Final Report: NED Vis

Michael Gottlieb mikemgottlieb@gmail.com
Emily Hindalong ehindalong@gmail.com
Dmitry Tebaykin dmitry.tebaykin@gmail.com

December 16, 2015

Abstract

Integrating neuroscientific data across labs is a challenging endeaver, and extracting meaning from this data is even harder still. NeuroElectro.org collects electrophysiology, neuron type, and experimental conditions data from published articles and provides all of this data in text format for analysis. Here, we present NeuroElectro Data's Visualization (NED Vis), an interactive visual interface to the data on Neuroelectro.org. It is the first tool of its kind in the Neuroscience domain and allows Neuroscientists to form hypotheses without having to download and wrangle with the data themselves. NED Vis consists of a filter menu sidebar and two tabbed panels: Overview and Explore. The Overview panel provides insight into the amount of data available for different combinations of data attributes. The Explore supports investigation of relationships between attributes of interest with interactively linked plots. The filter menu allows the user to narrow the scope of the analysis across both panels dynamically.

Introduction

Neuroscientists have conducted extensive research on the electrophysiological (ephys) properties of different neuron types, but there are barriers to comparing and aggregating results across different studies. This can be attributed to a lack of standard definitions and procedures as well as paywalls maintained by closed-access journals. To alleviate this, Tripathy et al.[1] have developed NeuroElectro - a freely available web-tool that allows users to directly compare data from different neuroscience articles. The primary goal is to "facilitate the discovery of neuron-to-neuron relationships and better understand the role of functional diversity across neuron types" [1].

NeuroElectro is a Django text mining and curation application http://neuroelectro.org developed mainly in C-Python with a javascript-based front end and an SQL back end. It currently hosts experimental data from ~900 articles and is expected to grow to host the experimental data of thousands of articles. The data for each article can be accessed by the type of neuron, its electrophysiological properties, or via a table of articles.

Neuroelectro stands apart from other text-mining projects in that it allows end users to interact with curated data directly. Most text-mining tools in the biomedical domain assume that the end user will want an association matrix for terms in a controlled vocabulary, such as MEDLINE or MeSH terms [2,3]. These tools automatically generate and output an association matrix without providing the user with a way to interface with the original data. This limits the analyses a user can perform.

Nevertheless, the power of Neuroelectro is limited in that does not provide an interactive visual interface for the data. The current visualizations provided by NeuroElectro are static: the user is only able to view data for one neuron type or one ephys property at a time in the form of fixed scatterplots.

NED Vis was designed to address this shortcoming. Our vision was to develop a new interactive visual interface that supports seamless browsing and analysis of select subsets of the data. To this end, we met with the developer of Neuroelectro to discuss high-level goals for the system. We abstracted these goals into abstract tasks and designed and developed a system accordingly. The remainder of this report discusses this process in detail.

Related Work

Solutions to Similar Problems

Neuroelectro provides some crude data visualizations in the form of static scatterplots and a single PCA analysis plot. Most text-mining tools in biomedicine do not use visualization at all, and those that do are restricted to analyses on the derived association matrix. For example, VOSviewer [4] uses colour and spatial position to visualize the semantic clustering and strength of association across text mined terms. The Trading Consequence project [5] focuses on mined trading documents supported by controlled vocabularies to generate maps of commodity trading over time.

Exploring relationships Exploration of relationships between some of the properties in Neuroelectro's dataset is supported by the current version of Neuroelectro (Brain region vs Electrophysiology only, metadata is only accessible via database). However, it is a static set of strip plots that do not account for all combinations of variables and do not allow any interaction.

Exploration of relationships is a common task in many analytics platforms such as Tableau [6], SAP BOBJ AOLAP [7] and Microsoft Excel [8]. Generally, these platforms provide a tabular view of the data in addition to customizable visualizations to enhance users exploration of the data. Our goal differs from these platforms as we are not including a tabular view, we are limiting the users choices to provide a simpler experience, and we are using plots that are not easily achieved with these platforms (e.g. interactive plots, hive plot).

Providing an overview of data Essentially, we are facing a problem of visualizing a network when we are trying to give an overview of our data. Over the years many solutions have been proposed for this type of task: hairball [9], matrix [10], arc diagram [11], call network [12], hive plot [13] are among the most common. Simply visualizing the network as a collection of nodes connected with edges (the hairball approach) seems impractical due to a large number of nodes and connections (currently: 150 nodes and ~10k edges), scaling is also a problem since the hairball only gets bigger with time. The matrix approach deserves some credit in terms of data visibility and it is a familiar visualization style to biologists, but there are 2 issues with utilizing matrices for this task:

1. Our data is 3 dimensional (neuron type, ephys. property, metadata) and 3D matrices are usually very hard to interpret, we could provide a faceted view of 1 matrix per metadata as a possible solution, but the amount of screen space that would require is enormous.
2. Matrices do not scale well, the labels get too small to be legible at some point. That said, a count matrix could provide a similar overview information to a hive plot as long as the number of parameters is small enough. In the implementation section we will discuss further our decision to use both a hive plot and a matrix approach.

A call network visualization would end up looking very similar to a hairball in our case, as a result we had to discard this possibility due to scalability issues. Arc diagrams came in as a close second as our visualization of choice - they are easy to interpret, pleasant to look at and they can scale reasonably well with the amount of evidence in the database. The problem with arc diagrams is that all nodes would end up being on one line and that does not represent the 3 distinct groups of nodes (neuron types, ephys. properties, metadata) in our data.

