New tools for web-enabled interactive visualizations of electrophysiological data

Electrophysiology datasets in neuroscience are becoming richer and more complex as data is collected on multiple scales, dataset sizes increase, and more sophisticated questions are asked of the data. Visualization is an essential tool for understanding these datasets at all stages of analysis, but current practices in visualization of electrophysiological data are limited in their ability to efficiently compare between visualizations (such as between spike raster plots of neurons or between regression model coefficients and raster plots) and filter complex data (such as comparing only correlations within a brain area versus comparing correlations between brain areas). Such difficulties are only magnified as the amount of data increases. This paper describes a set of composable, web-enabled interactive visualization tools developed for use in electrophysiological studies. These tools were developed to enhance exploratory data analysis, checking of raw data and statistical modeling assumptions, and data presentation for large, complex and multi-scale neuroscience data. Data from several experiments were used to test to the tools. These visualization tools are viewable in the web browser, making them easily shareable online, and open-source, allowing for modification and development by the neuroscience community.

Introduction

Current theories of brain functioning ascribe different roles to different scales: neurons, cortical layers, brain areas, networks between brain areas. For example, the Communication-through-Coherence Hypothesis postulates that communication in the brain happens primarily through phase coordination between groups of neurons (fries refs). This phase coordination between groups of neurons may differ between different layers, frequencies, within-brain areas, and between brain areas (refs). Through the use of multiple electrode arrays and laminar probes, we are beginning to collect data at these different scales and understand how they interact (ref - earl, all my electrodes). However, as electrode technology progresses, our understanding of the data is not limited by the amount of data we can collect, but by our ability to efficiently understand and model relationships in the data.

Take, for example, a typical analysis of an electrocorticography dataset in which grids of intracranial electrodes are placed across wide swaths of cortex. These grids span multiple brain areas and can measure both local field potentials and action potentials (Figure 1a). Given enough data, this allows us to ask questions about the properties at different spatial scales (units, multiunits, local field potentials, brain region summaries) and how they relate (e.g. correlation and coherence between local field potentials, local field potentials and neurons, neurons and neurons, see Figure 1b). Moreover, we can ask questions about how these change over time and/or relate to experimental conditions. This results in a dataset with many complex interrelations.

Understanding a dataset such as this becomes even more challenging as we can record from more electrodes, because the number of possible associations between electrodes scales quadratically. For example, 10 electrodes means analyzing 100 relationships between electrodes. Implantation of multielectrode ar-

rays with upwards of 100 electrodes are becoming common [@miller_all_2008; @einevoll_reliable_2012; @siegel_cortical_2015] and the number of simultaneously recorded neurons is projected to double every seven years [@stevenson_how_2011].

Visualization of data is one way that we can reduce data complexity — allowing us to make multiple simultaneous comparisons, easing the cognitive burden on working memory by efficiently encoding properties of the data into features salient to the visual system (ref). In addition, visualization is important in the understanding and checking of statistical assumptions — it helps reveal differences between the expected structure of the data and the observed data [Tukey 1972, 1979, Anscombe 1973]. This is important, from the initial stages of analysis to publication, for revising our assumptions and models and for understanding and communicating where and how often our models fail [Gelman 2004].

However, current practice with electrophysiology data relies on static visualization — requiring the generation of figures for each particular view. This makes it difficult to explore and check the data efficiently. For example, @liu_effects_2014 found that even a 500 millisecond delay between visualizations could reduce the amount of the dataset explored and affect the number of hypotheses and observations formed.

Adding interactivity allows the user to change perspectives and modify analyses on demand, facilitating comprehension and hypothesis generation. Neuroimaging studies, which generate large datasets with complex interrelations, make extensive use of interactive visualization tools (e.g. the freeview module in freesurfer, pysurfer, spm), but there are no such tools that exist for electrophysiology studies. One can design user interfaces using MATLAB, but these are hard to share and require commercial software.

We present a set of three tools aimed at providing basic interactive visualizations for electrophysiology studies: `SpectraVis` is a tool aimed at exploring task-related functional networks over time and frequency; `RasterVis` is a tool for dynamically displaying and sorting spike raster plots and peri-event histograms; and `GLMVis` is a a tool for displaying coefficients from generalized linear regressions — which are commonly used to describe the receptive field of neurons. Each visualization allows for examination of the electrophysiological signals (or summary statistic of the signal) over time relative to task-relevant events, comparison of different subjects or recording sessions, and aggregation or filtering of signals. The visualizations can also be easily linked together and static visualization can be exported for use in papers. Finally, all visualizations are web-based making them easily shareable and operating system independent, and open-source, allowing for modification and repurposing by the neuroscience community. To see working examples of all three tools, please visit http://ericdeno.com/research/.

