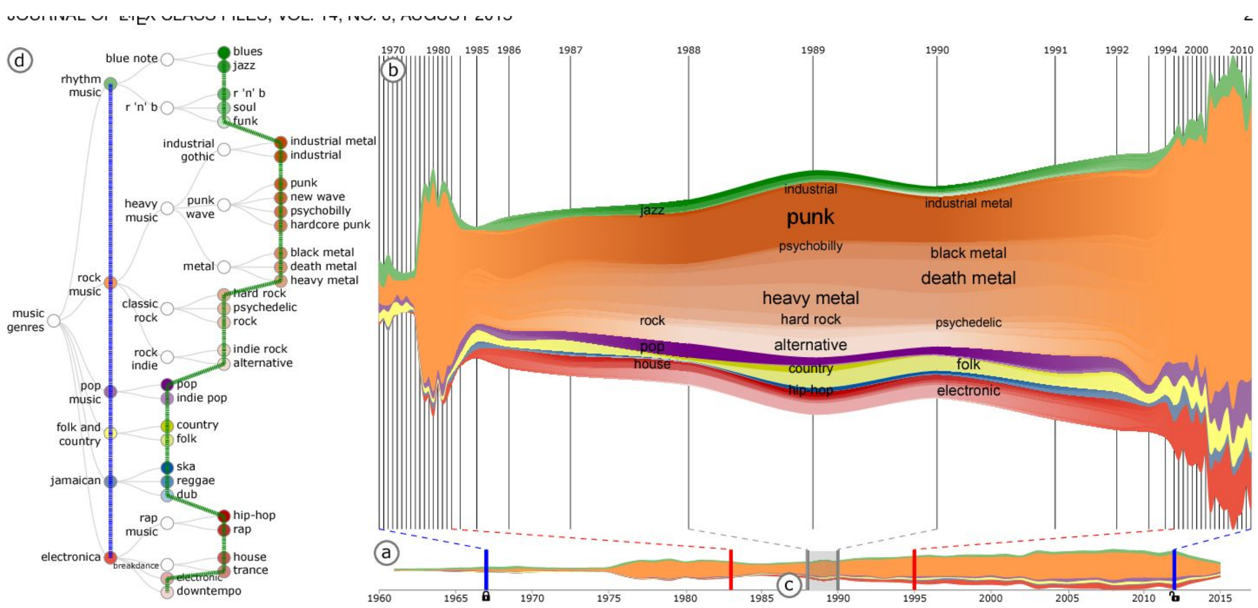
Process Book
Ellie Nguyen and Ryan Capps
Multistream Visualization Technique Study

Overview and Motivation

The goal for this project was to implement the MultiStream technique for visualizing hierarchical, time-series data and to investigate the effectiveness for this technique. The MultiStream technique seeks to allow a large amount of user exploration for this data while maintaining a clear idea of the context. It consists of three main visual sections: a node-link diagram representing the hierarchical structure of the data, a context steamgraph which displays the high-level data being visualized, and a focus steamgraph which displays lower-level data over a user selected range while showing how it fits into the higher level data. The image below shows the layout of these three sections. The hierarchy manager is on the left, the focus steamgraph is the upper steamgraph, and the context steamgraph is the lower. This technique also details various user interactions to aid in exploring the data, including using the hierarchy manager to change the level of data being displayed and using the context steamgraph to toggle the overall range of data being showed and the range over which the lower level data is shown.



Our main motivation for this project was a desire to gain experience working from technical visualization papers. We covered a large amount of material throughout the course but did not do much in terms of reading and understanding vis papers. We decided on doing a technique study as opposed to a design study because we felt being able to understand and work from these types of papers would be a valuable skill to learn.

Related Work
The work that most influenced our decision to choose this as our project was the second assignment from this class in which we worked with hierarchical time-series data. We both really enjoyed that assignment and knew that we wanted to work with that type of data for our project.

Questions: What are you trying to answer?
Our main questions for this project were: Can we make it work? And, how can we make it not work?

