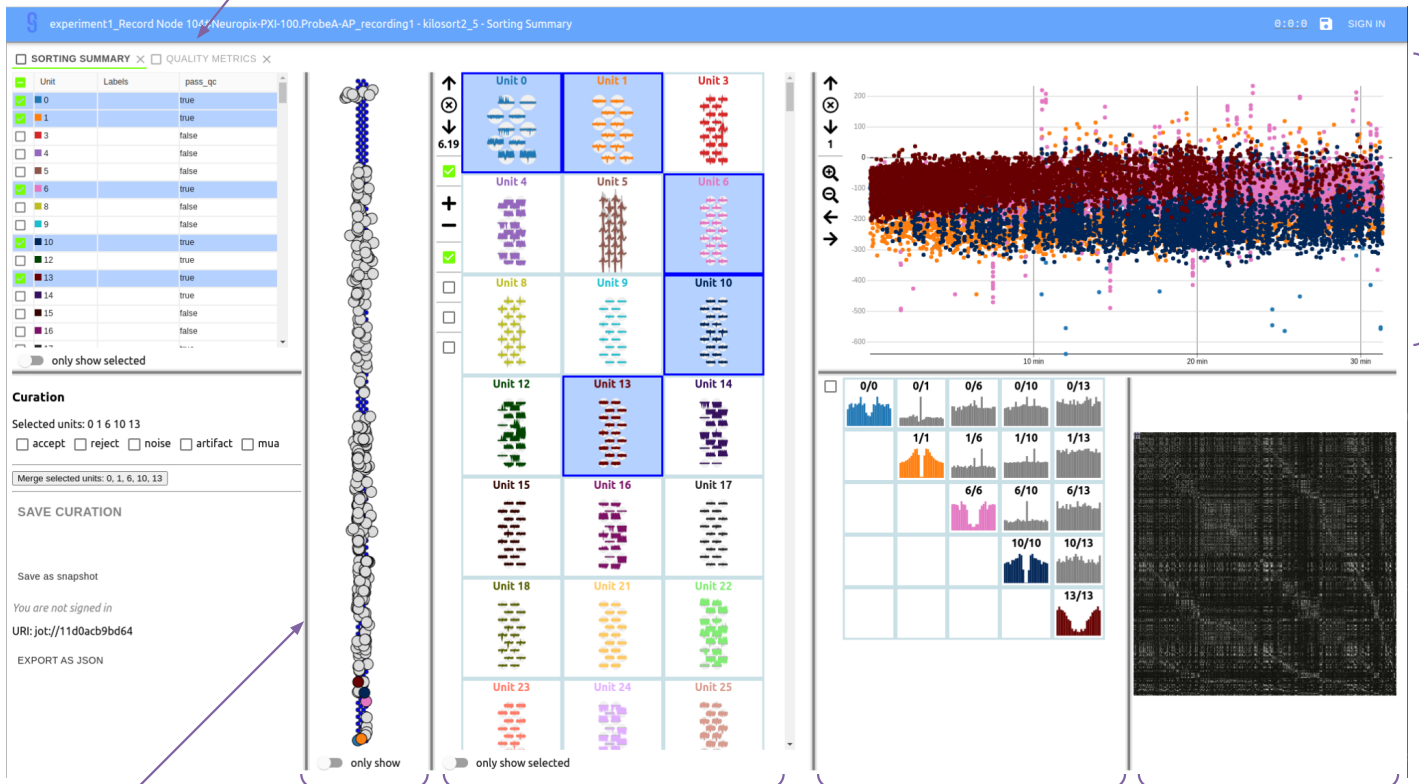
# Figurl: Embedding in presentations / websites



# Figurl architecture



# Figurl Neurophysiology Example (figurl)



FLATIRON INSTITUTE

# Figurl Neurophysiology Example

```
import spikeextractors as se

# Load the recording and sorting
recording, sorting = ...

# prepare SpikeInterface widget
widget = ...

# Prepare and print the figURL
url = widget.url(label='example')
print(url)
```

https://figurl.org/f?v=gs://figurl/spikesortingview-10&d=sha1://8d61e59b2806cf927ca1bd265923c23f5c37b990&label=experiment1_Record Node 104%23Neuropix-PXI-100.ProbeA-AP_recording1 - kilosort2_5 - Sorting Summary

FLATIRON INSTITUTE

# Figurl Timeseries Graph Example

```
import numpy as np
import sortingview.views as vv

G = vv.TimeseriesGraph(
    legend_opts={'location': 'northwest'},
    y_range=[-15, 15], hide_x_gridlines=False, hide_y_gridlines=True
)
n1 = 5000
t = np.arange(0, n1) / n1 * 10; v = t * np.cos((2 * t)**2)
G.add_line_series(name='blue line', t=t, y=v.astype(np.float32), color='blue')
n2 = 400
t = np.arange(0, n2) / n2 * 10; v = t * np.cos((2 * t)**2)
G.add_marker_series(name='red marker', t=t, y=v.astype(np.float32), color='red', radius=4)
v = t + 1
G.add_line_series(name='green dash', t=t, y=v.astype(np.float32), color='green', width=5, dash=[12, 8])
t = np.arange(0, 12) / 12 * 10; v = -t - 1
G.add_marker_series(name='black marker', t=t, y=v.astype(np.float32), color='black', radius=8, shape='square')

print(G.url(label='TimeseriesGraph-Example'))
```

https://figurl.org/f?v=gs://figurl/spikesortingview-10&d=sha1://e6ca2d115aa3b92b6da77643f07349cb8f9b5546&label=TimeseriesGraph-Example

# Anatomy of a Figurl URL

https://figurl.org/f?v=gs://figurl/spikesortingview-10&d=sha1://b2c6bf4ee2e5c866f29cb496a583e9336d36351b&label=experiment1

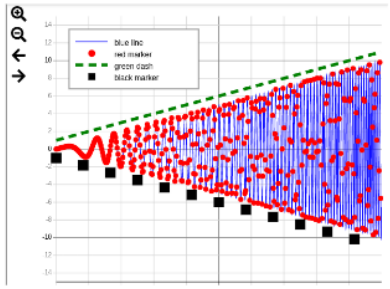
Figurl web application

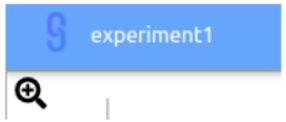
Visualization plugin

(static HTML bundle)

Data URI

Figure label





FLATIRON INSTITUTE

# Figurl Altair Example

```
import figurl as fig

# From: https://altair-viz.github.io/gallery/simple_histogram.html
import altair as alt
from vega_datasets import data

source = data.movies.url

chart = alt.Chart(source).mark_bar().encode(alt.X("IMDB_Rating:Q", bin=True), y='count()')

# Create and print the figURL
url = fig.Altair(chart).url(label='example altair chart')
print(url)
```

