Avant-Garde: Data Visualization Tool for HIV Epidemiology

Yu Xia
Advisor: Dr. Nadir Weibel
Computer Science and Engineering Department
University of California, San Diego
9500 Gilman Dr, La Jolla, CA 92093 y2xia@eng.ucsd.edu

ABSTRACT
The transmission of the HIV often involves complex social and sexual interactions among people. In border area cities such as San Diego and Tijuana, medical researchers try to understand and analyze the transmission networks of HIV to identify affected or at-risk populations so that they can either concentrate treatment or prevent further infections from happening. However, the multi-dimensional patient data set is hard to understand by looking at its raw form and is even harder to reason about the association between different socio-demographic and phylogenetic factors.

To tackle these challenges that researchers are facing when try to analyze the multidimensional data sets, we built a web-based data visualization tool which enables exploratory data analysis and "what-if" investigations. In particular, researchers can flexibly access data and perform meaningful analysis by leveraging a set of interactive data visualization components that are included within the tool.

Author Keywords
Data visualization; HIV; Interface Design; Visualization Path; Multi-Dimensional Data

ACM Classification Keywords
H.5.2 User Interfaces: Prototyping;

INTRODUCTION

The human immunodeficiency virus (HIV) is a world-wide epidemic that has affected the lives of millions of people. According to UNAIDS 2014 report, approximately 35 million people worldwide were living with HIV in 2013 [21]. Particularly, there is an emerging epidemic on the Mexico-US border. Situated on major drug trafficking routes that bring heroin, cocaine, and methamphetamine from Mexico into the United States, Tijuana, a Mexican border city which is contiguous to San Diego, experience high rates of local drug use and rank first in prevalence of illicit drug use in the country. In addition to that, sex tourism is also featured in Tijuana and the city has certain "tolerance zones" where sex work is openly practiced and regulated by the authorities. The pervasive drug usage, needle sharing, and the lack of protection in sexual activities all contribute to the growing number of HIV-infected patients in Tijuana area. And with Mexico's border cities serving as funnels for workers and goods traversing the two countries, Tijuana's AIDS crisis poses a direct threat to the United States.

In 2012, the National Institute of Drug Abuse (NIDA) awarded the Avant-Garde of HIV/AIDS Research Award to Dr.David Smith for the purpose of building a novel system that uses various factors such as demographics, viral strains and geographic data to identify HIV transmission networks to find potential hot spots [19]. The tool covered in this paper serves as the enhanced visualization component alongside with the previous data collection and analysis platform [18]

PROJECT SCOPE

The project is primarily focused on the San Diego - Tijuana border area. Most of the data sets are coming from the previous or on-going studies conducted by the UCSD Antiviral Research Center (AVRC) and multiple medical facilities around the San Diego (California) and Tijuana (Mexico) area which contains over thousands of patient's data including the virologic, epidemiologic, clinical, geographic and socio-demographic information. There are about 50 carefully identified variables in the data set that would serve as the starting point for the researchers to conduct their initial analysis.

BACKGROUND AND MOTIVATION

The project is originated from the ongoing research of Dr. Mehta at UCSD who works on the Avant-Garde project and conducts experiments on Molecular Epidemiology of HIV at the San Diego/Tijuana border. According to [20] on HIV epidemiology in Mexico, in 1980s, most HIV cases were identified in individuals who previously lived in the US. By 1991, only 44.3% of cases were in previous U.S. residents while, in the year 2000, the rate dropped to 12%. Although we can see a trend of rate drop in terms of HIV infected patients that have previously lived in the U.S, we still know little about the impact of migration on the development of the HIV/AIDS epidemic in Mexico.

To better understand those hidden patterns is essentially the main motivation that drives this research project. We want to build an intuitive and easy to use data visualization tool to allow researchers to identify and analyze the relationship between different data dimensions and hopefully identify patterns across geographic locations and times. In [18], Sandy Law had successfully built a new workflow for HIV researchers at UC San Diego AntiViral Research Center (AVRC) that uses a web-based system with a clear architecture able to harmonize data coming from a variety of sources while maintaining the consistency and integrity of the data. Our visualizations further leverage the data provided by this

system, with the longer-term scope to eventually get incorporated into the main system.

The remainder of this paper is divided into four sections, in the Data Set section we explain and discuss key attributes and properties of the collected patients data. This has a direct impact on the design and implementation of the presented visualization tools. The following Visualization Path section talks about how we combined different visualization components to create a visualization path that facilitates and persist exploratory data analysis. After this, we outline implementation detail of the visualization tool, including the client-side and server-side structure, how we pre-process the data and the different visualization components that we have implemented. Next, in the Results and Evaluation section, we describe the current usage of the system and other more general use case scenarios together with systems advantages and limitations. We end by wrapping up the contribution and present future work.