Our first goal was to work towards being able to read, understand, and implement a complicated vis technique from a technical paper. For the majority of the project we focused on making our visualization work. We selected a dataset we were certain would work with the technique and used that in order to test our vis as we built it up. We spent a lot of time reading the MultiStream paper, using the widget online to understand all the visualizations and interactions, and researching technical aspects of the project we didn't know how to implement.

Once we had our visualization in place and knew what we could put in to make it work, we wanted to find a dataset we thought would "break" it in someway to observe what happened. We wanted to see under what conditions this technique would work, and under what conditions it wouldn't.

Data
There are a few different data sources we plan on using for this project. The most important one for us is the dataset provided in Assignment 2 for this course (play counts for different music genres grouped in a hierarchy). This dataset is of the type that this visualization technique is designed for, and we already know that this data is clean and in a usable form. Additionally, it shares a very similar structure with the dataset used by the original authors to demonstrate this technique (similar depth and width in terms of total number of nodes and the number of children for each node). This dataset was the primary one that we used during our implementation of the MultiStream technique in order to make sure everything was functioning properly.

Once we implemented this technique to our satisfaction, we used the following dataset to investigate its effectiveness.

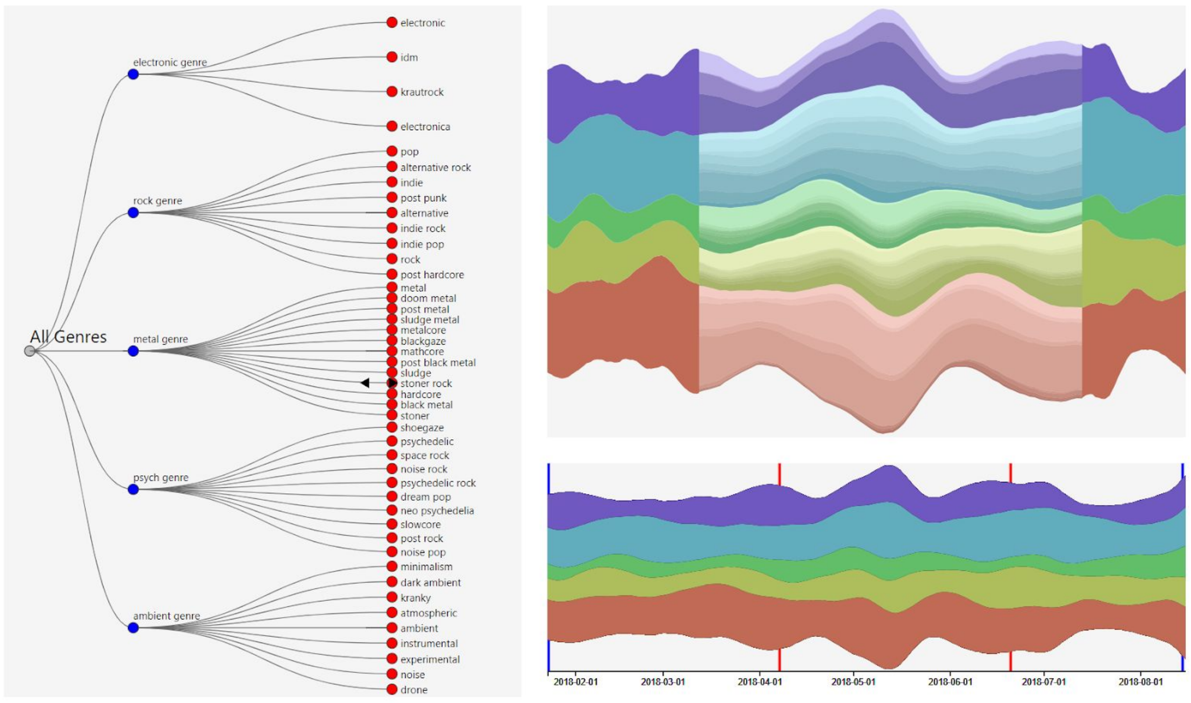
US Permanent Visa Applications:

https://www.kaggle.com/jboysen/us-perm-visas?fbclid=IwAR2hrGC7JHY2MI9F1UKo-fo4Ysv10Sw6j6Bp5Z0oATNsI6bwxUmLRMuxKx

The data was processed in R. The dataset required some basic cleaning (fixing spelling errors in city/country names, standardizing the letter case for the entries, etc.) but overall was very clean and well organized. The main challenge was adding in the hierarchical structure to the data. In order to do this we added more attributes to create additional levels while still staying true to the data: for example, adding in a "continent" field based on the country value given. We used d3’s nesting functionality in order to create the hierarchical structure and aggregate values for all of the non-leaf nodes.