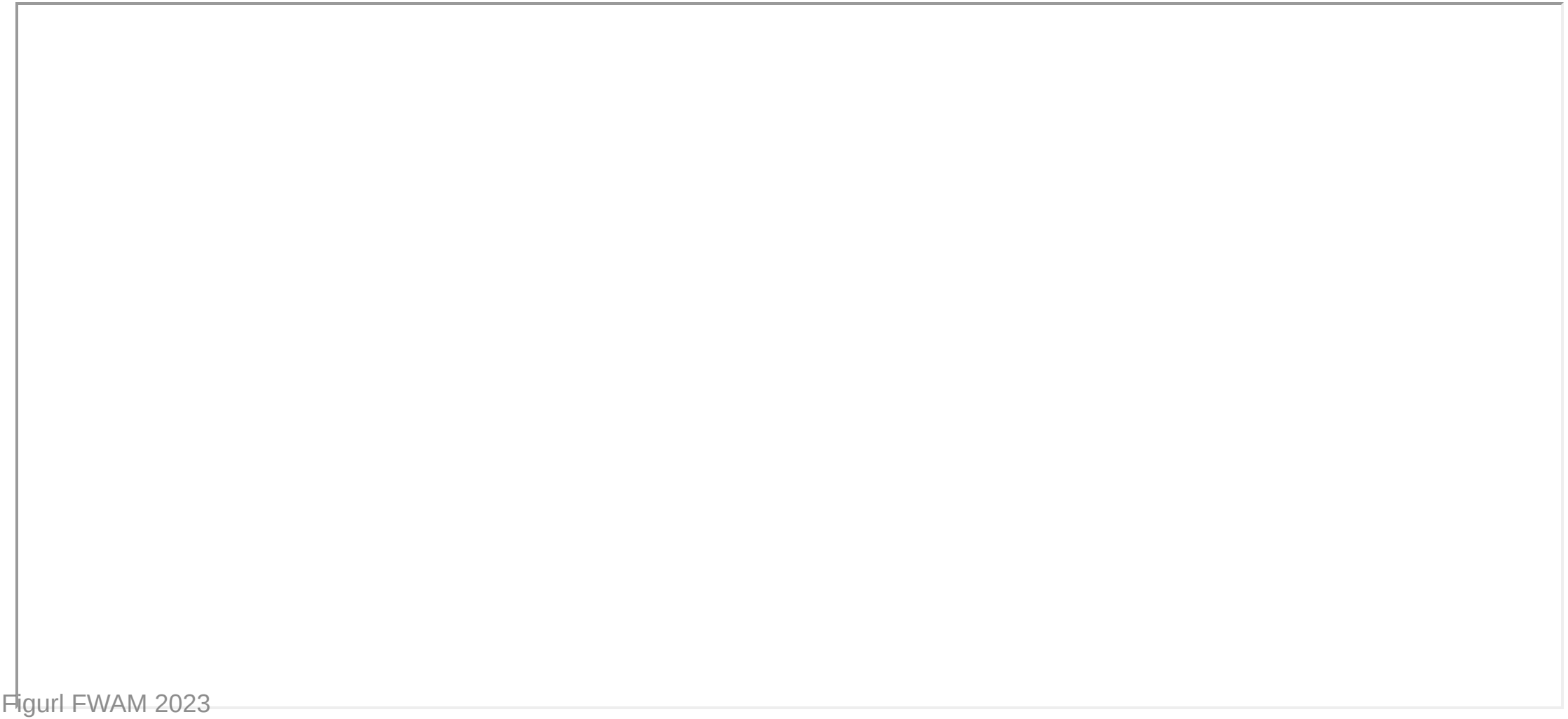
https://figurl.org/f?v=gs://figurl/vegalite-2&d=sha1://f5920cbea57e42211f7cf8306523013271 3a3f01&label=example altair chart

# Figurl 3D Surface Example

```
vtk_uri = 'sha1://e54d59b5f12d226fdfe8a0de7d66a3efd1b83d69?label=rbc_001.vtk'
vtk_path = kcl.load_file(vtk_uri)

vertices, faces = vv._parse_vtk_unstructured_grid(vtk_path)

W = vv.Workspace()
S = W.add_surface(name='red-blood-cell', vertices=vertices, faces=faces)
W.add_surface_scalar_field(name='scalarX', surface=S, data=vertices[:, 0])
W.add_surface_scalar_field(name='scalarY', surface=S, data=vertices[:, 1])
W.add_surface_scalar_field(name='scalarZ', surface=S, data=vertices[:, 2])

F = W.create_figure()
url = F.url(label='rbc_surface_scalar_fields')
print(url)
```

https://figurl.org/f?v=gs://figurl/volumeview-4&d=sha1://5a9cc08b0d8ce7a71c132b41bb5f88b9247568ba&label=rbc_surface_scalar_fields