DATA SET

The HIV patients data set we are focusing on is collected from a number of medical sites around the San Diego and Tijuana area. Despite the fact that the raw data collected from those various sites is heterogeneous, by using the current system [18], the data results harmonized and access to the data is based on consistency and integrity.

The data set contains around 50 variables, some of them are the aggregated results over other columns and others are duplicated but represented in different formats. In order to proceed with our visualizations, it is important to first understand the data sets structure. We have categorized the data variables into four major types: Unique identifier fields, Categorical fields, Numerical value fields, Date fields. The unique identifier fields contains a unique value for each data point, those fields are mainly used to identify each unique patient and it is not necessary to visualize them. We display those fields in the data table section part of our visualization tool interface so that researchers can easily identify each data point. The categorical fields are the main focus in terms of variables represented in our the visualization tools since they all have a limited number of unique values and are great candidates for clustering or categorization. Numerical fields are ideal for scatter plots and serve as categories when we visualize variable distribution. Date fields are mainly used for time-based visualization, to see how certain categorical or numerical fields change or migrate over time.

One issue with the current data set is that we don't have too much data point for patients from Mexico that is close to the San Diego - Tijuana border. The researchers can see and explore the patients geographical distribution mainly in the San Diego area but not in the Tijuana area. This have an impact on enabling researchers to understand certain critical questions such as:

- Are the San Diego and Tijuana epidemics connected?
- What is the direction of viral migration across the border?

Data Set Dependency

When we started designing our visualization tool, we wanted to make it as independent from the data source as possible. There are two main reasons behind it: the first one is that users of the tool might not know exactly how they want to visualize the data or what data variables they want to compare, so it would be best if we can present the complete original data set to the user without tailoring it. The second reason is that the data source might come from different domains: medical research, political science, social science etc, and without the corresponding domain knowledge or the specific spec the developer wont be able to implement the data formatting, aggregation, filtering in advance. So ideally the tool can be independent from the data set and delegate those actions to the users on the client side.

The approach we took is to first build a data schema and data model on the server side according to the given data set, and to then categorize the variables into the four types mentioned in the previous section. When are then able to render the visualization components, exposing all the variables to the user through two main visual components: the data table, where we present detailed information about each entry, and the interactive visualization, where users can focus on specific explorations of the data. We allow users to choose what variable theyd like to include or ignore in the data table so they wont get overwhelmed if there are too many variables in the data set. Similarly in the interactive visualization component, users can select and filter the through check boxes and drop down menus that are filled with specific variables.

VISUALIZATION PATH

In order to enhance the exploratory power of the data visualization, as well as offer a way to persist specific data explorations within our tool, we introduce the novel concept of visualization path. Essentially a visualization path includes a set of visualization components and their selected values which can be saved by the user into the database and reloaded into our system later on. As shown in Figure 1 The researchers can use different visualization components to visualize a particular part of the data performing filtering or selection and then choose a different visualization to analyze the filtered data in a path of subsequent visualizations. Manipulating filtering and selection in any of the visualizations will impact the whole visualization path and the single components will change accordingly.

In our prototype tool, we implemented four major visualization components, circle packing, parallel coordinates, scatter plot and pie chart. The researchers can first see the visualized data in circle packing where the data can be clustered for example based on virus strain cluster ID. Then he/she can click on the virus cluster 15 and select the parallel coordinates visualization as the next one in the visualization path. Now since the researcher already filtered the global data to only cluster 15, the parallel coordinate will only visualize the cluster 15s data point and enable further data brushing on them. In this simple scenario, the visualization path contains two visualization components: circle packing and parallel coordinate and one filter with cluster-id 15. The researcher can then name this path and save it for future usage or share it with

Figure 1. Visualization path includes control elements for the user to choose the next visualization component and to save the current path plus see the filter and data selection on each component

other colleagues through the system.

IMPLEMENTATION

The prototype visualization tool is a typical web application that can be divided into client-side and server-side components.

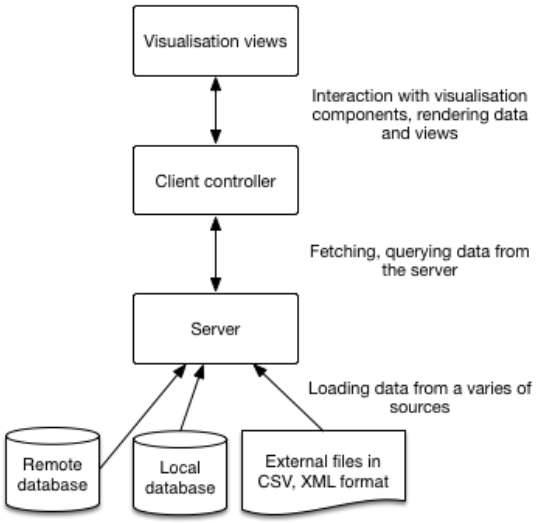

Figure 2. System architecture overview

Server-side Components

Our server-side component is mainly responsible for handling client requests, communicating with the database, executing the query and returning the formatted, parsed data back to the client-side. We used the Ruby on Rails [12] framework to build the server-side component. Ruby on Rails is known for fast prototyping, it comes with its own Object Relational Mapping library which is quite flexible to use to capture the database table structure.