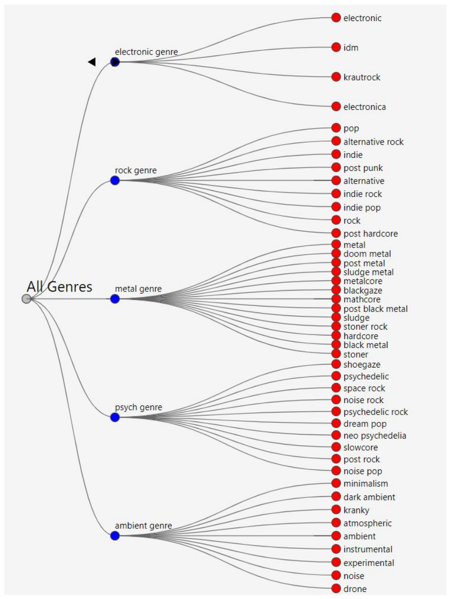
Implementation

This is an overview image of our implementation of the MultiStream technique. Each individual component and the interactions associated with it will be discussed further below.



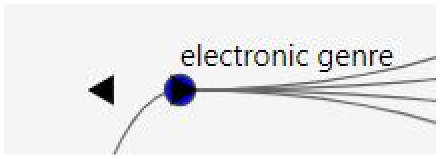
Hierarchy Manger

The hierarchy manager is located on the left-hand side of the visualization. It is a node-link diagram representing the hierarchical structure of the data being represented. Our tree-drawing algorithm, while not the most sophisticated, was able to effectively draw readable trees for this project. The reason a tree-drawing algorithm was written as opposed to using d3's built in functionality is because of difficulties encountered when trying to use data read in from a csv file. d3.tree() needs a very specific structure in order to work which was difficult to implement when trying to create a hierarchy from flat csv data.

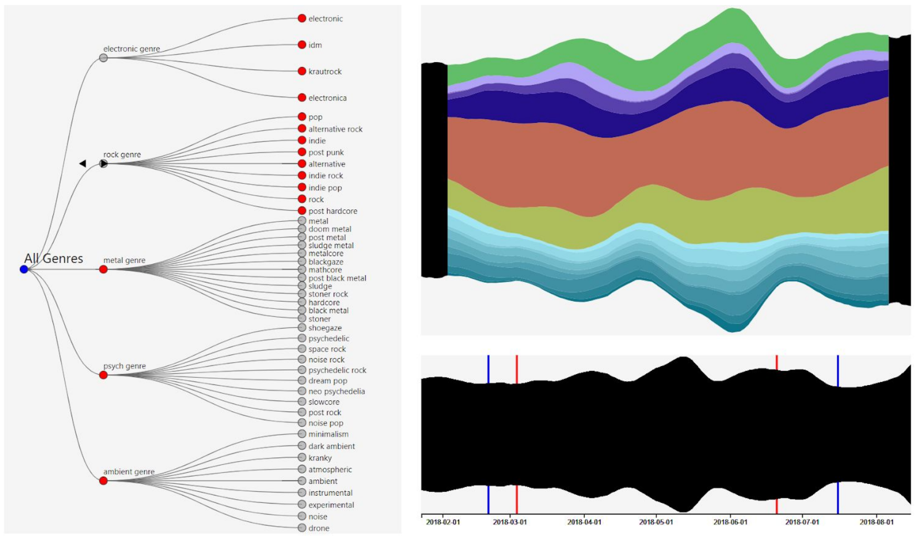


The main interaction involving the hierarchy manager is in the selection of data to be displayed. As seen in the above image, the node at depth 0 is empty, the nodes at depth 1 are all filled blue, and the nodes at depth 2 are all filled red. The blue nodes correspond to the "context" data, or the highest level of data being displayed. The red nodes correspond to the "focus" data, or the most specific data being displayed. The depth of the data being displayed can be changed by the user.