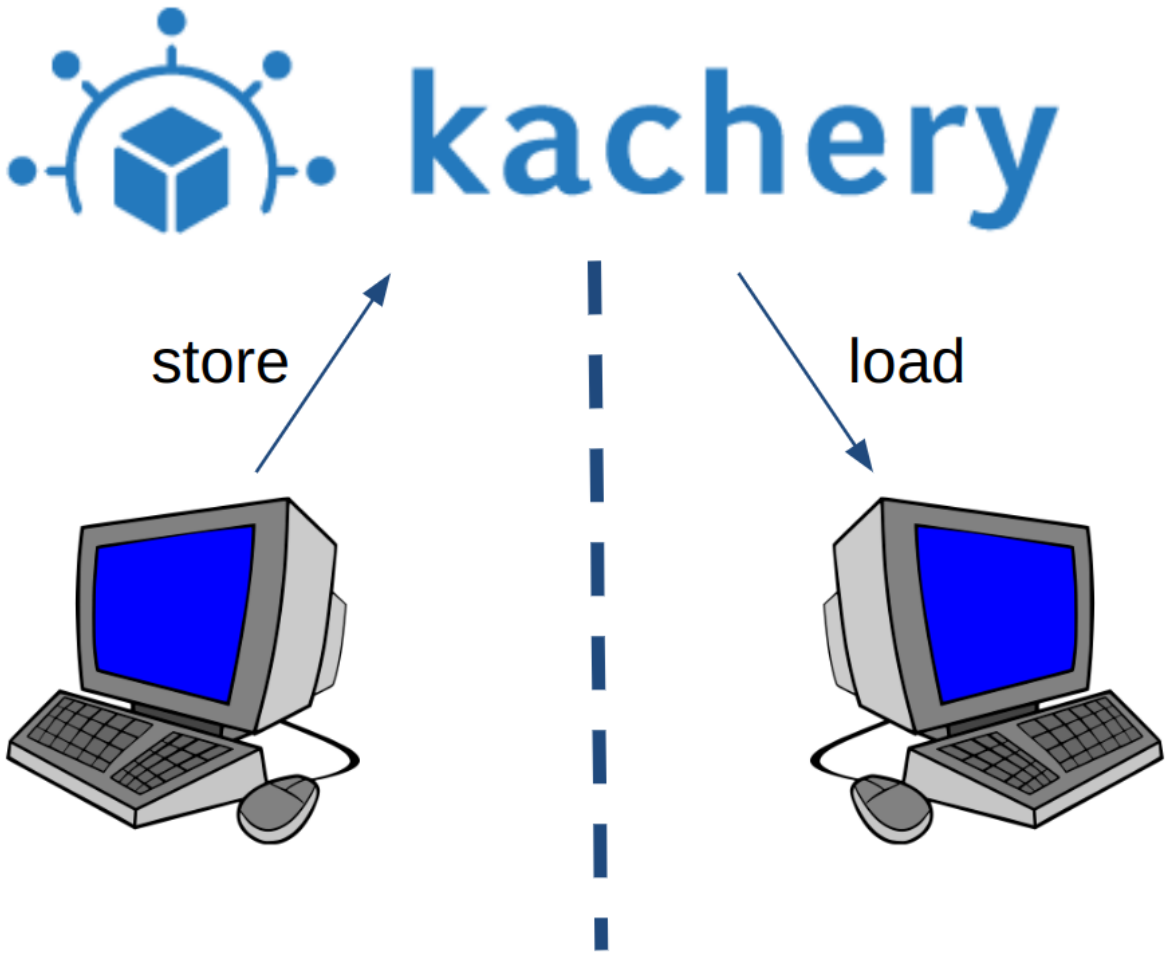
# 3D surface



# Figurl uses Kachery

Content Addressable Storage (CAS) database in the cloud

- Minimal configuration for upload
- Download from anywhere
- Python client or Command-line client
- Serverless infrastructure
- Organized into zones (labs can host zones / pay for storage)



FLATIRON INSTITUTE

# Storing kachery data

```
echo "test-content" > test_content.txt
kachery-cloud-store test_content.txt
# output:
# sha1://b971c6ef19b1d70ae8f0feb989b106c319b36230?label=test_content.txt
```

From Python

```
uri = kcl.store_text('example text', label='example.txt')
# uri = "sha1://d9e989f651cdd269d7f9bb8a215d024d8d283688?label=example.txt"
```

FLATIRON INSTITUTE

# Retrieving kachery data

```
kachery-cloud-load sha1://b971c6ef19b1d70ae8f0feb989b106c319b36230
```

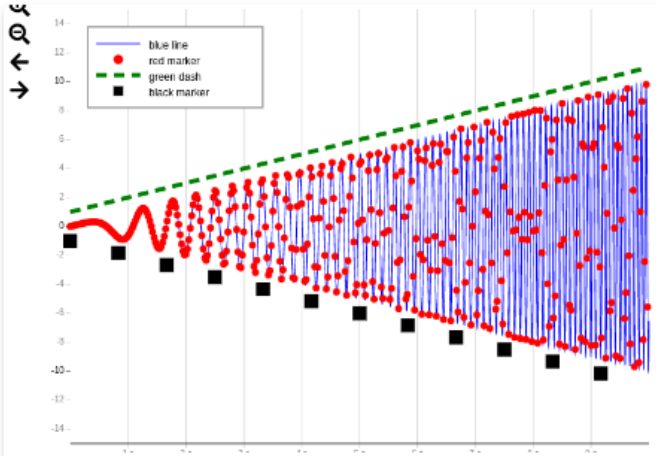
```
w = kcl.load_text('sha1://d9e989f651cdd269d7f9bb8a215d024d8d283688?label=example.txt')

x = kcl.load_json('sha1://d0d9555e376ff13a08c6d56072808e27ca32d54a?label=example.json')

y = kcl.load_npy("sha1://bb55205a2482c6db2ace544fc7d8397551110701?label=example.npy")

z = kcl.load_pkl("sha1://20d178d5a1264fc3267e38ca238c23f3e2dcd5d2?label=example.pkl")
```

# Figurl gallery


## Plotly

The Figurl Plotly interface allows any Plotly figure created in Python to be shared as a Figurl figure.

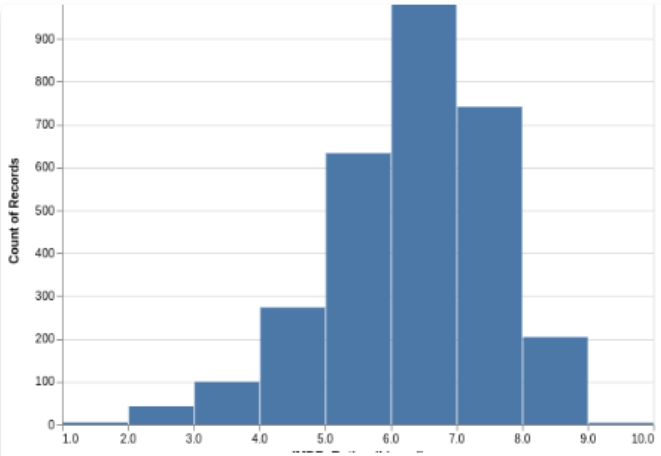
LEARN MORE




## TimeseriesGraph

Scrollable view of timeseries elements, synchronized with other timeseries views.

LEARN MORE

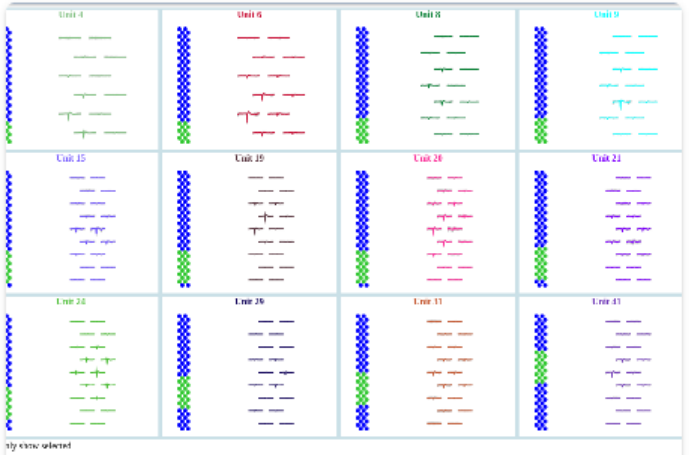



## Altair

The Figurl Altair interface allows any Altair chart created in Python to be shared as a Figurl figure.