Data loading

In our current implementation, the raw data are processed using a joined and flattened CSV format file, so we created a data model against it and loaded the CSV data into a dedicated MySQL server. However, our tool and architecture is general and we provide an alternative configuration that support direct communication with a remote MySQL server. By leveraging the Rails framework, we can handle a variety of data sources and formats natively. The only issue is that developers are needed at setup time to programmatically change the configuration, load the data from different sources, and create dedicated data models.

Data querying

The server-side app receives data requests from the client-side which might be based on a series of query and filtering parameters. The server-side is then responsible for running these query against the data source and retrieve the results.

Data rendering

Another major responsibility of the server-side component is to respond with the data in specific formats. Different visualization components might consume the data in different ways and based on different structures. To reduce the code complexity on the client-side, we delegate this task to the server. Ruby on Rails contains a feature rich template engine, combined with other powerful ruby gems allows us to construct the representation of the returning data with ease.

Client-side Components

The client-side implementation leverages a combination of 3rd party Javascript libraries. In order to enable rich interactivity and not sacrifice functionality, we have adopted the front-end MVC framework Angular.js [ref] to serve as the backbone of this web application. By using Angular.js, we can inject other 3rd party visualization libraries into the front-end controller and use them right away.

General visualization components

In order to implement a set of general visualization components we have adopted the D3.js [3] visualization library. In particular, we used the Angular-nvD3 library [2] which is based on an Angular.js directive wrapper around nvD3.js [10] , a reusable charts and chart components for D3.js. It provides parallel coordinate Figure 7, scatter plot Figure 11 and pie chart Figure 12 by itself with many other visualization components out of the box. We also implemented our own visualization component: circle packing and managed to merge the code into the Angular-nvD3.js code base.