In the end, we decided to use hive plots for providing main overview of our data. Krzywinski, Birol, Jones and Marra [14] describe the advantages of hive plots in terms of gaining quantitative understanding when visualizing networks. They also support: multiple axes, information encoding in the nodes and edges, scaling. The one issue with hive plots is that they are a fairly new visualization style and researchers may have trouble understanding what they are looking at. However, we plan to provide a guide to interpreting the hive plot as well as provide links to the supporting literature. This feature would be on-demand and can be disabled in the application preferences.

Applications of Similar Solutions

Filter panels The filter panel paradigm, where one panel is used to control what data appears in the main panel, is well established in visualization domain [15,16,17]. An alternative solution is the filter bar, which uses less screen real estate [18]. However, we have opted to stick with filter panels because they will never interfere with the main view and will make it easier for the user to track which filters are applied at any given time. Furthermore, the number of filtering options that we offer will require a larger section of the screen.

There are two basic attribute-based filtering paradigms: drill-down and parallel selection [19]. As the referenced blog post describes, Amazon uses drill-down filtering and Kayak uses parallel filtering. Our solution uses a hybrid of these, allowing the user to drill-down categories and apply parallel selection within. We intend to refer to this blog post when designing the specific details of our filter panel.

Connected scatterplots Since Neuroelectro data is rather diverse (dozens of electrophysiology properties for each of over one hundred neuron types), we plan to utilize scatterplots [20] and connected scatterplots [21] for answering research-oriented questions. Haroz et al. showed the effectiveness of the latter in representing time-series data: even though connected scatterplots are novel to many users, they are excellent at being intuitive to understand and capturing and holding the viewer's attention.

Linked highlighting There are a number of interaction approaches to linked highlighting in scatter plots [22]. Through our consultation with the stakeholder, linked highlighting on hover was emphasized as a critical element. However, this is not the only means of linked highlighting available. For example, linked brushing, where the user selects a subset of points to be highlighted, is a popular choice for multi-selection [22, 23].

Hive plot Overviews of the data will make use of the work done by Krzywinski et al. [14] and Hanson [24] for our proposed hive plot. Various good examples of hive plot visualizations of networks are shown on the hive plot website [13]. We have developed our visualization based on the features that are most meaningful for our data and the questions we are trying to answer with the view (sparsity of data, node degree, edge weights, outliers and general trends in the data). We also use a matrix and a table approach to supplement the hive plot, since alone the hive plot does not provide enough details about connections between data types.

Colour We have decided to use colour as a channel for linked-highlighting. Previous research suggests that colour is one of the most powerful visualization channels [25,26] and, when used correctly, it can provide insight into the data so intuitively that the user wouldn't even need a legend to understand what kind of data the channel encodes for. On the overview panel colour is used as a measure of how many articles exist for each connection between nodes.

Data and Task Abstractions

Domain-specific data and tasks

NeuroElectro is a database of neuroscience articles and it applies text-mining as well as curation approaches to extract electrophysiology measurements, neuron type information and experimental setup conditions (metadata) from the html-encoded articles. At this point, text-mining alone is not reliable enough since neuroscientists authoring the articles in questions were not writing them using guidelines. As a result, each article is a snowflake of sorts - even with very well written algorithms automated text-mining is not at human text interpretation level yet. Hence the need for training undergraduate curators to verify the text-mining results and correct its errors. We focus on visualizing only the curated data, meaning that we have ~1000 articles worth of data. Note that not all articles contain all data types that NeuroElectro is able to store.

Domain-specific data types

1. Electrophysiology measurements
    - These are intrinsic neuron properties: membrane potential at rest, spike threshold (minimum membrane potential that causes a spike), input resistance, rheobase (minimum amount of current one needs to inject to cause an spike), etc.
    - Neurons communicate with the help of action potentials (voltage spikes) which are caused by cell's membrane voltage rising above the spike threshold causing a cascade of Sodium ions to flood into the neuron, propagating the action potential signal down its axon and to other neurons. The cell then closes Sodium channels and opens Potassium channels in order to return to its original state.
    - Neuron signalling is an electrochemical process and electrophysiology aims to record all meaningful characteristics that describe this process.
2. Neuron types
    - It is no secret that brain contains many different kinds of neurons. Neuroscientists have not decided on exactly how many neuron types there are and the debate has been ongoing for over a hunder years. Nevertheless, there are resources on the Internet that attempt to offer a classification for neuron types. NeuroElectro utilizes enhanced NeuroLex neuron classifications, eventually NeuroElectro may be offering its own neuron type hierarchy as we gather more data. For the purposes of our visualization, we distinguish 2 levels in the neuron type classification hierarchy - all neurons are assigned a brain region and a neuron type within that region. Each neuron type is assigned exactly one brain region (Neuron types that are present in many places or if their location is unknown comprise the "Other Region" brain region). These assignments were performed by a Dr. S.J. Tripathy - a neuroscience postdoc and the original developer of NeuroElectro.
3. Experimental conditions (metadata)
    - This data type stores information about the electrophysiological experiment itself, such as: species, strain, age and weight of the animal used, electrode type with which the measurements were taken, chemical solutions used to keep the brain slice moist and semi-alive, recording temperature, etc.
    - This data is important in order to compare ephys measurements from different experiments and labs.