LEARN MORE

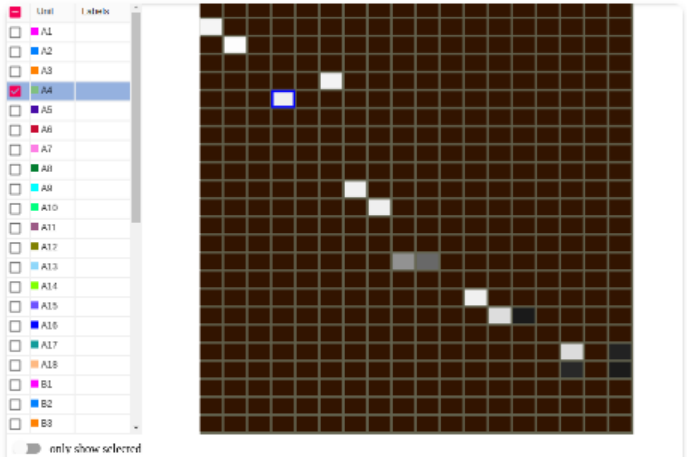
# Figurl gallery



## Average waveforms

The average waveforms plot shows the mean and standard deviations of spike waveforms for units in a spike sorting output.

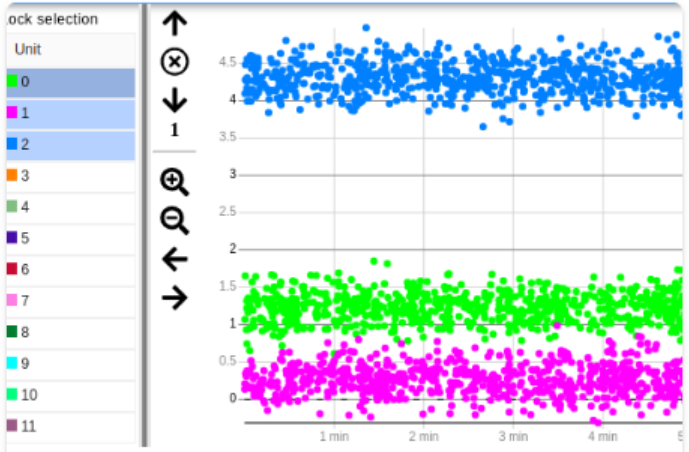
LEARN MORE



## Confusion matrix

The confusion matrix shows the agreement between different spike sortings of the same dataset.

LEARN MORE

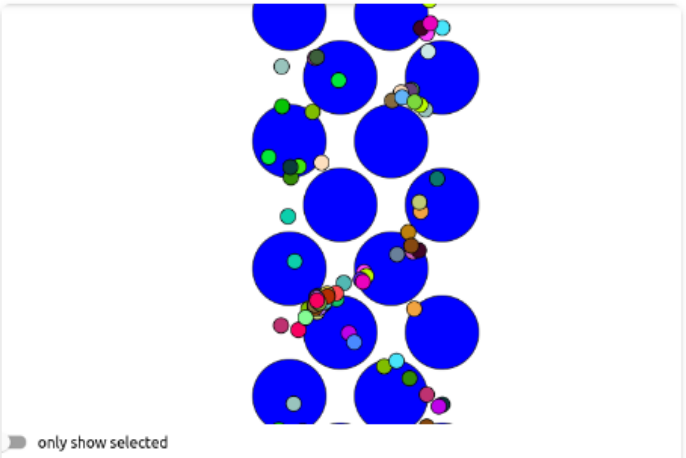



## Spike amplitudes

The spike amplitudes plot shows the peak amplitude of spike waveforms vs time for neural spike trains. This view is scrollable and synchronized with other timeseries views.

LEARN MORE

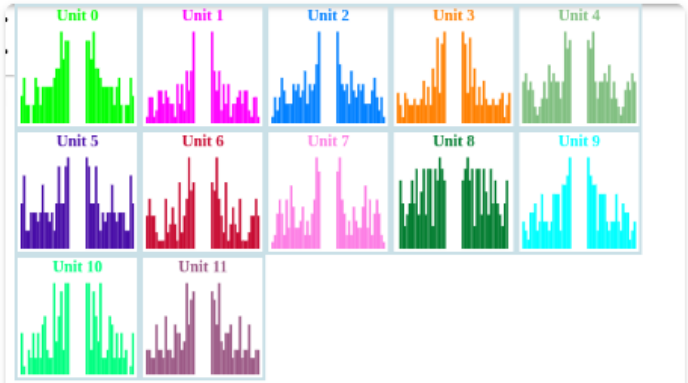
# Figurl gallery




## Unit locations

The unit locations plot shows the geometric center of mass of all spike sorting units based on their average spike waveforms.

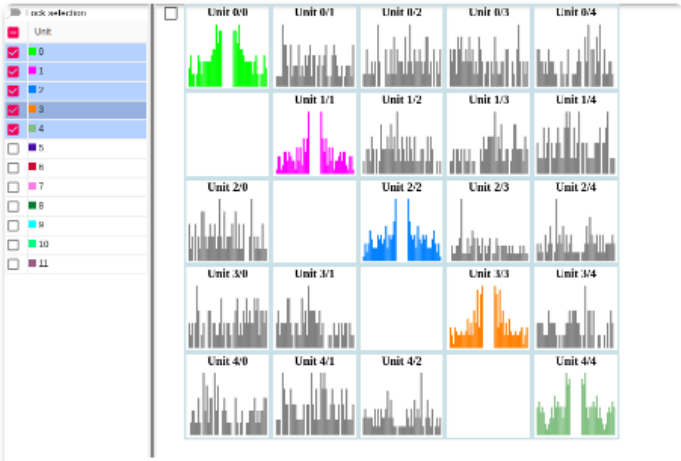
LEARN MORE




## Autocorrelograms

An autocorrelogram is a graphical representation of the auto-correlation of a spike train. When the spike train is pure (represents the firing activity of a single neuron), the histogram exhibits a central dip, reflecting the refractory period for the cell.