On mouseover, each node in the hierarchy manager will display a set of arrows (forward and back) allowing the user to change the level selected.



The implementation for aggregation and de-aggregation (left and right arrows respectively) are slightly different. When de-aggregating data, any one node can be changed without it affecting others. For example, we could display the more detailed breakdown for just the electronic genre and still have rock, metal, ambient, etc. displayed at their highest level.



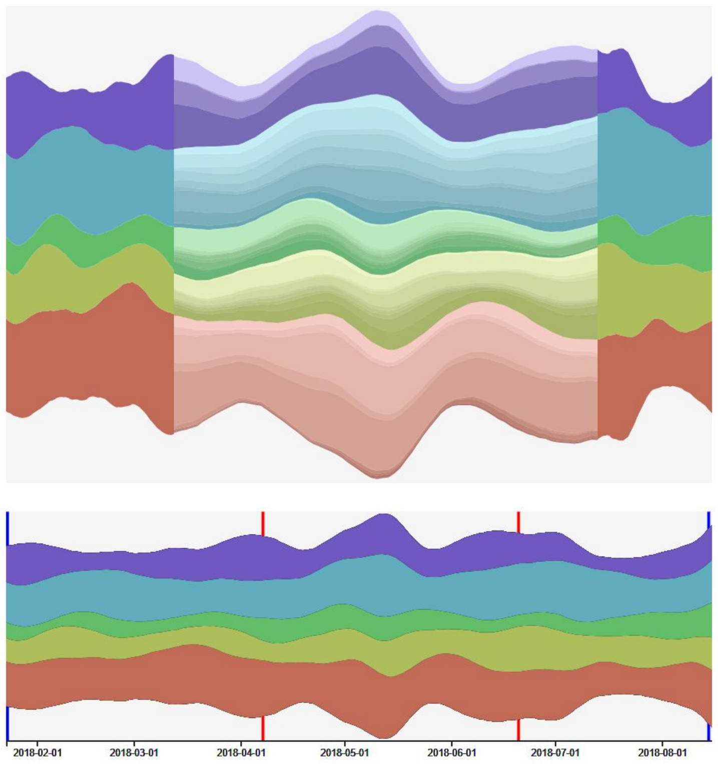
Aggregation of the data is much simpler: if any one child is selected to be aggregated, all children must be aggregated.

Steamgraphs

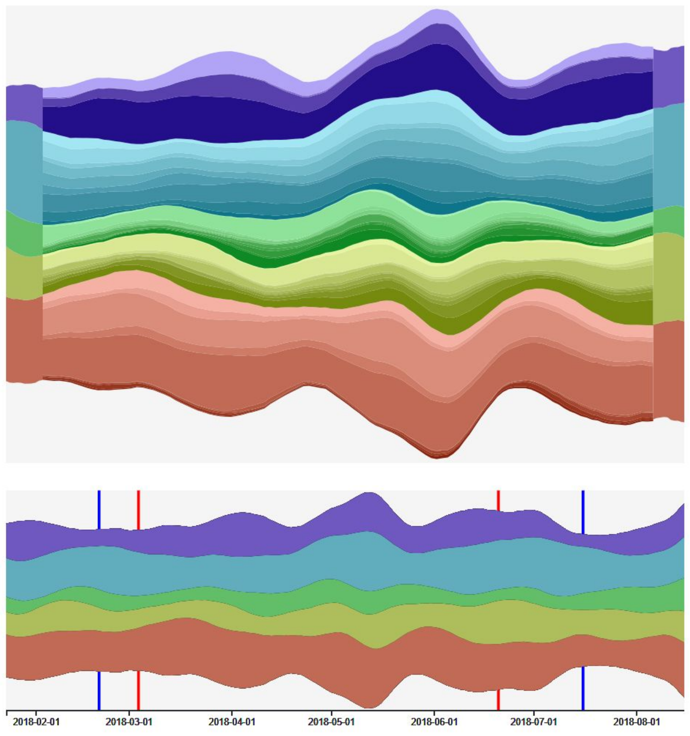
The other main aspect of this project was the implementation of the steamgraphs. In the lower right hand corner is the "context" steamgraph. This steamgraph displays the higher-level data selected by the user in the hierarchy manager along the entire timeline of the data. Here, the user is able to control the time-range of the data being viewed in the "focus" steamgraph.