NeuroElectro also stores data about each article: title, publication year, authors, etc.

Domain-specific tasks

1. Explore relationships between neuron types, ephys properties, and experimental conditions.
    (a) Find the neuron type, ephys measurement and metadata of interest.
    (b) View specific electrophysiology measurement for a specific neuron type (e.g. view rheobase values for Hippocampal CA1 pyramidal cells).
    (c) Compare ephys measurements across neuron types (e.g. resting membrane potential across all or a selected set of neuron types).
    (d) Explore the effect of metadata on an ephys measurement (e.g. action potential amplitude change with animal age).
2. Identify how many data points exist for different combinations of neuron types, ephys properties, and experimental conditions.
3. Find out how many articles support a specific analysis.
4. Summarize the data in the current analysis scope.
5. Look up details for individual evidence lines extracted from articles.

Abstract data and tasks

Abstract data Our dataset is a table where each item (row) is data for particular neuron type taken from a single article. That is, each item is uniquely identified by the neuron type and article ID attributes. The attributes of the data include 18 experimental condition indicators, 36 ephys properties, and ten that provide additional information about the article, neuron type, and organism. The current dataset has 947 rows.

At the advice of our stakeholder upon seeing the prototype, we have limited our application to analyses on the most interesting metadata properties. These attributes in detail are:

Experimental Metadata Attributes

Organism:

1. Species (categorical)
2. Strain (categorical)
3. Weight (quantitative continuous, grams)
4. Age (quantitative discrete, days)

Other:

1. Electrode Type (categorical)
2. Prep Type (categorical)
3. Jxn Potential (binary categorical)
4. Recording Temperature (quantitative continuous, degrees C)

Article Metadata Attributes

1. PubMed ID (quantitative discrete)
2. Title (categorical)
3. Year (quantitative discrete)
4. Author (categorical)

Neuron Type Attributes

1. Brain Region (categorical)
2. Neuron Name (catgorical)

Additionally, we were advised to limit our application to the 10 most interesting ephys properties (according to our stakeholder). We were supplied with an additional data table containing a units column and a column indicating whether the property's axis should be log10 transformed when plotted. All of these attributes are quantitative continuous. They are:

1. Input resistance (MΩ) - log10 axis
2. Resting membrane potential (mV)
3. Spike threshold (mV)
4. Spike amplitude (mV)
5. Spike half-width (ms) - log10 axis
6. Membrane time constant (ms) - log10 axis
7. AHP amplitude (mV)
8. Cell capacitance (pF) - log10 axis
9. Rheobase (pA) - log10 axis
10. Maximum Firing Rate (Hz) - log10 axis

Abstract tasks The primary usage context for this tool is discovery. The purpose of the tool is to help Neuroscientists achieve new insights and develop new hypotheses. This tool is not concerned with any produce tasks at this time.

Under the umbrella of discovery, a number of mid-level tasks can be identified:

1. Explore relationships between data attributes.

(a) Browse data available for analysis
(b) Identify distribution of counts across values of categorical attributes.
(c) Identify distribution of counts across values at the intersectiong of two categorical attributes.
(d) Identify distribution of a quantitivate attribute across different values of a categorical attributes.
(e) Identify correlations between quantitative attributes.

2. Explore how much data exists for different combinations of data types.
3. Narrow scope of analysis
4. Explore how much data exists for different combinations of data types within a narrowed scope of analysis
5. Lookup details for individual data items.

Solution

NED Vis uses linked views and multiple panels to enable the seamless exploration of NeuroElectro's database by the target user, which is mostly neuroscience students, postdocs, research associates and professors. To cater to our target audience's needs we provide an overview of the data and allow its filtering as well as exploration, all performed within the app.

High-Level Design Idioms

In this section, we describe the idioms we used to support the tasks above.

We faceted the display into two panels - Overview and Explore. One panel may be viewed at a time, and the user may switch between panels using a tab selector. The content of both panels will be linked via a common filter menu.

Filter Menu

The filter menu partitions attributes available for for filtering into three groups: Neuron Type, Organism, and Ephy Property. Each groups of filters was placed in a collapse panel to grant the user flexibility in terms of how screen real estate is used. THe filter menu supports abstract tasks 1a and 3.

The Neuron Type panel provides a nested treeview with tickbox selectors for neuron types and brain regions. Neuron types are grouped by brain region for ease of navigation. An "All" node is provided to make it easy for the user to select and deselect all nodes at once. A search bar is provided at the top of the tree view for ease of lookup.

The Organism panel provides a nested treeview with tickbox selectors for species and two range sliders for each of Age and Weight. The scale of the Age slider was adjusted to a log 2 scale in order to better use the space alocated to the slider (as most of the ages oare in the lower range).

The ephys panels provides range sliders for the ten ephys properties. These have been partitioned into two panels alphabetically to reduce use of screen space. encode attributes available for analysis in the form of a list (supporting task 1a). This list will contain, at the very least, neuron type, each ephys property, and each experimental condition. Each of these list items may be expanded to reveal possible attribute values - either in the form of a list (for categorical attributes) or range (continuous quantitative attributes). The user may

then filter the data in the scope of analysis by either selecting/deselecting values to include or by specifying a range. All three views will dynamically apply the any filters selected here. (Supports tasks 3 and 4, in conjunction with the other views.)

