Last.fm Explorer: An Interactive Visualization of Hierarchical Time-Series Data

Maxwell A. Pretzlav
University of California, Berkeley
alex@turnlav.net

ABSTRACT

This paper describes an interactive web-based visualization of a large and complex dataset: the record of a single user's (or pair of users') music listening history. Last.fm Explorer facilitates sophisticated data exploration with interactive controls to drill-down through hierarchical levels of data, by showing the same dataset in two different visualizations, providing an interactive search tool, and utilizing animation to make transitions between multiple views clear.

It can be accessed at
`http://alex.turnlav.net/last_fm_explorer/`

INTRODUCTION

Since 2003 [13], the website Last.fm [6] (previously called Audioscrobbler) has been directly recording what songs people listen to. By installing a plugin for their music playing software which sends data to Last.fm, users can easily keep a record of all the songs they listen to. Last.fm makes this data available via a publicly accessible API, and offers a few simple visualizations on their website for casual users to view.

In 2006, Lee Byron created a minor sensation in the Last.fm and design communities with his "streamgraph" visualization [9, 14] of his own Last.fm listening history. By focusing on the onset of discovery of new artists and subsequent fade in interest as newer artists rose, Byron created a compelling and informative visualization. His work proved so captivating that several imitators cropped up that generate similar graphs for any user of Last.fm [8, 5]. While these visualizations are interesting to look at, they do not realize the full potential of Last.fm's data.

The author has been using Last.fm since 2004 to track his listening habits. In that time he has logged over 8,000 unique artists and many times that number of unique tracks. Viewing visualizations like Byron's streamgraph or those generated by [8] on a computer is a less-than-satisfactory experience; the graph is so wide it has to be scrolled horizontally to view different time periods, and as it is a fixed rendered image, there is no way of seeing more data than what is immediately visible in the image. The first shortcoming of these visualizations is that they only show one level of a users' data: artists listened to. Music data typically has four levels of hierarchy that users understand: genre, artist, album, and track. By going higher in the hierarchy, trends can be more easily identified, while detailed data about particular instances can be found by going down. Additionally, if someone is only interested in seeing the data for a particular artist or set of artists, it is very difficult to pick them out of the huge number of artists in a large fixed streamgraph.

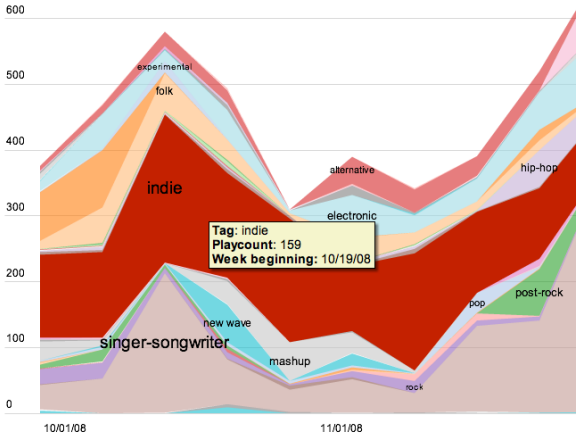

Figure 1. Last.fm Explorer's Stacked Graph

Last.fm Explorer attempts to resolve these issues through interaction. It initially presents a stacked graph of a single users' tag[1] listening data (Figure 1). By using controls adjacent to the graph and interacting with the graph directly, users can view different levels of hierarchy, filter the display by date and by text searches, view exact data for particular time points, switch between a stacked graph and a basic x/y plot, and re-arrange the stacked graph to better view changes of a particular tag, artist, or track.

Last.fm Explorer attempts to be accessible to typical Last.fm users. It runs in a web-browser, so it is not tied to any particular operating system and requires no software installation. It downloads a users' data on command, and shows results quickly and responsively. Additionally, it allows a

[1]Last.fm allows users to "tag" artists with identifying terms. Last.fm Explorer uses the most frequently marked tag for each artist, which is almost always the genre that the most people believe the artist fits into.

limited amount of social context by allowing comparisons between two users' history in parallel. In the author's own testing, this has brought up amusing and interesting results when comparing his data with friends' listening history.

RELATED WORK

As mentioned, Last.fm Explorer borrows heavily from work by Byron and Wattenberg [9, 14]. Previous web-based visualization systems include Many Eyes [20], a website which visualizes user-contributed data in a number of selectable visualizations, Name Voyager [21], a site that allows visualization of baby names over the last 100 years, and sense.us [19], a site that facilitated collaborative visualization, discussion, and annotation of U.S. Census data.