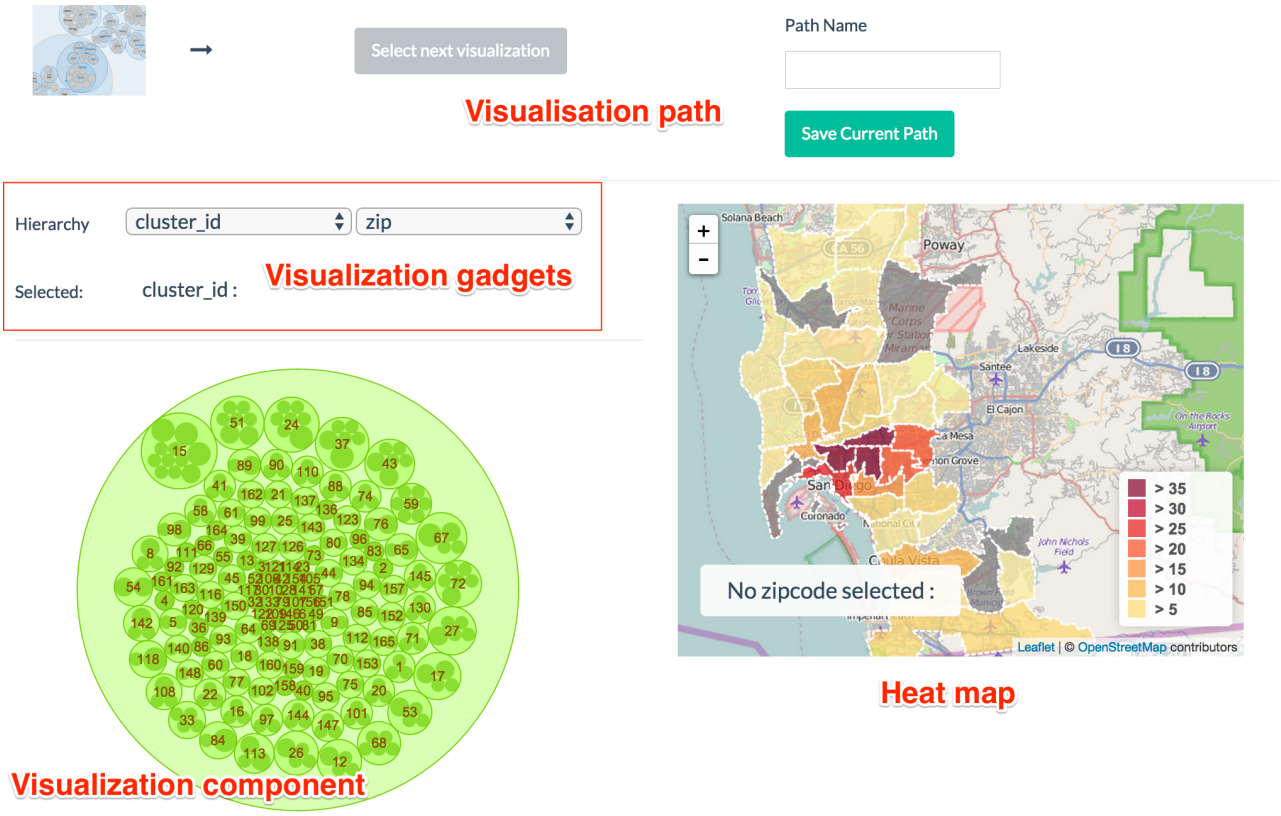

Figure 4. Visualization tool layout

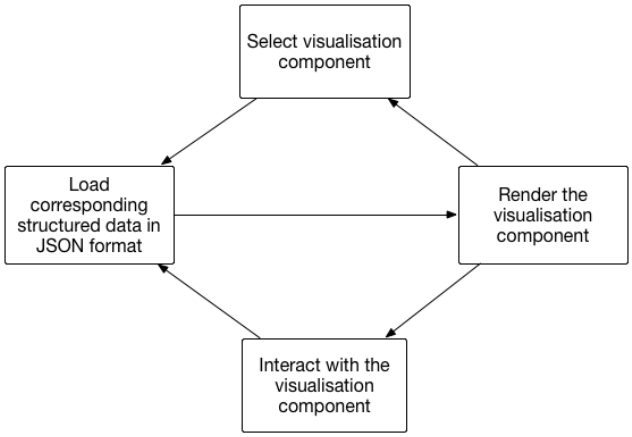

Figure 3. Client-side interaction workflow

Time-based visualization components

We have also incorporated the time-based visualization components from [15] into the client-side. These visualizations, particularly the motion chart and annotation chart, are built based on Google chart tools [6] particularly the motion chart and timeline chart. In order to use them in our client-side Angular.js environment, we have leveraged a 3rd party library [14] angular-google-chart.js for Google chart tools which enable us to seamlessly integrate Google chart tools functionality into the existing code base.

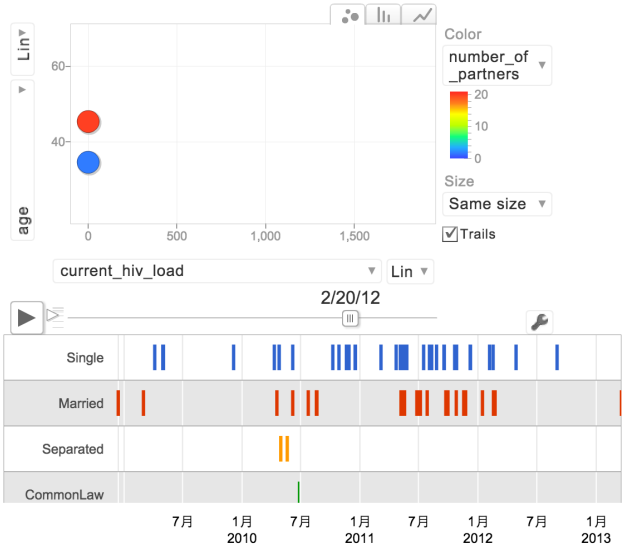

Figure 5. Motion chart & Timeline chart

Through the motion and the timeline chart, the researchers can see the development and the progression of certain data columns against time

Heat Map

Since we want to allow researchers to leverage our visualization tool to understand and explore the relationship between geographical distribution of variables and other attributes, we also implemented a heat map feature mapping variables to corresponding regions on a map based on the ZIP code attribute of each data point.. We use different colors to indicate the number of data points in each different ZIP code area.

For the map, we used leaflet.js [8] which is a Javascript library loading the OpenStreetMap [11] data. In order to work seamlessly with other Angular.js components, we have also incorporated the Leaflet directive for Angular.js [7] which will render the same global data. In this way, the heat map can also reflect actions performed on the visualization components such as selection, filtering and only render the designated data set.

To show the ZIP code area and the heat map, we need to know the geo-encoded boundaries for each ZIP code area and convert them to GeoJSON [5] format which can then be rendered on top of the leaflet.js map. For the ZIP code geo-encoded boundaries data, the data source is from the 2010's U.S. census ZIP code tabulation areas data [1] in shape format [13], then we used GDAL - Geospatial Data Abstraction Library [4] to convert shape format into GeoJSON format data.

When we finally receive from the server the data set that we need to visualize, in our example the HIV patient's data. the ZIP code information is pulled and run through a standalone parser script to retrieve all the areas included in the GeoJSON data which will be then passed to the client-side and used to render the heat map component.