Overview Panel

This panel facets the data into a hive plot contain a hive plot (or alternative encoding) with three axes - neuron type, ephys property, and experimental condition. The coordinates along the ephys property and experimental condition axes are attributes, and the coordinates along the neuron type axis are attribute values (one for each type of neuron). A link between axis coordinates means that data exists for that pair. These links will be size coded according to the number of items that exist for that pair. When the analysis scope is the entire dataset, this will allow users to see which combinations have been considered in previous studies at a glance. When the analysis scope is a subset of the data, it will allow users to see the degree to which particular combinations are represented in the current scope. (Supports tasks 3 - 5).

The Explore Panel

The explore panel facets the data into four side-by-side plots. The user is able to select which attributes to plot on the x and y axes from dropdowns above each of the plots.

Rather than having the user select a plot type, we simplify the user experience by automatically selecting a plot that best suits to the data. When both attributes are quantitative, a scatterplot is created where each point represents a single data item. When one is quantitative and the other is categorical, a stripplot is created. When both attributes are categorical, a frequency matrix is created where each count represents a single data item that has those values. This eliminates the point occlusion problem that frequently occurs when two categorical variables are plotted against each other on a stripplot. Finally, there is a blank value to the y-axis selector so that the user may select an attribute for the x-axis only. This yields a histogram of x-axis attribute values. Together, these support Abstract Task 1b - 1e.

We have kept the default behavior of R plots, which is to scale the axes according to the range of values plotted. This means that the axes ranges may change as filters are adjusted.

Additionally, the user is able to interact with the data points on the scatterplots and stripplots. Hovering over a point displays a tooltip with the x and y values and the PubMed ID and Title of the article the item is from (supporting Abstract Task 5). Clicking on a point highlights that point in all stripplot and scatterplots on the page. Multiple points may be highlighted at a given time. Highlighted points can then be removed by clicking "Remove Highlighted" button. This allows the user to remove outliers from the plot without using the filter menu. Ultimately, we would like to allow the user to highlight points on brush rather than click, but the highlight on brush feature for R Shiny is not … Similarly, we ran into problems with highlight on hover …

No significant new algorithm or data structure developments have been performed - we used existing idioms, tools and libraries adjusting them to our needs and ensuring they perform well together. The bulk of the data wrangling, loading, plot generation and interaction work is carried out in the back-end part of our app (server.R, global.R, hive.R), while the front-end is defined by ui.R (layout, css, embedded javascript, tooltips, some interactivity).

What-Why-How: Recap

1. System: NeuroElectro Data's Visualization
2. What: Data
    - Categorical brain regions and neuron types data
    - Quantitative sequential electrophysiology data
    - Mixed experimental setup data, some of it is categorial and some quantitative sequential
3. What: Derived
    - Node degree, edge weight (number of articles that contain information for both nodes)
    - Log transform specific electrophysiology and metadata variables to better fit target user's needs

4. Why: Tasks
    - locate, identify, compare, summarize, navigate, filter
    - distribution, trend, similarity, correlation
5. How: Encode
    - Scatterplot, heatmap, re-orderable and searchable table, hive plot, histogram, strip plot
6. How: Facet
    - Partition; juxtapose; multiform, overview/detail; linked highlighting (explore tab)
7. How: Manipulate
    - Select; Navigate: attribute reduction: cut
8. How: Reduce
    - Filter: items, attributes

Implementation

Shiny app (general)

Our solution was built using the Shiny web application framework [27] for the R language [28]. Shiny server and UI components handle all transactions between the front and back end of our application. Each major component took advantage of a number of existing libraries, which is explained in more detail in the following subsections.

The first step of our application performs data loading, cleaning, and wrangling. It takes a csv dump of Neuroelectro's database as input. We load, clean, and manipulate data using base R and dplyr [29] functions. At this point we also generate and cache or load the cached version of a modified dataset to speed up matrix view generation as the filters are changed. Once the data is loaded and prepped, it is passed to the UI and server components that house the core functionality of our app.

Navigation/filtering panel

The filtering panel uses collapse panels from the shinyBS package [30]. The Neuron Type and Organism trees use the shinyTree package [31]. The shinyTree source code was modified to improve appearance and introduce text wrap to long labels that encrouched on the space of other components. The shinyjs and V8 packages [32,33] were used to add JavaScript commands on startup to modify the shinyTree component's unruly behaviour. The filter options for continuous features use slider bars from the base Shiny framework. As with the shinyTree component, they were not perfectly suited for our needs. We modified the sliders to use log scales via JavaScript commands called via shinyjs and V8. We implemented how the filter states were applied to the data set and observers to update the plots only when the selected data had been changed. Default behaviour resulted in all plots being redrawn whenever a filter element was touched, even if its value was not changed (e.g. expanding a node on a filter tree or moving a slider without deselecting any data points).

Explore panel

The four plots of the explore panel share data that is filtered based on the state of the filter panel. The data displayed on each plot is determined by the two axis selectors above each plot. The axis selectors are standard Shiny UI components that did not require modification. The plots in the explore panel are generated dynamically depending on the type of data that the user selects to view. We used the ggvis package [34] to generate the plots in the explore panel as they promised easy interactivity. Our dataset changed based on

filtering rules and axes selected and was passed to a function we implemented to determine what type of plot to show and to reduce repeated code. This led to problems, as interactive ggvis features, such as hover and brush handlers, do not work with well or at all with dynamic datasets and inside functions. While we considered other plotting options, we have implemented an on click handler that highlights points in all plots in which they appear. We also implemented action buttons to clear highlighting, remove highlighted points from all plots, and restore removed points.