LEARN MORE

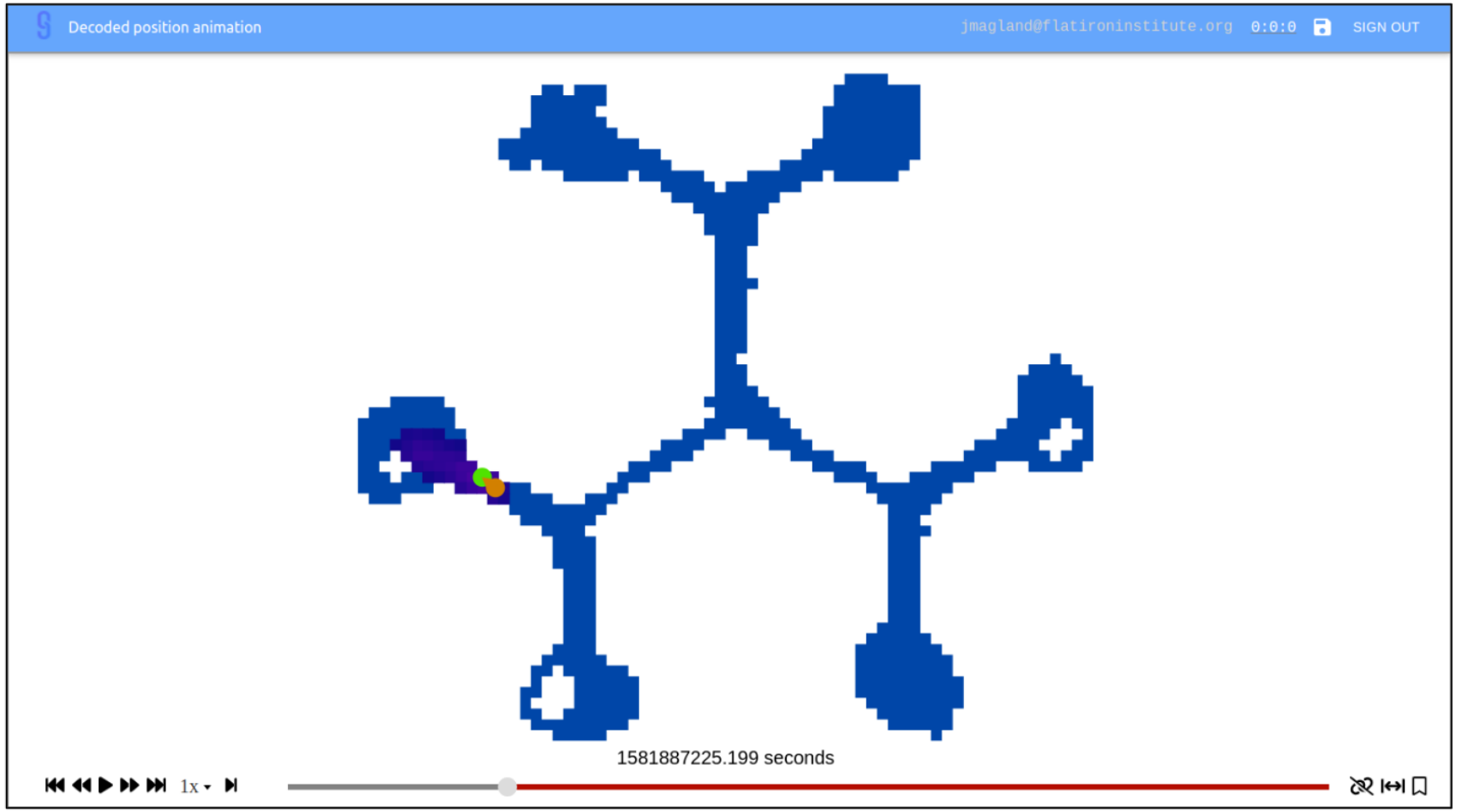


## Cross-correlograms

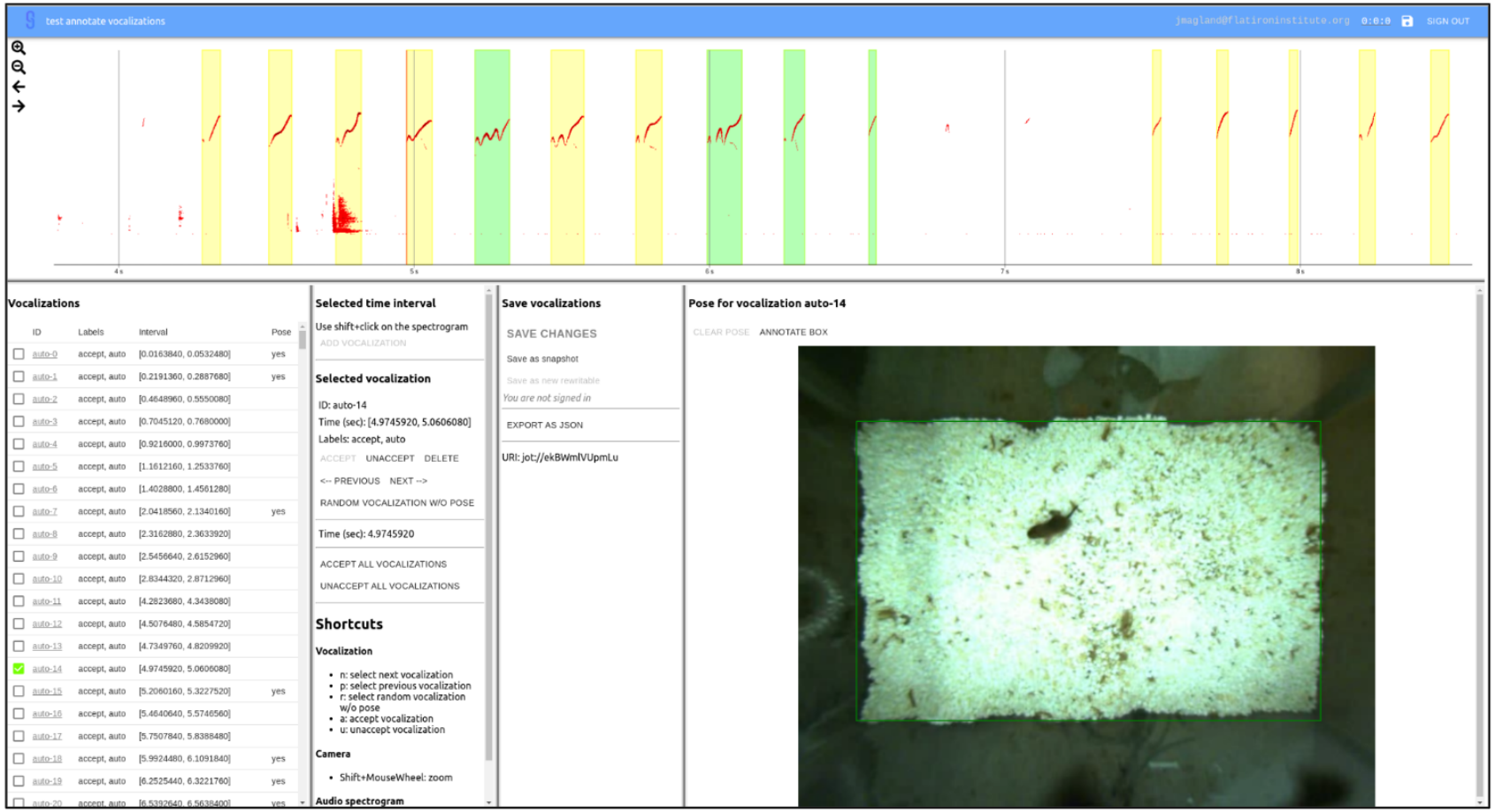
A cross-correlogram is a graphical representation of the cross-correlation between two spike trains. It is useful for analyzing the temporal relationship between the spiking activity of two units and can help determine whether clusters belong to the same neuron.