Related Work

There have been previous attempts to make web-enabled dynamic, interactive visualizations in neuroscience. Here we describe several notable visualizations.

The pycortex webGL MRI viewer is a web-enabled interactive visualization tool that displays the results from @huth_continuous_2012. In the study, Huth and colleagues had subjects view two hours of movie trailers. They then categorized objects and actions in the movies, regressed the categories on the BOLD fMRI signals collected on the subjects watching the movies, and performed a principal components dimensionality reduction to recover a "semantic space". The visualization displays a single subject's color-coded 3D cortical surface representation of this semantic space where similar colors indicate similar categorical representations. The visualization also displays a map of the semantic space itself.

The visualization has interactive controls that allow the user to click on a category to see how it is represented throughout the cortical surface. Conversely, the user can click on a voxel on the cortical surface to see the different categories associated with that voxel. The visualization also provides button controls

that dynamically transform the view of the cortical surface (e.g. from inflated to superinflated or from superinflated to flat) and sliders that control the thresholding of the surface colors. The cortical surface can be rotated by dragging the cortical surface. A user can also obtain a permanent web link to a particular voxel of interest by clicking a button. Code for the pycortex viewer is available on Github.

The Allen Cell Types Database is a visual interface for a database of neuronal cell types in mouse lateral geniculate nucleus and primary visual cortex. The visualization has several interlinked views including an anatomical cell location view, a parallel coordinate plot of cell features, and a list of cells with more detailed information about the experiment and a brief visual summary of its morphology and electrophysiological response pattern to a step current.

Interactive controls allow the user to filter results for layer type, mouse line, and hemisphere. Clicking on cells in the anatomical cell location view, highlights that cell in the list of cells and in the parallel coordinate plot. The parallel coordinate plot provides a way to visually filter by cell features such as FI curve slope or rheobase. Clicking on cell summary brings the user to another web page with more detailed information about that neuron such as the cell's response to different types of currents and comparison to common computational models (e.g. leaky integrate and fire) fit to the data.

The Allen Mouse Brain Connectivity Atlas is a similar interface that allows the user to explore the results of 2173 tracer injection experiments on mouse brains. The visualization consists of a 3D cortical surface with labeled injection sites, a brain section image and whole brain projection image corresponding to a specific experiment, and a list of all the projection sites. A user can filter by target or source of the injection or click on an injection site to get the corresponding section and projection image.

Lastly, Freeman and colleagues have incorporated interactive visualizations into their library of distributed computing tools for large scale neuroscience [@freeman_mapping_2014]. Their tools — Lightning and Thunder — allow for basic chart types such as line graphs, network force diagrams, and heatmaps and custom visualizations to be constructed and updated in real time from data pushed from a server. They demonstrated on whole brain zebrafish recordings how these can be made into interactive visualizations. For example, using tuning curves estimated from moving stimuli in different directions, they visualized the spatial layout of the preferred direction of all neurons in the zebrafish. Mousing over the spatial layout shows the firing rate time course of a neuron in that region. Code for Lightning and Thunder are also available on Github.

Materials and Methods

Design

A set of tools is only as useful as the number of people that can use them. We decided that each visualization should adhere to a set of design principles to make them maximally useful to both the users of the visualizations and developers of visualizations based on our toolkit. To that end, our approach is to create interactive visualizations that are:

1. **Shareable** — so others can easily view the visualizations online or in print.
2. **Modular** — so the visualizations can be used independently or linked together to provide an integrated view of an electrophysiological dataset.
3. **Extendable** — so others can implement their own visualization algorithms and modify the visualizations for their own use.
4. **Configurable** — so visualizations can dynamically display different datasets or be preset to a particular view state.

Shareable

To make the visualizations shareable, the visualizations were written with modern web technologies — HTML, CSS, and Javascript. They can be deployed via a local or remotely hosted webserver and viewed with any modern browsers (Firefox 4+, Chrome 4+, Safari 4+, Opera 9.5+ and IE9+). As a result, the visualizations require no specialized software (beyond a browser) to view.

Users can share a particular state of the visualization using permanent links (permalinks) — each visualization has a button which provides the URL containing the parameters necessary to generate the current view. For example, with `SpectraVis`, if you wanted to share a snapshot of the correlation network at a specific time (e.g. 100 ms after stimulus onset), clicking on the link button would provide a URL that could then be shared with colleagues.