Overview panel

The overview panel has three distinct components: The hive plot, the heat map, and the table. The hive plot uses the HiveR package[35] as well as the RColorBrewer package [36] for colour selection. It also uses modified functions designed by an R blogger [37] for the hive plot data wrangling. The hardest part was to optimize hive plot data gathering (counting number of co-occurrences for each pair of ephys, brain regions and metadata). Initially, with a naive implementation it took about two minutes to generate and refresh the hive plot. Through a much more effective algorithm and built-in R co-occurrences counting functions, the two minutes have been cut down to a couple of seconds. Nodes have been positioned identical distances apart on the axes for readability. Potentially, node distance could encode some information in the future. Node colour represents its degree and edge colour correlates with the number of articles supporting that edge. At first, we had encoded a few network properties into node position, size, edge width (e.g. node centrality, reachability, edge weight), however, we realized that this information is not valuable and serves only to make the hive plot confusing.

The heatmap uses a derived dataset that contains whether or not each datum contains information regarding features of particular interest to the stakeholder. It computes an association matrix on demand which is passed to the pheatmap package [38] to display the associations and annotations. Like the hive plot, it uses the RColorBrewer package for its colour pallette.

The table is generated based on filter rules and is made entirely using the DT package [39] with some non-default parameters. As its title implies, it is simply a wrapper for the DataTables JavaScript library.

Heatmap, data table, and hive plot respond to the filter panel. However, filtering and sorting done in the data table do not get reflected in the other views at this time.

Results

Use Case Scenarios for Abstract Tasks

Abstract Task: Browse data available for analysis Specific Use-Case 1: Browse attributes available for plotting and filtering.

1. User navigates to “Overview” tab
2. User clicks on x-axis selector drop-down for any of the four plots and scrolls through options.
3. User repeats for y-axis selector and discovers that the lists are the same.
4. User expands “Neuron Type” collapsible panel in filter menu and sees a list of selected brain regions.
5. User expands a tree view node for a brain region of interest to see what neuron types are listed under that region.
6. User expands the “Organism” collapsible panel to see what filters are available.

Abstract Task: Identify distribution of counts across values of categorical attributes. Specific Use-Case 2: User identifies distribution of values for AnimalSpecies.

1. User navigates to “Overview” tab
2. User selects AnimalSpecies using x-axis selector for any of the four plots.

3. User selects — using the y-axis selector for the same plot, indicating no selection.
4. User sees a histogram of values for animal species.

Abstract Task: Identify distribution of counts across values at the intersectiong of two categorical attributes. Specific Use-Case 3: User identifies distribution of counts for each combination of BrainRegion and AnimalSpecies.

1. User navigates to "Overview" tab
2. User selects AnimalSpecies using x-axis selector for any of the four plots.
3. User selects BrainRegion using the y-axis selector for the same plot.
4. User sees a frequency matrix where each cell represents to number of data points at that intersection.

Abstract Task: Identify distribution of a quantitivate attribute across different values of a categorical attributes. Specific Use-Case 4: User identifies distribution of spike amplitude property for animal species.

1. User navigates to "Overview" tab
2. User selects AnimalSpecies using x-axis selector for any of the four plots.
3. User selects spike.amplitude using the y-axis selector for the same plot.
4. User sees a stripplot of spike amplitude values for each species.

Abstract Task: Identify correlations between quantitative attributes. Specific Use-Case 5: User identifies correlation between AnimalAge and spike.amplitude.

1. User navigates to "Overview" tab
2. User selects AnimalAge using x-axis selector for any of the four plots.
3. User selects spike.amplitude using the y-axis selector for the same plot.
4. User sees a stripplot of spike.amplitude values for each species.

Abstract Task: Explore how much data exists for different combinations of data types.

Specific Use-Case 6: Explore how much data exists for pairwise combinations of neuron types, ephys properties, and metadata.

1. User navigates to Overview tab.
2. User expands each of the collapse panels and manually clears all filters.
3. User inspects hive-plot and sees that there is data for every combination of brain region, ephys property, and metadata property.
4. User inspects hive-plot and sees that the brain region to epy property connections are most sparse.
5. User inspects the numerical values on heatmap and sees:

- Each combination of ephys property and metadata property are well represented (>10 entries),
- Each combination of brain region and metadata is present but not necessrily well represented
- Not every combination of brain region and ephys property is present

Shortcoming: User must manually remove all filters as there is currently no reset button. This feature will be added in the next iteration of development.

Specific Use-Case 7: Identify ephys properties for which there is no data for the Pallidum brain region.

1. User completes steps 1 and 2 from above.
2. User finds the row in the heatmap for Pallidum.

3. User locates cells that intersect with ephys property columns.
4. User sees that there is no data for maximum firing rate.