LEARN MORE

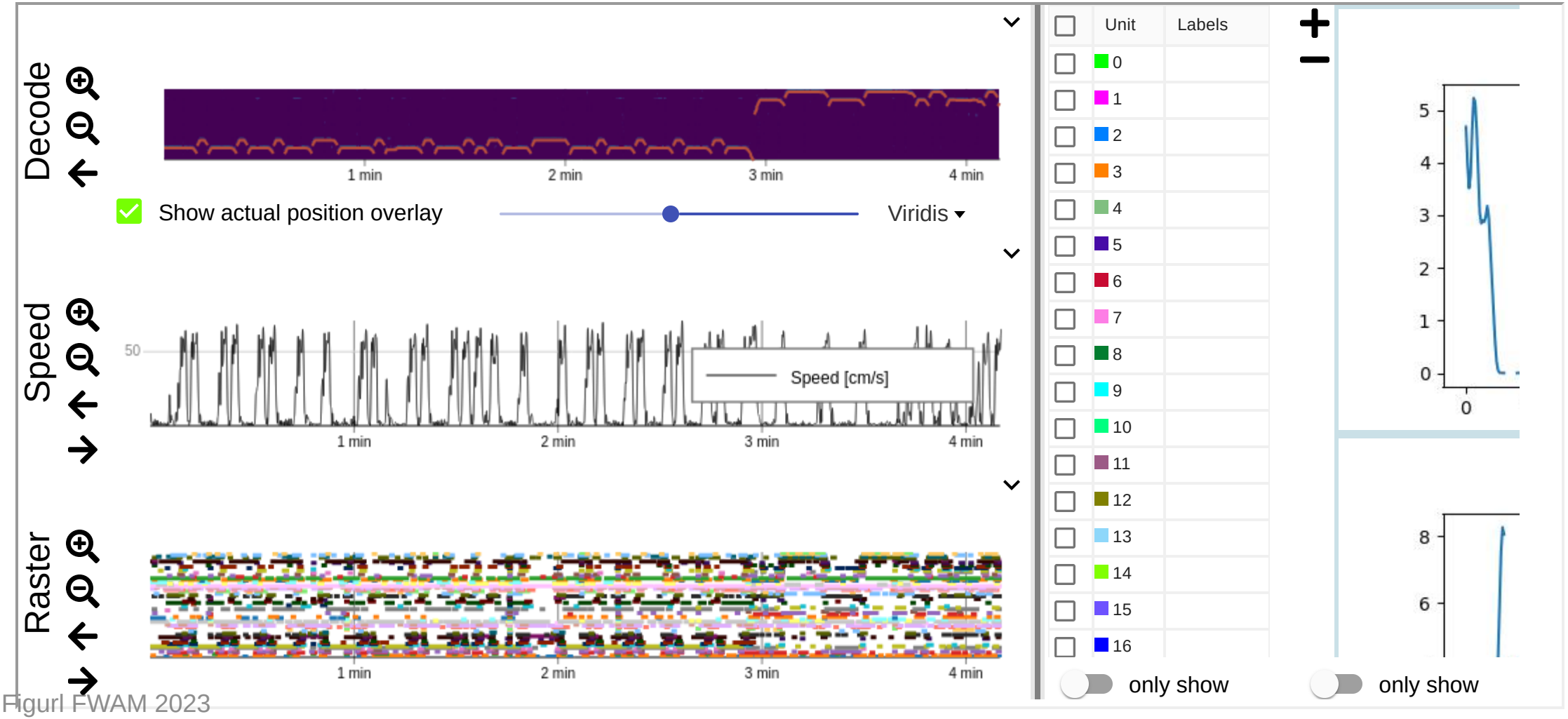
# Lab-specific visualization plugins (Loren Frank Lab)



# Lab-specific visualization plugins (with Ralph Peterson)



# Live example (link) (Loren Frank Lab)



# Live example (link) (SLEAP-type interface)

SKELETON

NODES EDGES YAML

Node

FRAMES

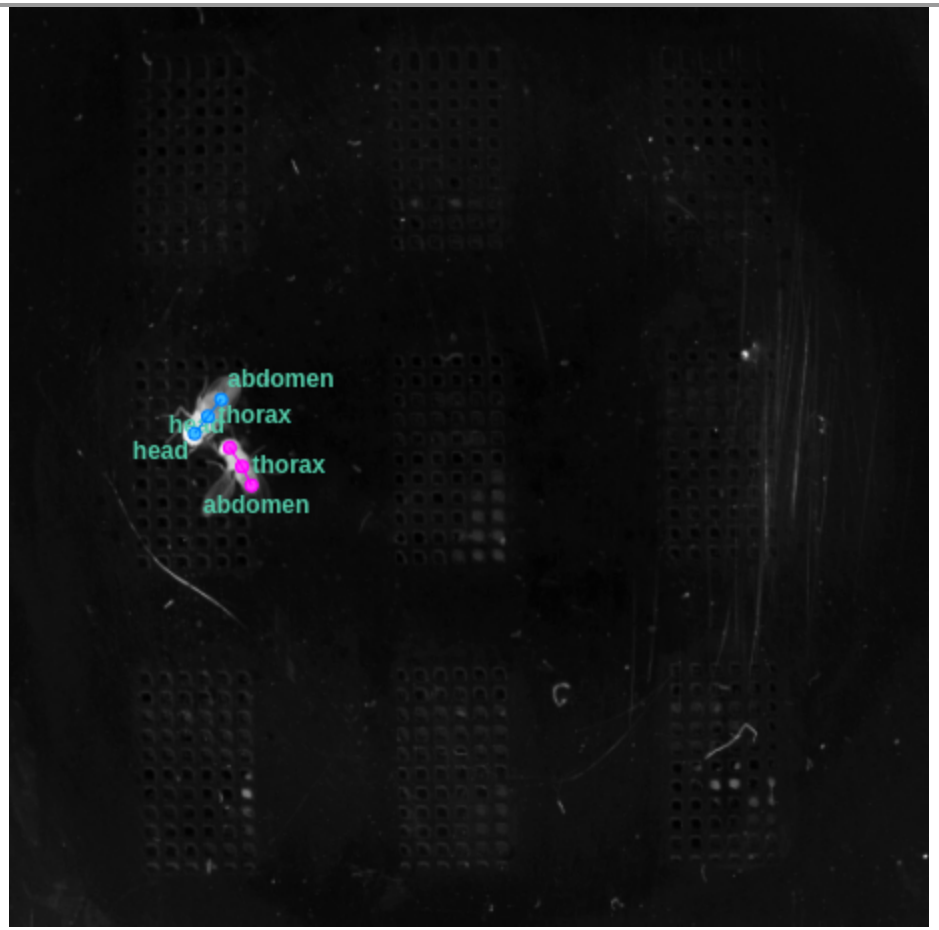
| Frame | # instances |
|---|---|
| Frame 0 | 2 |
| Frame 1 | 1 |
| Frame 2 | |