Additionally, static visualizations can be saved for publication purposes. Each visualization includes a button to download the current view of the visualization in scalable vector graphics (SVG) format. This format has the advantage that it can be resized without loss of resolution — making it useful for both presentations and publications — and can be imported into a graphics program of choice such as Inkscape or Illustrator for further modification. The New York Times, which frequently uses interactive graphics online and in print, has used this workflow successfully.

Modular and Extendable

Each visualization is self-contained and works independently of the other visualizations. The visualizations can be selectively linked together by using the aforementioned permalinks — which allow specification of a particular state of the linked visualization. For example, `GLMVis` might display a neuron's receptive field response to several experimental stimuli. By a simple modification of the code, this can be linked to the neuron's raster plot in `RasterVis` — showing the spiking response of the neuron to each experimental stimuli.

The visualizations' internal code is also constructed modularly — separating the internal visualization modules from data loading modules and from user interface elements such as buttons. Developers can import and export these modules selectively or make their own modules, making the visualization customizable to the developers' needs. For example, in `SpectraVis`, a developer might want to customize the layout of the correlation networks, spatially grouping nodes by brain area or some other metric. Constructing the code modularly allows a developer to implement this new layout without interring with the rest of the code internals.

Finally, each visualization has its own online software repository. The repositories are hosted on Github and can be downloaded and installed — including all software dependencies — using the node package manager (npm). This ensures the development tools, such as deploying a local webserver (allowing the user to view the visualization on their own computer without having to host it remotely), are included. This helps developers extending the visualizations to get started developing as quickly as possible. These repositories are also open-sourced under the GNU General Public License (version 2), meaning the code is available to anyone to use and develop as long as the code remains open-source.

Configurable

The visualizations are configurable in three ways: parameters can be passed through the URL, parameters can be preset using Javascript via the `init` function for each visualization, and data and data labels can be loaded using the JSON file format.

The JSON file format is a readable, XML-like format allows the visualization to dynamically display different datasets — adjusting axes, labels, and the display for each dataset. The JSON files are required to be formatted with a specific structure, details of which can be found on the wiki for each visualization. Importantly, JSON files can be exported from MATLAB structures (using, for example, the open-source toolbox JSONlab) and Python — providing an important bridge between commonly used analysis tools and the visualizations.

Results

SpectraVis

Functional network analysis is a growing area of neuroscience research, driven in part by technological improvements allowing us to record from more sensors simultaneously. However, as researchers record from more sensors, network analyses can become unwieldy and hard to interpret, because the number of possible network connections scales quadratically with the number of sensors (e.g. electrodes). Further, we expect neural processes to form dynamic networks that vary over time, frequency, and spatial scales (e.g. within and between brain regions), adding complexity to network analyses.

SpectraVis is an interactive web-based visualization application that: (1) displays task-related functional networks over time and frequency, (2) compares individual and associative measures on sensor pairs (e.g. spectra, coherences), (3) compares different measures of association (e.g. correlation vs. coherence, binary vs. weighted networks), and (4) views networks at two spatial scales (sensor- and region-of-interest-level). The different modules of SpectraVis are dynamically linked, highlighting relationships between the metrics in response to user interaction.

[@fig:figure6] shows a typical view of SpectraVis. The network view shows the anatomical location of the sensors (circles with sensor number) and edges (lines) weighted by the edge statistic. In this example, the edges are binary, representing significant changes in local field potential coherence between *Speech* — subjects reading aloud the words of the Gettysburg Address — and *Silence* at a particular frequency (10 Hz) and time (187.5 ms after speech onset)[^1]. The network has dense connectivity within and between primary motor and primary somatosensory cortices (M1 and S1). The controls can be used to play a movie of the network over time, showing increased connectivity starting within M1 300 ms before speech onset and spreading to S1 100 ms before speech onset.

Below the network view is a sensor view (dotted box) which depicts the relationship (spectra, coherences) between a selected pair of sensors (circled in black, network view, sensors 85 and 90) at all times and frequencies. Here, the edge between M1 (sensor 90) and S1 (sensor 85) represents a 10 Hz increase in speech coherence relative to silence. The increase co-occurs with higher frequency beta (15-25Hz) power suppression on the M1 sensor. Mousing over these displays updates the network view to the time-frequency bin under the cursor.