Abstract Task: Narrow scope of analysis

Specific Use-Case 8: Limit scope of analysis to Cerebellum data from Mice

1. User conducts steps of Specific Use-Case 2
2. User expands “Neuron Type” collapse panel in filter menu
3. User Untick “All” then ticks “Cerebellum”

- The plots are redrawn using only data points from the Cerebellum

3. User expands “Organism”” collapse panel in filter menu
4. User unticks “All” then ticks “Mice”

- The plots are redrawns using only data from Mice Cerebella; number of bars on histogram is reduced to 1

Specific Use-Case 9: Limit scope of analysis to data where input resistance and spike magnitude are in a specific range

1. User conducts steps of Specific Use-Case 5
2. User expands “Neuron Type” collapse panel in filter menu
3. User Untick “All” then ticks “Cerebellum”

- The plots are redrawn using only data points from the Cerebellum

3. User expands “Organism”” collapse panel in filter menu
4. User unticks “All” then ticks “Mice”

- The plots are redrawns using only data from Mice Cerebella; number of bars on histogram is reduced to 1

Abstract Task: Explore how much data exists for different combinations of data types within a narrowed scope of analysis

Specific Use-Case 10: Explore how much data exists for different combinations of data types when dataset is limited to the Cerebellum and Mice

1. User conducts the steps in Specific Use-Case 8
2. User navigates to Overview tab
3. User examines hiveplot and sees that there is data for only one ephys property - input.resistance

Shortcoming: No way to get the number directly.

Abstract Task: Lookup details for individual data points.

Specific Use-Case 11: User looks up details for a data point on scatterplot.

1. User conducts steps form Specific Use-Case 5
2. User hovers over a point of interest in a scatterplot.

3. User sees a tootip, which displays the PubMed ID and title. User takes note of title.
4. User navigates to the Overview panel.
5. User scrolls to the Table view below the hiveplot and heatmap view.
6. User begins to type the title in the Title column. Results filter automatically.
7. User locates row corresponding to data point of interest.

Shortcoming: User must remember or write down the title as copy-paste functionality is not available from the tooltip. Ideally, the details would be available on click.

Evaluation by Stakeholder

Our stakeholder has evaluated the result of our project and has given the f

The thing he likes most about the app is that it very cleanly allows the user to visualize only the data that they're interested in, be it specific brain regions, neuron types, or data matching specific experimental conditions, using the organism/metadata filter tab. This was one of his key requirements, and he think it is has been implemented very well. In the short time of playing with the application, he has found new things that he was not able to fully apprecaite before. For instance, he had not appreciated how correlated the ephys properties rheobase and input resistance are.

He quite likes the heatmap on the overview page because it clearly indicates how much data is at the intersection of each pair of attributes. But he expressed concern that the hiveplot and heatmap express much of the same thing, and are therefore not the best use of screen real estate. He prefers the heatmap because it is harder to see specific data using the hive plot because all nodes are more or less connected to eachother.

He requested that the heatmap and hive plot colormap be log transformed because it is hard to tell the difference between features pairs with no data and just a small amount of data.

He said that detail lookup is good but could be improved by linking data points in the Explore tab to the underyling data source directly.

Overall, he feels the application is particularly "feature rich" at this point and I worry that the naive user would be overwhelmed with the overall smorgasbord of data.

Discussion and Future Work

Implementation Approach

Originally, we planned to use javascript with a few libraries (D3.js [40], angular.js [41]) for our project, however, the team's familiarity with R and the desire to learn Shiny [27] played the decisive role in our framework choice. Given the chance to choose the framework again, we would have chosen Shiny due to its ability to handle large amounts of data and generate complex plots with relatively high speed. D3 has better integration with the front-end, but Javascript and CSS injections in Shiny serve a similar purpose, even if less elegantly.

We have learned that Shiny provides a solid framework for front-end web development and R data analysis visualizations. The dynamic integration of data frames and vectors in R with the rendered Shiny plots is quite impressive. Unfortunately, customizing the default behaviour becomes a choire because of the modular nature of Shiny. One could say that getting something to work in Shiny is fast and simple, but getting the exact feature to work (if it is not the default behaviour) can be very difficult and time consuming.

Data wrangling speed is a usual concern when dealing with large volumes of data that are transformed into plots. We ran into this problem with the overview panel - both the hive plot and the heatmap took a very long time to generate initially and we had to optimize their data gathering, parameter calculating and plotting features in order to achieve reasonable online app refresh speeds. The hive plot generation is already

on an acceptable time scale, but the overview heatmap still takes too long - its generation pipeline can be optimized further if data wrangling is united with the hive plot's.

As noted above, we ran into major problems using the ggvis package. The decision to use this package was made based on the considerable buzz surrounding its ease of use and rich interactive features. These features, particularly brush and hover handlers, are incompatible with functions and dynamic data sets. While there are work arounds to make these features work to some degree with dynamic data sets, we were unable to find a way to make them worked when called via a funciton. Our function to determine the nature of the data to plot and remove missing values (something that ggvis also does not do, unlike it's ancestor ggplot) is used by all four plots in the explore panel and is about 120 lines long. It would be possible for us to move this funcitonality out of our plot generating function, but it would result in 480 lines of duplicated, spaghetti code, which would make future work on the project extremely difficult (and according to one of the authors, it would 'hurt his soul'). As we aspire to follow the DRY principles [42] in our work, we will explore other plotting libraries going forward that support clean code.

In the near future we would like to add more interactivity to the hive plot: each point and edge should provide a detailed tooltip upon mouseover or click, brush selection should allow to filter the data similar to the filter panel (and brush selection in the explore view). The explore view brush selection currently does not work - the user has to click each point that they want to remove individually (or use the filtering panel). The scatter plots in the explore panel still have the colour channel available, we could make use of that by allowing multi-variable plotting or splitting up one of the variables into groups.