Data table

When researchers start to explore or filtering the data set through different visualization components, we also want to present them with the corresponding raw data set information. In order to do that, we have used the ng-table library [9] which provides a simple table widget that includes sorting and filtering on Angular.js. Whenever the user triggered data selection or filtering on the supported visualization component, we will filter the loaded global data and present the filtered results in the data table section.

Figure 6. Data table layout

Client-server communication

In order to retrieve and visualize data, the client-side controller issues query for JSON format data to the server-side and renders the visualization component using the returned data. We leveraged Angular.jss `$resource` service to create resource objects in a structured way which can be then used to interact with the server-side data

Client-side interaction

In our project, we used Angular.js to implement the visualization tool in a single page application. The application is model to provide a more fluid user experience akin to a desktop application. For each visualization component, a set of gadgets can be enabled so that users can interact with them. The user can also manipulate the component directly. Any of these interactions will issue new data queries to the server and the returned data will become the new application level global data. Also, when users choose a different visualization path, the client- side controller will issue a new data query to the server-side for the new visualization component, since different visualization components consume data in different formats.

DATA VISUALIZATION

In our prototype tool, we have implemented four visualization components, namely circle packing, parallel coordinate, scatter plot and pie chart. In this section, we will explain these components in detail.

Parallel Coordinate

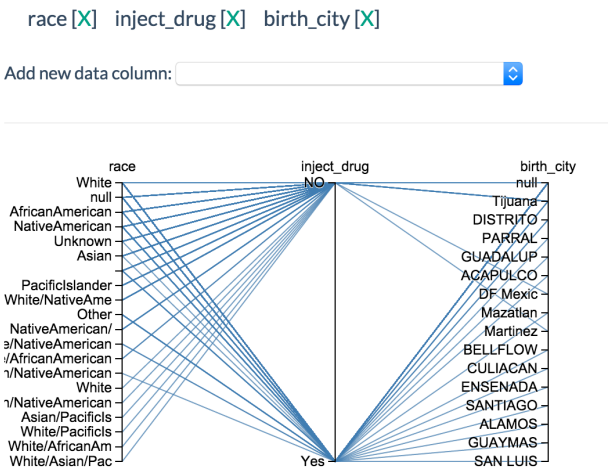

Figure 7. Parallel Coordinate

Parallel coordinates is a well-established visualization technique first proposed by [16]. It is a scalable framework in the sense that the increase of the dimensions correspond to the addition of extra axis. In Figure 7, we have visualized the patient's data set over three variables: race, drug usage, and birth city. The researchers can immediately see the correlation between these variables which is more insightful than looking at the raw data set.

In our implementation, we also support data brushing which allow the users to interact with the data by selecting the data

region for each variable. Data brushing will trigger filtering on the global data where the filtered result will be reflected on the heat map and data table section. In figure 8, we have selected the ZIP codes that are from the downtown region of San Diego, and we can see that the heap map now only show the patient's data in the downtown San Diego area.

There is, however, one significant issue with parallel coordinate visualizations: the visual clutter caused by an excessive overlaying of polylines that limit the effectiveness of parallel coordinates in visualizing a dense data set. Our current solution is that we calculate the number of data point in each column, and then use the column that have the largest number of data point to determine the parallel coordinate graphs height. This will help reducing the overlaying of polylines issue but will make the graphs height relatively large.

Circle Packing

Circle packing [22] is a method to visualize large amounts of hierarchically structured data and its design was inspired by the treemaps visualization [17] which is another method for the visualization of hierarchically structured data. In circle packing, tangent circles represent sibling nodes at the same level; to visualize the hierarchy, all children of a node are packed into that node (and thus determine its size). The size of a leaf-node can represent an arbitrary property, such as average the patient's age. An advantage of this algorithm is the good overview of large data sets and the clear representation of groupings and structural relationships.

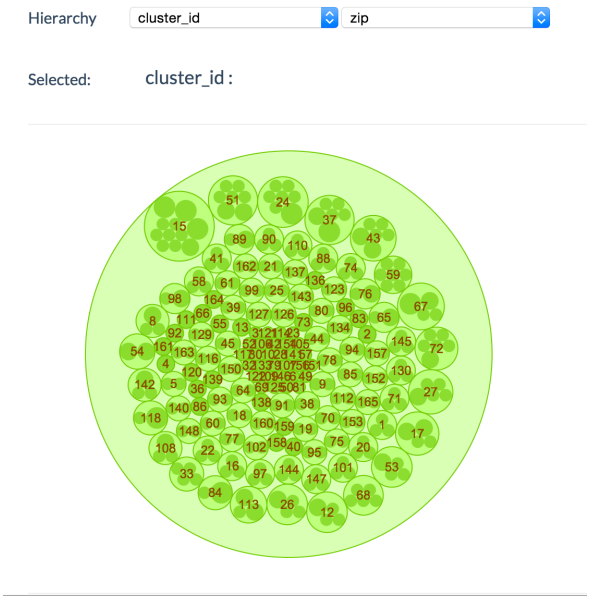

Figure 9. Circle packing

As a space-filling technique, Circle Packing visualizes information as a spatial extension. It places any number of brother nodes, represented by circles, in an arrangement that has a high density and retains a convex shape approximating a circle as a number of nodes grows. Due to this property, a set of sibling nodes can always be packed into its (also circle-shaped) parent node.