<< < prev frame 0 next > >>

Save annotation

Save to GitHub | Save snapshot | Export as JSON

Ready



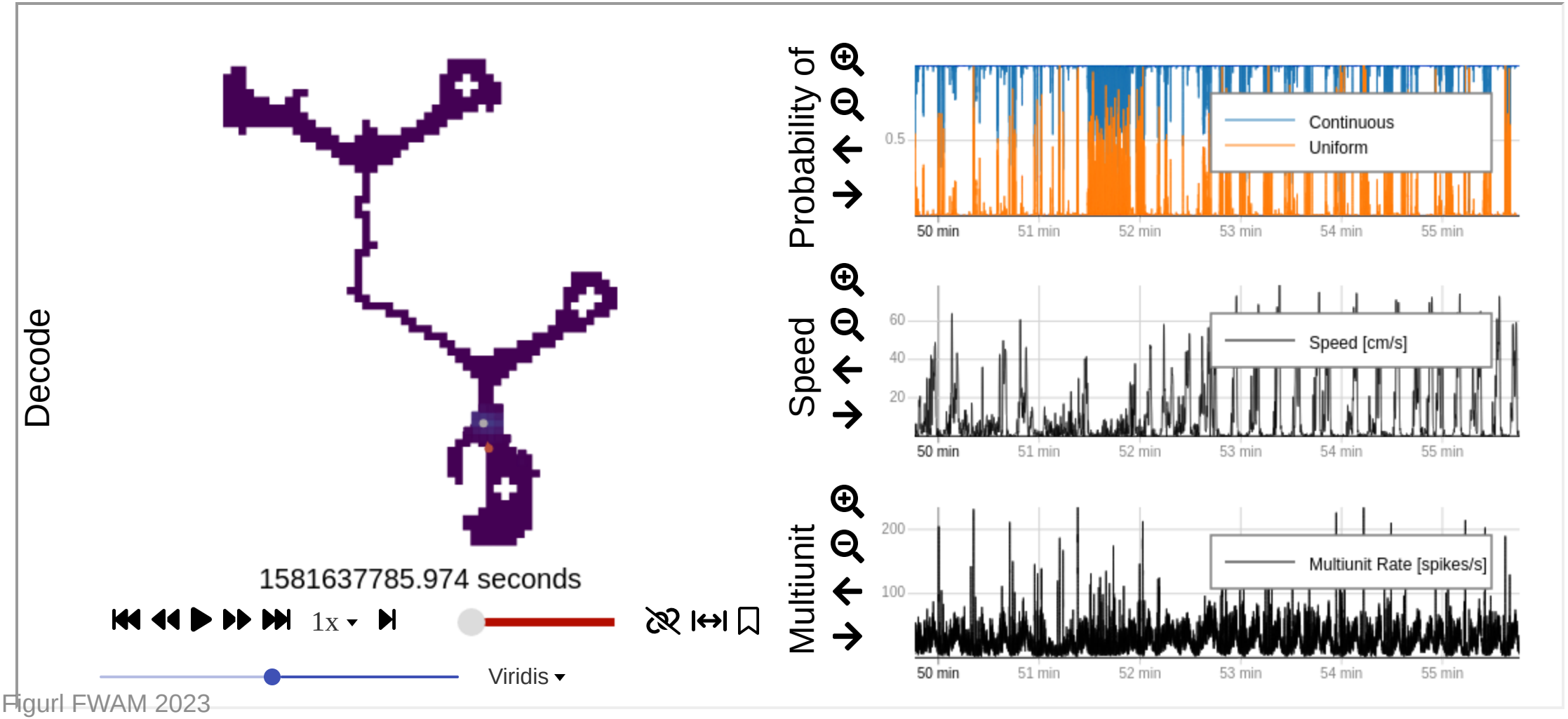

README

Instance

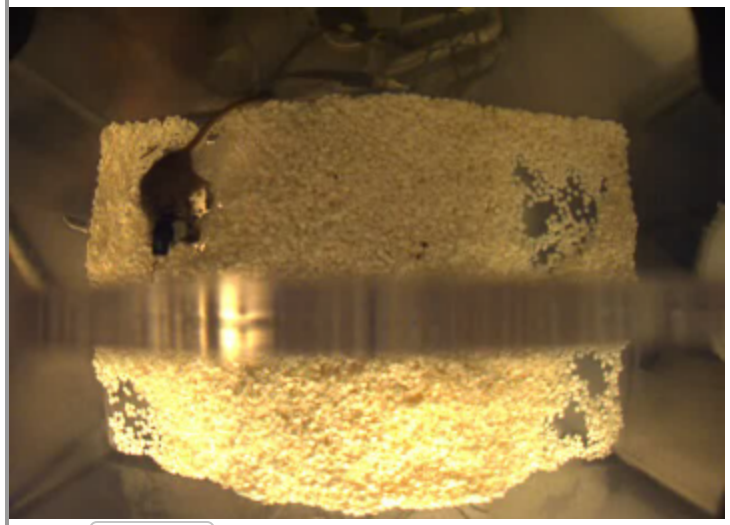
Instance 0

Instance 1

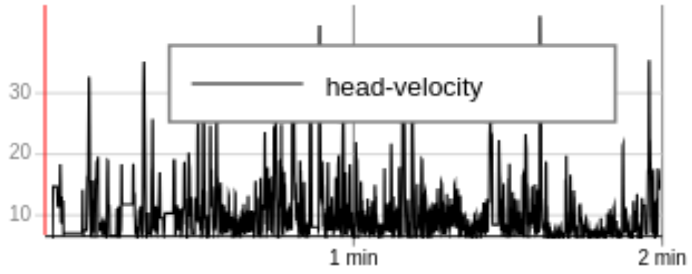
# Live example (link) (Loren Frank Lab)