Analytical and Semantic Shortcoming

One limitation of the current system is that it is limited to a select subset of ephys and metadata properties. This decision was made in conjunction with the stakeholder in order to improve user experience. Paring down the properties list made it so that the number of entries in the axis selector dropdowns was reasonable and the hiveplot less cluttery.

However, there could be value in exploring less common ephys properties and metadata measurements. In fact, one of the original motivations for the Overview panel was to discover attributes for which little data exists, i.e. rare attributes. With these removed, the utility of the Overview panel is less obvious. Ideally, there would be a way for the user to access this data without increasing the clutter in the system, perhaps by having an advanced options menu.

A potentially misleading aspect of the system is how it handles NA values in conjunction with filters. There are many missing values in our data, and if we removed every row with a missing value, we would have very little data to work with. However, when a user applies a filter, the assumption is that they want to look at values that are within that range for certain. Currently, we include data points with null values unless a filter on that value is explicitly applied. For instance, data points for which AnimalWeight is NA are plotted unless the user applies a filter using the range slider for Weight. Because there are so many NA values for AnimalWeight, the user need only move the slider a little bit and many points are removed fromt he plot. This creates the illusion that the removed points fell outside of the specified range.

Another problematic aspect of the filter menu is that the user can collapse a panel while a filter is applied and then forget about it, producing an "out of sight out of mind" effect. Ideally, we would have a non-intrusive way to encode the fact that a filter has been applied on a given panel, perhaps by altering the color.

The Overview tab is busy and difficult to make sense of unless much of the data is filtered out. One problem is that the heatmap contains a number of redundant cells (the upper triangle) as well as intersection within data type (Brain Region by Brain Region). We could increase the signal-to-noise ratio by plotting Ephys Properties and Metadata on one axis and Brain Region and Metadata on the other, which would eliminate much of the uninteresting and redundant data. However, we will need to consult with our stakeholder to confirm that within-type intersections are not of interest. Another feature that could make the heatmap more approachable would be a row/column for the sums of values across columns and rows.

As it stands, the system only supports hypothesis generation, not hypothesis testing. The analystical power of the system could be improved considerably by integrating some basic statistical functionality, such as means, ranges, correlation coefficients, and p-values. Doing so would allow the system to benefit most fully from the strengths of the R framework.

User Experience and Style Shortcomings

There should be an easier way for users to get data from specific points on demand. Currently, the tooltip gives the title and PubMed ID for an article, but if the user wants more information of the article, they must note the title, switch to the OVerview tab, and enter it in the search bar. This information should be available on demand without requiring a tab switch.

There are a number of improvements that could be made to the filter menu. First, there should be a reset button so that the user can remove all filters in one click. Second, there should be a scrollbar for the filters menu that appears when its contents extend below the plots. This is not trivial to do in R-Shiny panels and will require direct manipulation of underlying Javascript objects.

Other stylistic improvements that could be made include:

1. More consistent styling of axis labels and dropdown menu items
2. Cleaner, shorter axis labels with smarter spacing
3. Better placement of data type legend for heatmap (not trivial to do with pheatmap library)

Filters Menu First, there should be a reset button for the filters menu so tha the user does not have to manually remove all filters. This should be easy to add in the next iteration.

Ideally, the filters menu to would have scrollbars when its content extends below the plots.

Conclusions

We have successfully developed a newer and better visualization tool for the NeuroElectro website. Shiny framework ensures NeuroElectroVisuals's integratability with the NeuroElectro database and makes it capable of performing the main tasks defined by its original developer - Dr. S.J. Tripathy (outlined above in the data and tasks abstraction section). The new visualization provides an overview of the database with multiple plots and table as well as a more finely targeted exploration panel that allows neuroscientists to investigate the brain regions and electrophysiology properties of interest. The app also has a filtering panel that is integrated with both overview and explore panels and allowes subsetting the database by brain region, ephys values and experimental metadata. This functionality enables users to analyze the relevant data.

Our visualization can and will be improved to better serve field specialists. During the first few months of the beta deployment period we plan to gather usage data and feedback. With these we plan to ensure that we have covered every use case scenario and that the tool is running smoothly on various platforms. We are also hoping to get data about stress-resistance (online concurrent usage of the visualization app). If NeuroElectro becomes wildly successful and we have an absolute need for a more customized visialization, the possibility exists to re-write the existing code in javascript. Alternatively, more efficient Shiny implementation is also possible as not all the algorithms and functions have been optimized in the current implementation due to time restrictions.

References

[1]S. J. Tripathy, J. Savitskaya, S. D. Burton, N. N. Urban, and R. C. Gerkin, "NeuroElectro: A window to the world's neuron electrophysiology data," *Frontiers in neuroinformatics*, vol. 8, 2014.

[2]L. French and P. Pavlidis, "Informatics in neuroscience," *Briefings in bioinformatics*, vol. 8, no. 6, pp. 446–456, 2007.

[3]D. Rebholz-Schuhmann, A. Oellrich, and R. Hoehndorf, "Text-mining solutions for biomedical research: Enabling integrative biology," *Nature Reviews Genetics*, vol. 13, no. 12, pp. 829–839, 2012.