There are two main time-ranges that can be controlled from the context steamgraph: the overall time-range being displayed, and the time-range over which the lower-level data is being displayed. These are controlled by a set of blue and red lines respectively. The interface is fairly straightforward: the user can click, hold, and drag the lines to change the values for either of the time-ranges. An important feature to note is that the focus steamgraph implements a "fisheye" view of the focus data. The focus data will always be centered and expanded to allow created exploration and easier visualization. This means that moving the lines in the context steamgraph will not result in a one-to-one change in what is being visualized above. The images below demonstrate how the steamgraphs change when the context and focus time ranges are changed.

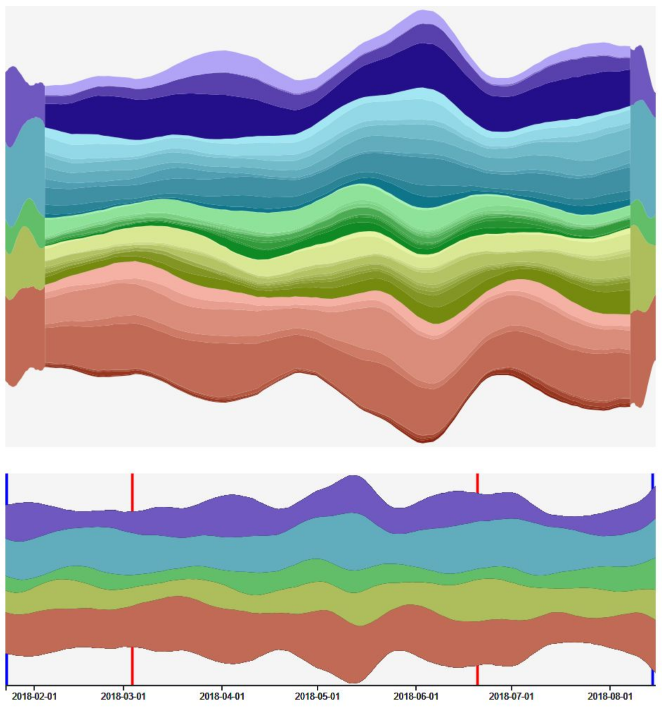
Before Any Changes



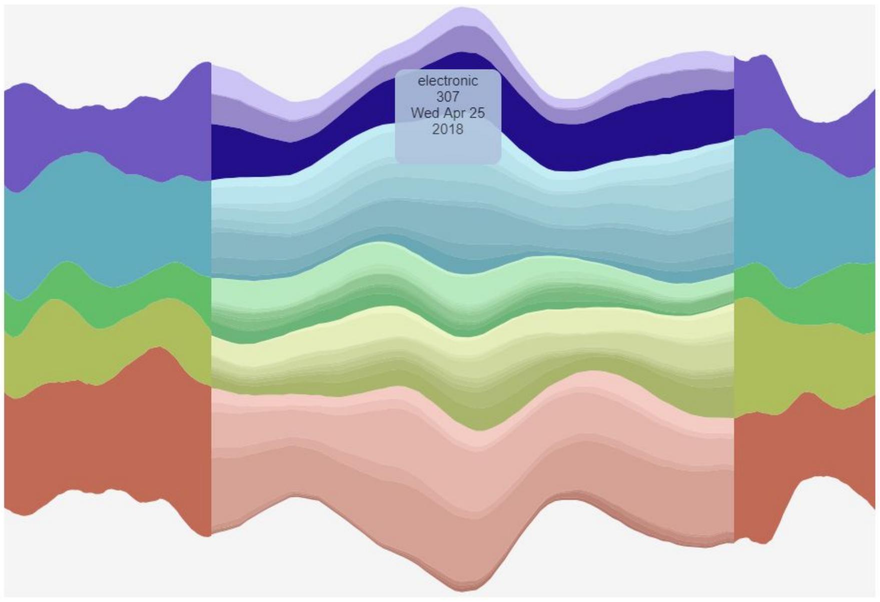
Change in context time-range



Change in focus time-range



In the upper right hand corner of the visualization is the “focus” steamgraph. It displays the lower-level data selected by the user over a certain time-range with the higher level data surrounding it to provide additional context. The focus steamgraph is where most of the data exploration occurs. As discussed earlier in this section, the user can use the hierarchy manager and context steamgraph in order to select what level of data is being displayed, and over what time-ranges to display that data. One additional feature for this steamgraph is the implementation of a mouseover interaction. On mouseover, the user will be provided with additional information of the point they are on including: the genre, date, and value. Being able to see the individual values for any point being explored allows for quantitative analysis between genres as well as visual.



Visualization Technique: Difficulties in Implementation