Compared to that of treemaps, a circle-packing visualization - while possibly a bit less space-efficient - is easier to visually comprehend, while it still retains the ability to (theoretically) display all hierarchies at once. For example in Figure 9, we visualized the patient's data set in two hierarchy: virus strain cluster and patient's ZIP code. As we can see the viral strain cluster 15 currently contains the largest set of patients data.

In our implementation, users can interact with the circle packing by clicking on one of the clusters which will zoom in and show the packed circles detail. At the same time, this will trigger a global data filtering, so for instance, the selected heat map and the data table will reflect the filtered data. Figure 10, shows how after selecting the virus strain cluster 15s circle, and requesting this clusters patient data, the filtered results are also reflected on the heat map to the right as well as in the data table section.

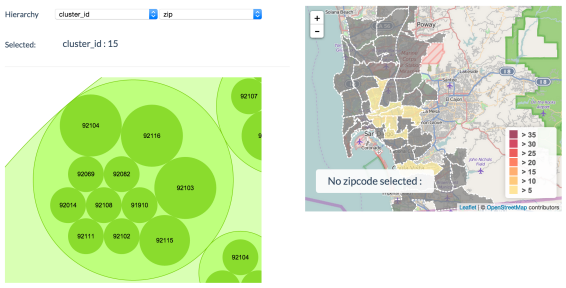

Figure 10. Circle packing

Scatter plot

Scatter plots are used to plot data points on a horizontal and a vertical axis in the attempt to show how much one variable is affected by another. Each row in the data table is represented by a marker whose position depends on its values in the columns set on the X or Y axes. Our system allows the user to group the data on arbitrary variables, which all the unique value will be used to group the data, each data groups data point will share the same color and a customizable unique shape.

In Figure11, we visualized some of the data from the patients data set as a scatter plot. The data is grouped by gender: Female, Male, or TG. Different colors are applied to those three groups data point as well. The x-axis depicts patients HIV level, the y-axis depicts patients age distribution. The size of each data point circle depicts the number of sexual partners the patient have.

Currently, the users interaction with the scatter plot is limited. In the future, we plan to implement data brushing to allow users to select data segments that interest them the most. The filtered result will again be available for example on the heat map and the data table.

Pie chart

A pie chart (or a circle graph) is a circular chart divided into sectors, illustrating relative magnitudes or frequencies. In a pie chart, the arc length of each sector (and consequently its