[4]N. J. Van Eck and L. Waltman, "Text mining and visualization using vOSviewer," *arXiv preprint arXiv:1109.2058*, 2011.

[5]U. Hinrichs, B. Alex, J. Clifford, A. Watson, A. Quigley, E. Klein, and C. M. Coates, "Trading consequences: A case study of combining text mining and visualization to facilitate document exploration," *Digital Scholarship in the Humanities*, p. fqv046, 2015.

[6]Tableau, *Tableau*. 2015.

[7]SAP, *SAP businessObjects analysis, edition for oLAP 4.0*. 2015.

[8]Microsoft, *Excel 2016*. 2016.

[9]M. Baitaluk, M. Sedova, A. Ray, and A. Gupta, "BiologicalNetworks: Visualization and analysis tool for systems biology," *Nucleic acids research*, vol. 34, no. suppl 2, pp. W466–W471, 2006.

[10]N. Henry and J.-D. Fekete, "MatrixExplorer: A dual-representation system to explore social networks," *Visualization and Computer Graphics, IEEE Transactions*, vol. 12, no. 5, pp. 677–684, 2006.

[11]S. Gaston, *Arc diagrams in r: Network visualizations*. 2013.

[12]K.-K. Yan, G. Fang, N. Bhardwaj, R. P. Alexander, and M. Gerstein, "Comparing genomes to computer operating systems in terms of the topology and evolution of their regulatory control networks," *Proceedings of the National Academy of Sciences*, vol. 107, no. 20, pp. 9186–9191, 2010.

[13]M. Krzywinski, *Hive plot website*. 2013.

[14]M. Krzywinski, I. Birol, S. J. Jones, and M. A. Marra, "Hive plots—rational approach to visualizing networks," *Briefings in bioinformatics*, vol. 13, no. 5, pp. 627–644, 2012.

[15]R. Bearavolu, K. Lakkaraju, W. Yurcik, and H. Raje, "A visualization tool for situational awareness of tactical and strategic security events on large and complex computer networks," in *Military communications conference, 2003. mILCOM'03. 2003 iEEE*, 2003, vol. 2, pp. 850–855.

[16]M. Dumas, J.-M. Robert, and M. J. McGuffin, "Alertwheel: Radial bipartite graph visualization applied to intrusion detection system alerts," *Network, IEEE*, vol. 26, no. 6, pp. 12–18, 2012.

[17]A. Sopan, A. S.-I. Noh, S. Karol, P. Rosenfeld, G. Lee, and B. Shneiderman, "Community health map: A geospatial and multivariate data visualization tool for public health datasets," *Government Information Quarterly*, vol. 29, no. 2, pp. 223–234, 2012.

[18]N. Turner, *Why filter bars are better than left-hand filters*. 2015.

[19]G. Nudelman, *Best practices for designing faceted search filters*. 2009.

[20]M. Friendly and D. Denis, "The early origins and development of the scatterplot," *Journal of the History of the Behavioral Sciences*, vol. 41, no. 2, pp. 103–130, 2005.

[21]S. Haroz, R. Kosara, and S. Franconeri, "The connected scatterplot for presenting paired time series," 2015.

[22]S. Few, *Coordinated highlighting in context*. 2010.

[23]W. Chang, *Linked brushing in ggvis*. 2014.

[24]M. B. A. Hanson, "Package 'hiveR'," 2012.

[25]T. Munzner, *Visualization analysis and design*. CRC Press, 2014.

[26]V. Setlur and M. C. Stone, "A linguistic approach to categorical color assignment for data visualization," *Visualization and Computer Graphics, IEEE Transactions on*, vol. 22, no. 1, pp. 698–707, 2016.

[27]W. Chang, J. Cheng, J. Allaire, Y. Xie, and J. McPherson, *Shiny: Web application framework for r*. 2015.

[28]R Core Team, *R: A language and environment for statistical computing*. Vienna, Austria: R Foundation for Statistical Computing, 2015.

[29]H. Wickham and R. Francois, *Dplyr: A grammar of data manipulation*. 2015.

[30]E. Bailey, *ShinyBS: Twitter bootstrap components for shiny*. 2015.

[31]Trestle Technology, LLC, *ShinyTree: JsTree bindings for shiny*. 2015.

[32]D. Attali, *Shinyjs: Perform common javaScript operations in shiny apps using plain r code*. 2015.

[33]J. Ooms, *V8: Embedded javaScript engine*. 2015.

[34]W. Chang and H. Wickham, *Ggvis: Interactive grammar of graphics*. 2015.

[35]B. A. Hanson, *HiveR: 2D and 3D hive plots for r*. 2015.

[36]E. Neuwirth, *RColorBrewer: ColorBrewer palettes*. 2014.

[37]V. M, *Network visualization – part 4: 3D networks*. 2013.

[38]R. Kolde, *Pheatmap: Pretty heatmaps*. 2015.

[39]Y. Xie, *DT: A wrapper of the javaScript library 'dataTables'*. 2015.

[40]M. Bostock, V. Ogievetsky, and J. Heer, "D3: Data-driven documents," *Visualization and Computer Graphics, IEEE Transaction*, 2011.

[41]P. Darwin and P. Kozlowski, *AngularJS web application development*. Packt, 2013.

[42]B. Venners, "Orthogonality and dRY principles." Artima Developer website, http://www. artima. com/index. jsp, March, 2003.