RasterVis

RasterVis incorporates two canonical visualizations for single and multiunit spiking data — the raster plot and peri-event time histogram. The raster plot describes spike times for each trial relative to a trial event. The peri-event time histogram is a simple but useful summary of how, over a series of trials, the spike times are distributed across time bins relative to the time of a trial event. Because these two types of visualizations are familiar and represent the "raw" spiking data, they are an ideal building-block visualization. Furthermore, they can also be used to compare raw spiking data to model-generated data in order to check statistical modeling assumptions (posterior predictive checks) — so they can be useful in understanding how models reflect the data.

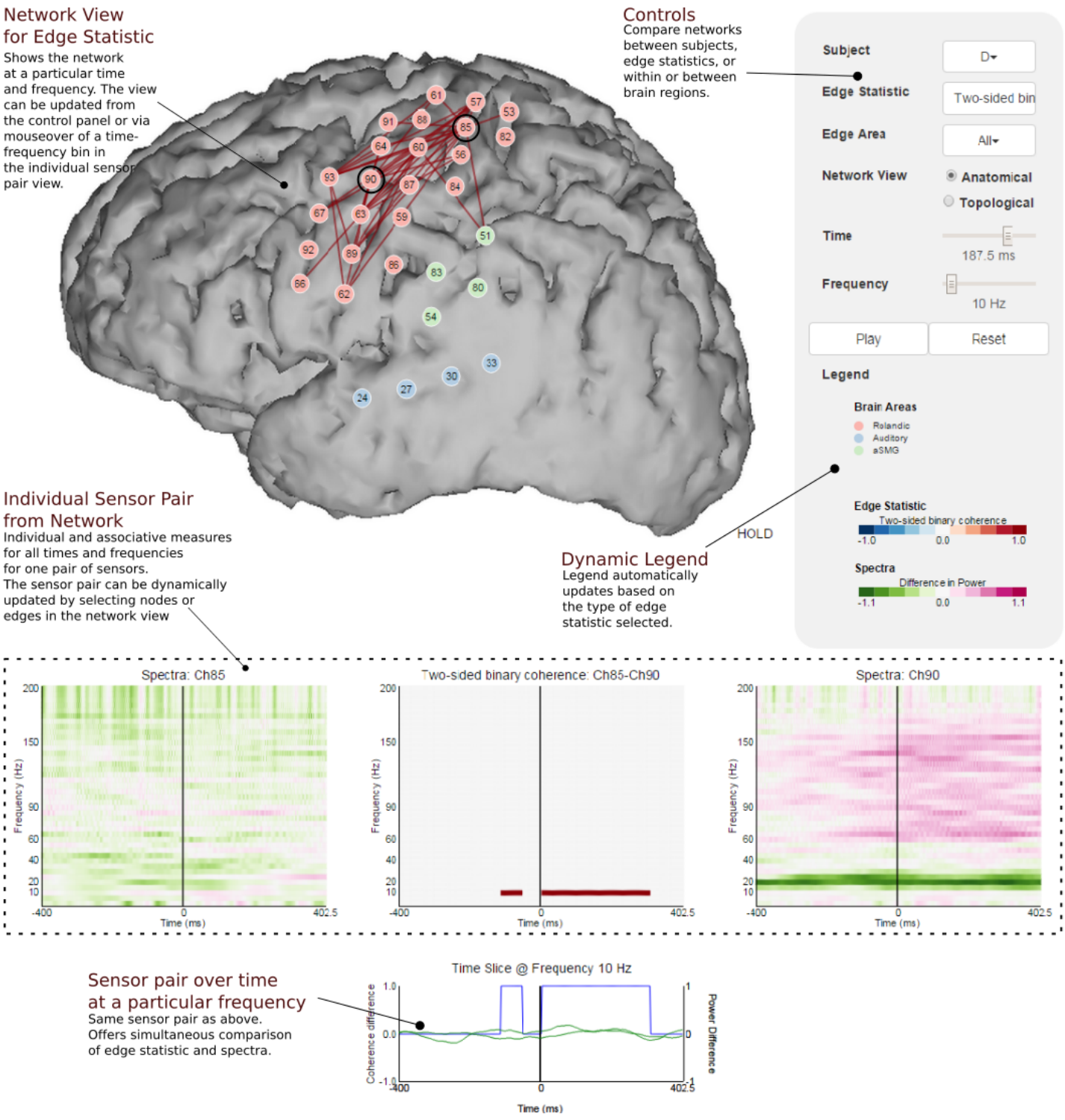

Figure 1: A static screenshot of the SpectraVis interface with the ECOG overt reading data.