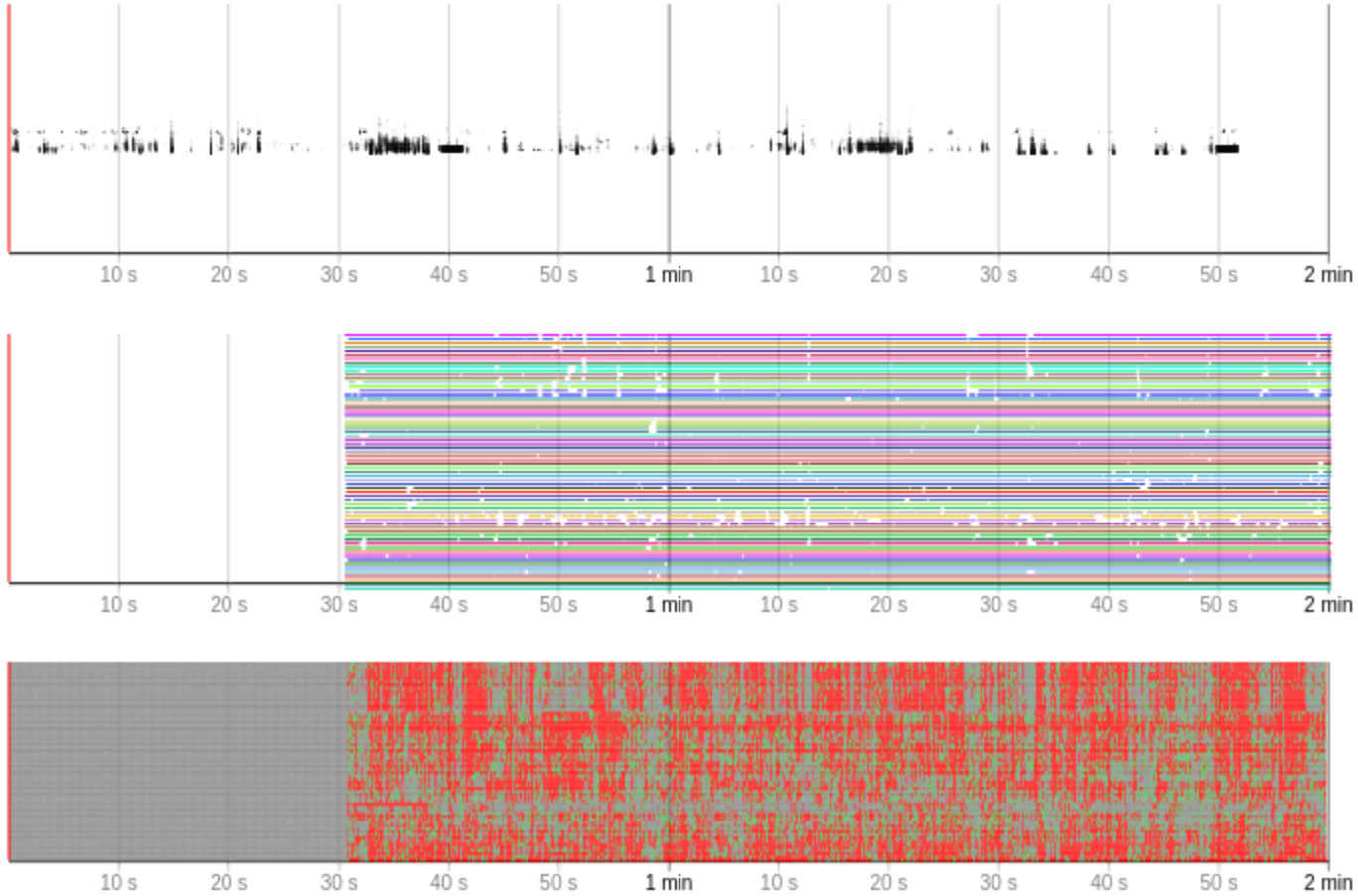


# Live example (link) (with Ralph Peterson)



1x




Bin size (s): 0.1 Color scale:

# Advanced: Embedding Figurl Figures in Markdown Documents

https://github.com/dcmnts/isosplit-paper/blob/main/isosplit.md

yields

https://doc.figurl.org/gh/dcmnts/isosplit-paper/blob/main/isosplit.md

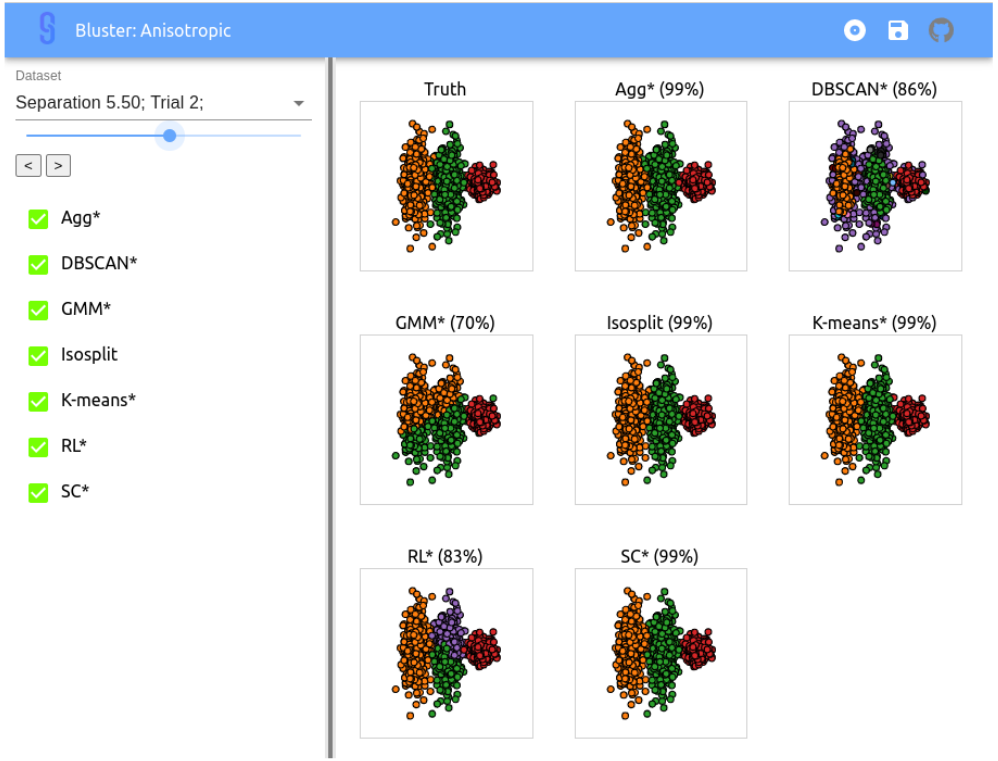



Figure AC1. Performance of clustering algorithms for three clusters, one spherical and two anisotropic, with varying separation distances. Algorithms with an asterisk have optimal parameters set based on known properties of the datasets (e.g., number of clusters). Use the interactive controls to explore all simulations.

FLATIRON INSTITUTE

# Summary

- Many powerful visualization tools exist, but frictionless sharing is still a challenge
- Figurl is an open-source browser-based visualization tool
  - Uses the clickable hyperlink method
  - Reduces friction for sharing interactive visualizations
  - Promotes scientific collaboration

FLATIRON INSTITUTE

# Future directions

- Develop additional visualization plugins
- Expand collaborations and attract developers
- Improve kachery infrastructure

# Thank you!


- Flatiron: Jeff Soules
- Frank lab: Loren Frank, Eric Denovellis, Kyu Hyun Lee, Alison Comrie, Michael Coulter
- Allen Institute: Alessio Buccino


https://github.com/flatironinstitute/figurl

https://github.com/flatironinstitute/kachery-cloud