1. Reading in files. In class, reading in files was pretty well taken care of for us and we didn't have to concern ourselves with it too much. In order to implement this technique however, we had to learn how to read files in ourselves. The biggest confusion stemmed from the returns of d3.json and d3.csv. While I was expecting some kind of useable dataset, the return is actually a "promise": the function will return and appear "completed" but trying to use the dataset in another function before the promise is fulfilled will result in an error. In the assignments this problem was taken care of by using a .then statement after the d3.json call. We used this for our .json file but used a an async function with an await call for our .csv data. This was done just to learn more about some of the other functionality in js not covered in class.
2. Javascript's implementation of dates was a little confusing at first. In our aggregation function of the .csv data, we have a check to see if a certain date is already in our aggregation array before adding another entry. If you have two dates in js - d1 and d2 - you can not use d1 == d2 to check equivalence. You have to use d1.getTime() == d2.getTime(), +d1 == +d2, (d1>=d2 && d2<=d1), or any other variety of methods people have come up with. JS can not compare the "Date" value of the date, but instead needs to compare the numeric value of the date.
3. In implementing the focus and context steamgraphs, we utilized d3's built in steamgraph library. There was a pretty steep learning curve to using this effectively. The data had to be flattened out and built back up every time we wanted to draw a graph and the syntax and methodology for implementing graphs with this library proved unnecessarily complicated.
4. In order to implement the user interactions for this project we utilized tooltips which were not covered in class or the paper. Tooltips have the flexibility needed to implement the

labels on mouseover, the interactions with the lines on the context graph, and the arrows in the hierarchy manager for selecting the level of data to be viewed. We experimented with some other ways to implement this (creating functions on mouseover, creating shapes and redrawing on interactions, etc.) but found the tooltip implementations to be more straightforward and more responsive.

Evaluation

We used our two datasets in order to evaluate this technique. One served to evaluate how the technique worked under ideal conditions, and the other to evaluate how it performed with unideal data.