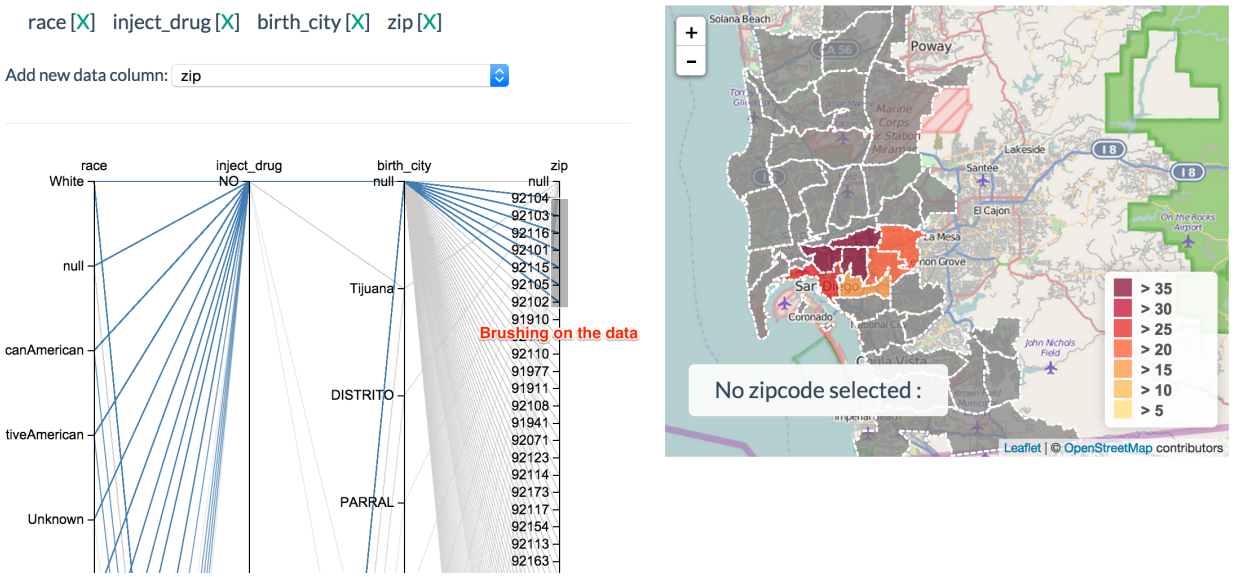

Figure 8. Brush the data to see San Diego downtown patient's distribution

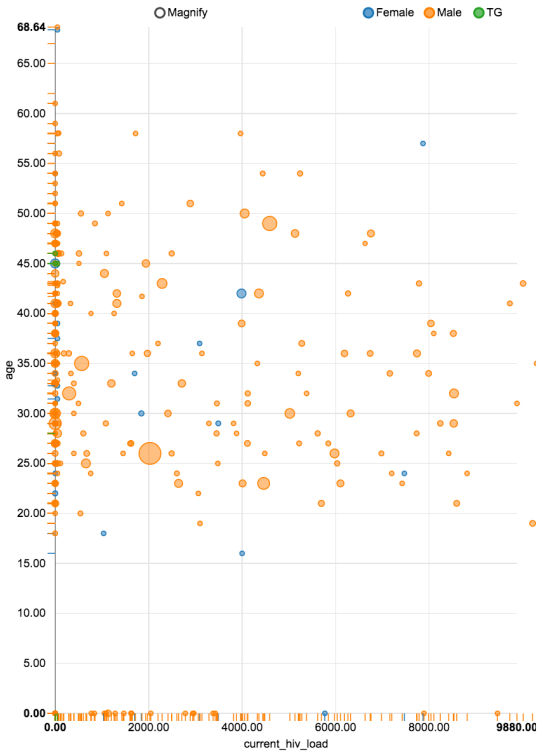

Figure 11. Scatter plot

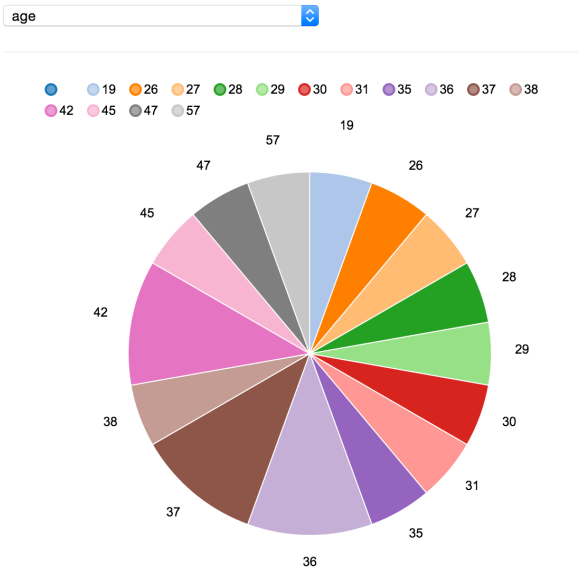

Figure 12. Pie chart

central angle and area), is proportional to the quantity it represents. Together, the sectors create a full disk.

Users can select which variable they want to visualize in the pie chart. In our example, shown in Figure 12, we visualize the HIV patients' data set using the pie chart to show the age distribution.

EVALUATIONS

In order to evaluate the prototype visualization tool, we have deployed the tool on a secure server under the AVRC and provided access to the medical researchers. We have adopted a simple agile development methodology, based on a tight design-develop-feedback loop. After we build a new prototype feature, we will share this with the researchers who test and evaluate its functionality. This typically results in specific feedback that help guiding with the next development cycle. According to their usage feedback, the prototypes interface is quite intuitive to use. Researchers generally like the clear layout that differentiate the visualization path components, and specifically the different way the system can visu-

alize the data through. From our initial evaluation, it seems that overall the visualization path is functional and can enable interesting preliminary exploratory data analysis.

At the same time, researchers have reported problems within the prototype tool. As part of each visualization component, we have provided relevant filters to allow researchers to determine what variables they want to visualize. Our current implementation loads all the variables from the database table, however, not all of them are suitable for visualize under certain visualization component. For example, in the scatter plot, the x and y axis should only be numerical fields, or categorical fields that have string value mapped to numerical value. Currently, although we have manually identified and categorized the data columns into four categories, we didnt implement it into the server-side nor the client side components. Mainly because those hard coded columns information would make the tool deeply coupled with the given HIV patients data set.

Another major issue is that the data filtering among visualization components would not be carried across. For example, if the user brushes a set of data from the parallel coordinate and then the user pick the pie chart as the next visualization path. The pie chart would query the entire data set again without applying the filtering constraint in the previous parallel coordinate. We didnt implement this feature because the given patients data set has a relative small size after the user apply a multiple round of data filtering, there might not be enough data point left to visualize.