RasterVis uses interactivity and animation to supplement the raster plot and peri-event time histogram in order to make it easier for the user to accomplish typical tasks in the analysis of spiking data (See [@fig:figure4] for a screenshot of the RasterVis interface).

For example, RasterVis allows for dynamic alignment of spike times and "on-the-fly" computation of peri-event histograms relative to experimental trial events (e.g. visual stimuli, timing of rewards, presentation of fixation points). Animated transitions emphasize how spike timing relative to trial event relates to another. This helps a user quickly compare the timing of individual spikes and aggregate spiking (via histogram) to different cues and conditions. Different levels of aggregation (Gaussian smoothing) for the histogram can be compared as well.

RasterVis also allows for dynamic sorting by experimental task factors. This feature creates different plots for each condition within the task factor. For example, if a task factor is a visual cue with two experimental conditions — color and orientation — sorting by the visual cue creates two plots for the color condition and the orientation condition. This is essential for multidimensional analysis which may compare several different factors and conditions.

Finally, RasterVis allows users to find and select neurons by subject, recording session, or name. This is useful for fast comparison between neurons, linking to other visualizations (other visualizations can directly link to a specific neuron by name via a parameter passed via the URL), and general exploratory analysis of the dataset.

Figure 2: A static screenshot of the RasterVis interface.

GLMVis

We also will build an interactive visualization for the generalized linear models that will: (1) show the relationship between the multiple dimensions of the model fit over time, (2) show the relationship between multiple models, and (3) show the relationship between multiple brain areas. To show the relationship between multiple dimensions, we will use the parallel coordinate plots [@wegman_hyperdimensional_1990].

[@fig:figure5] shows a prototype of the visualization, which we call glmVis.

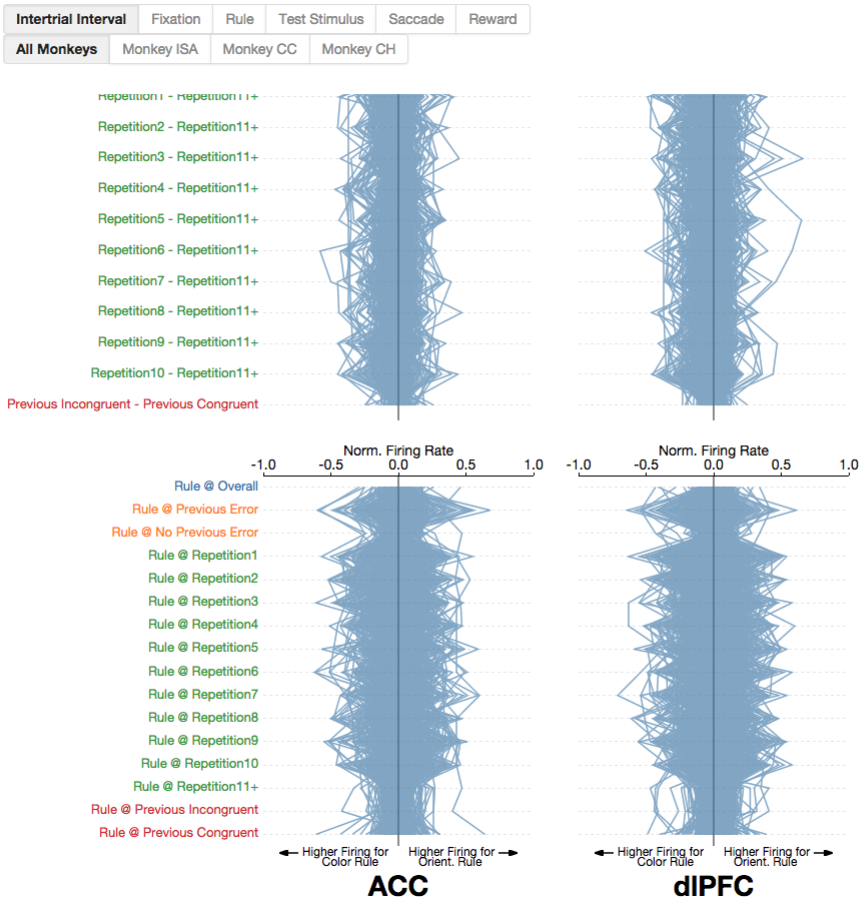

Figure 3: A static screenshot of the glmVis interface.

Discussion

Summarizing the last bit

Importance of Visualization for Open Science

Future Directions