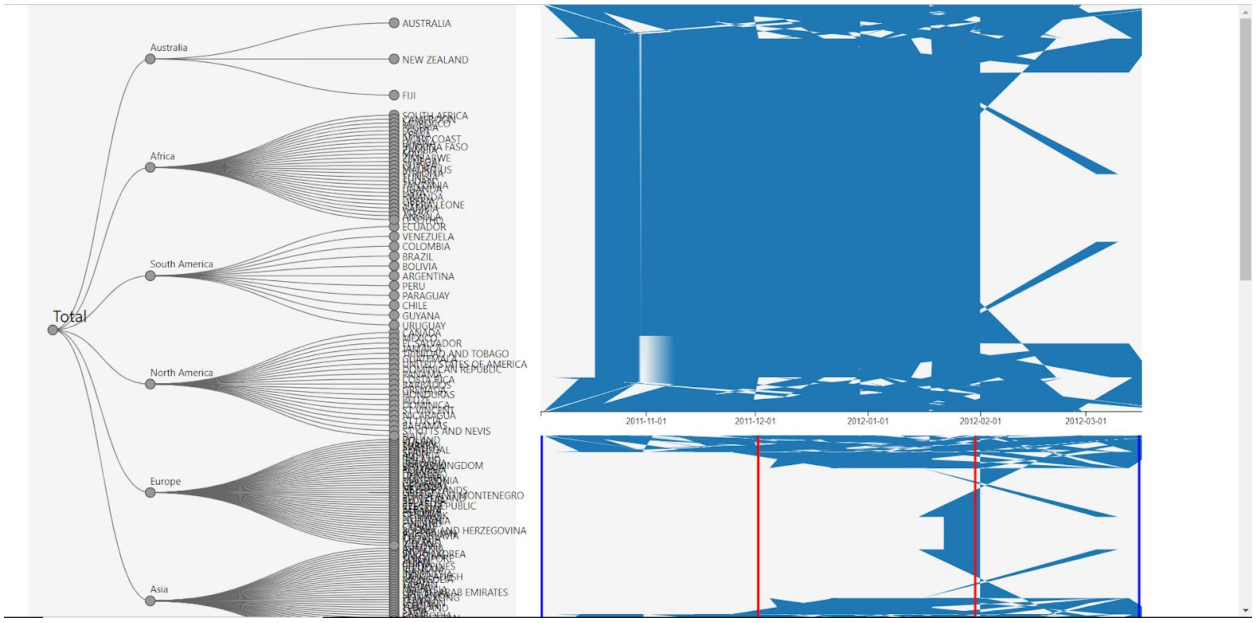
The ideal dataset used was the music series data from Assignment 2. As discussed earlier, this dataset was great to work with because it was similar shape/size to the dataset used in the original paper and was in JSON format (hierarchy was well-defined and extremely easy to read in).

The unideal dataset was the US Visa Application data we found. While this dataset still was a hierarchical time-series, there were a few aspects that we thought would test the technique well. Firstly, the hierarchy in this dataset is much wider than that of the original paper; meaning that each node had many more children, and there were many more points overall. Secondly, this dataset had irregular sampling of points. In the dataset from the original paper and in the music series dataset, every leaf node had the same number of samples, each sample on the same date, and each sample having a non-zero count. With this dataset, however, the sampling between leaf nodes was less consistent. Even when adding in data for the “missing” days, the values for them would be zero.

Analysis

When working with the ideal dataset, the technique works extremely well. Compared to some other vis techniques we looked at for this type of data, MultiStream provides a much greater degree of user freedom to explore and much better context for these exploration. Having the hierarchy manager and context steamgraphs displayed in addition to the focus steamgraphs makes it much easier to understand how what you are looking at fits into the bigger picture.

The unideal dataset gave us a view into some problems this technique may have. The figure below shows the output for our implementation when fed the US Visa Application dataset. The problems are pretty apparent. Firstly, the node-link diagram for the hierarchy manager is incredibly clustered with a large amount of overlap. While we did not use the best method for creating our tree, it is not enough to explain away all of the clutter. Given a set amount of space in which to visualize the hierarchy manager, it will always tend towards more clutter as the number of nodes increases. The problem with the steamgraphs can be explained by the issue with sampling discussed in the previous section. D3’s built in steamgraph library needs a continuous stream (read regularly sampled) with non-zero values in order to draw a steamgraph. With a custom written steamgraph function this technique may be able to work with this type of data, but we did not have time to try that for this project. If the steamgraph had worked as anticipated we would still be faced with a similar problem as the hierarchy manager. The sections of the steamgraph would become extremely thin and cramped, and with the number of colors that would need to be used they would be very hard to tell apart as well. While our implementation may not have been perfect to work with this type of data, it does still serve a valuable purpose in demonstrating how this technique would have difficulty with larger datasets.



Total
Australia
Africa
South America
North America
Europe
Asia
AUSTRALIA
NEW ZEALAND
FIJI
SOUTH AFRICA
ECUADOR
VENEZUELA
COLOMBIA
BRAZIL
BOLIVIA
ARGENTINA
PERU
PARAGUAY
CHILE
GUYANA
URUGUAY
TRINIDAD AND TOBAGO
UNITED STATES OF AMERICA
SAINT KITTS AND NEVIS
UNITED KINGDOM
MONTENEGRO
BOSNIA AND HERZEGOVINA
UNITED ARAB EMIRATES
2011-11-01
2011-12-01
2012-01-01
2012-02-01
2012-03-01