The researchers have also complained about having trouble with some of the data columns name. Since not all of the researchers are familiar with the raw data set, and the columns naming are normally in their acronym or abbreviated way which is easier for data table schema creation.

DISCUSSION AND FUTURE WORK

In the future, we plan to continue on consolidating the code base, refactor the current fat controller and split each visualization component related code into individual Angular.js service. In addition to that, we also want to implement some of the missing interaction functionality or explore new opportunities for some of the visualization components.

Another part of the future work would be exploring how can we build a visualization tool that can be deployed in a very simple way, almost like a plug and play solution. Although we have implemented the prototype as independent from the data source as possible, developers still need to look at the given data set and categorize the data columns.

Using the current 3rd party libraries, we can easily incorporate more visualization components in the future, or even implement new ones using D3.js. Those newly added components can be seamlessly incorporated into the current tool.

Researchers at the AVRC have also expressed their interests to use the presented visualization tool for additional data sets that are much bigger than the current one, including the geographical distribution that would potentially be much wider compared to the current one,

CONCLUSION

In this project we have introduced a web-based prototype data visualization tool that supports four major visualization components towards exploratory data analysis. The tool itself is designed to be independent from the data source, only a minor development efforts is currently needed to incorporate a new data set. We have also introduced the concept of visualization path which is a composition of multiple different visualization components that allow the users to explore the data and understand them from a new perspective.

We have evaluated the tool by importing the San Diego - Tijuana border areas HIV AvantGarde epidemiology data set and provided the toolkit to medical researchers at AVRC. Based on their feedback, the prototype tool is quite intuitive and flexible to use, the combination of general data visualization components and the geographical heat map have helped the researchers to better understand the HIV epidemiology near the border area.

ACKNOWLEDGMENTS

I would like to thank Dr. Sanjay Mehta for helping us narrow the scope of our project, choosing relevant data fields, and explaining in detail the current workflow process for HIV researchers. I would especially like to thank my advisor, Professor Nadir Weibel for the continuous aid, advice, and feedback. I would also like to thank Sandy Law and Yingyan Hua, who also worked on related projects and have provided invaluable help as well.

REFERENCES

1. 2010 u.s. zip code tabulation areas. **https://www.census.gov/cgi-bin/geo/shapefiles2010/main**.
2. Angular-nvd3.js. **http://krispo.github.io/angular-nvd3**.
3. D3.js. **http://d3js.org**.
4. Gdal - geospatial data abstraction library. **http://www.gdal.org/**.
5. Geojson. http://geojson.org/.
6. Google chart tool. https://developers.google.com/chart.
7. leaflet directive for angular.js. **http://tombatossals.github.io/angular-leaflet-directive/**.
8. leaflet.js. http://leafletjs.com.
9. ng-table.js. **https://github.com/esvit/ng-table**.
10. nvd3.js. http://nvd3.org.
11. Open street map. **http://http://en.wikipedia.org/wiki/OpenStreetMap**.
12. Ruby on rails. **http://rubyonrails.org**.
13. Shapefile format. **http://en.wikipedia.org/wiki/Shapefile**.
14. Wrapper directive for google chart tools. **https://github.com/angular-google-chart/angular-google-chart**.

15. Hua, Y. Next Generation Time-based Visualization Tools for HIV Epidemiology.

16. Inselberg, A. Dimsdale, B. Parallel coordinates: a tool for visualizing multi-dimensional geometry. *Visualization, 1990. Visualization '90., Proceedings of the First IEEE Conference on*.

17. Johnson, B., and Shneiderman, B. Treemaps: A space-filling approach to the visualization of hierarchical information structures. *In Proceedings of the IEEE Information Visualization* (91).

18. Law, S. Avant-Garde HIV Research: Harmonizing and Visualizing Patient Data.

19. NIDA Avant-Garde Program for HIV/AIDS and Drug Use Research. **http://www.drugabuse.gov/about-nida/organization/offices/office-nida-director-od/aids-research-program-arp/avant-garde-award-hivaids-research.**

20. Sanchez, Melissa A; Lemp, G. F. M.-R. C. B.-G. E. C. S. R. J. D. The Epidemiology of HIV Among Mexican Migrants and Recent Immigrants in California and Mexico. *Acquir Immune Defic Syndr 37* (2004).

21. World AIDS Day 2014 Report - Fact sheet. **http://www.unaids.org/en/resources/campaigns/World-AIDS-Day-Report-2014/factsheet.**

22. Weixin Wang, Hui Wang, G. D. H. W. Visualization of large hierarchical data by circle packing. *In Proceedings of the SIGCHI conference on Human Factors in computing systems* (2006).