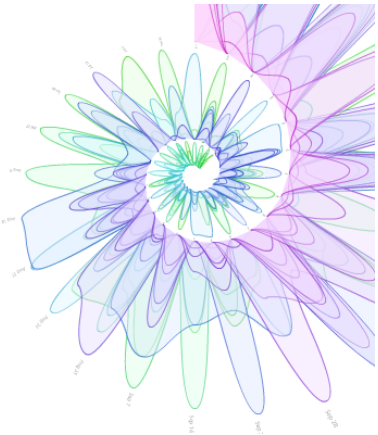

Figure 2. Last.fm Spiral

A number of visualizations of Last.fm data already exist. In addition to [5, 8, 9], the Last.fm Spiral [7] is a web-based interactive visualization of a single users' listening history (Figure 2). Its visualization style, however, is heavily stylized and difficult to read, moreover its interactive elements are non-obvious and difficult to use. The bar chart visualizations provided by Last.fm are easy to use but very simplistic (Figure 3). They allow interactive selection of what time period to view, but no visualization showing change over time, nor any sort of filtering mechanism.

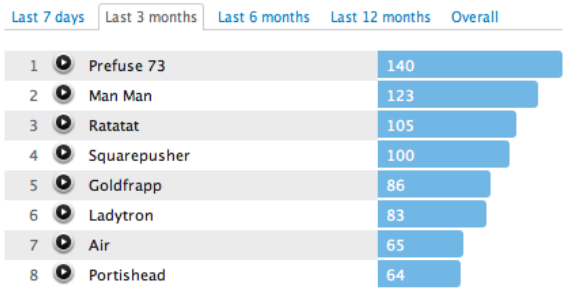

Figure 3. Last.fm's Provided Bar Charts

Last.fm Explorer uses animation heavily to make changes in visualization state clear. This was shown by Heer and Robertson [18] to significantly improve graphical perception while interacting with statistical data graphics. To make this possible, Last.fm Explorer uses Jeffrey Heer's Prefuse Flare [10] graphics library, which is heavily based on Heer's previous Prefuse library for Java [17].

In [15], Cleveland and McGill convincingly show that humans are much better at perceiving differences in the position of objects as opposed to length. This demonstration, as well as personal experimentation and discussion, was a strong motivating factor behind Last.fm Explorer's use of multiple visualizations as well as interactively allowing a user to re-arrange the stacked graph.

METHODS

Once finished loading a users' data, Last.fm Explorer presents a stacked graph of that users' tags over the last 20 or so weeks (Figure 5). At the top of the interface is a search box; the display is filtered based on any text typed in this box. For instance if the user types "rock" into the box, only data items with "rock" in their name will be shown—if the graph is currently showing tags, things like "post-rock" and "inde rock" will be shown in addition to simply "rock", if the graph is currently showing artists, only artists with "rock" in their name will be shown. The pipe character can be used to combine searches, such as shown in Figure 4.

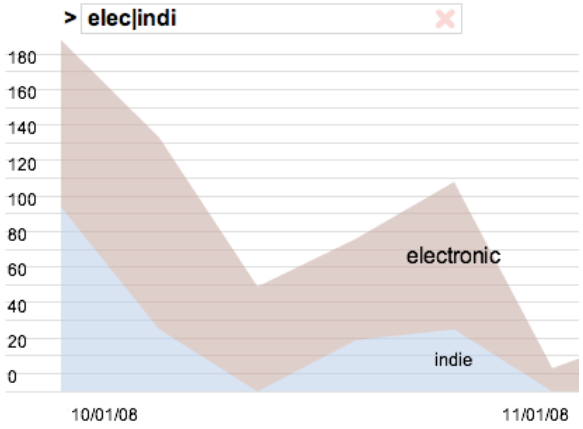

Figure 4. The Result of a Combined Query

To the right of the graph are three sets of visualization controls. The top pair allows the user to switch between the stacked graph (Figure 1) and the x/y "Line" graph (Figure 12). When viewing the stacked graph, the second pair of controls switch between an absolute stacked graph (Figures 1 and 5) and a vertically full percentage view stacked graph (Figure 8). When viewing a line graph, controls in the same position switch between using a standard linear vertical scale and a logarithmic vertical scale (Figure 13). The last set of controls allows the user to switch the visualization between Tags, Artists, and Tracks (Figure 6).

Below the main graph is a pair of arrowhead shaped handles on a slider which allow the user to adjust the left and right limits of the graph by date (bottom of Figure 5). This can be used to zoom in on a specific time span for a more detailed

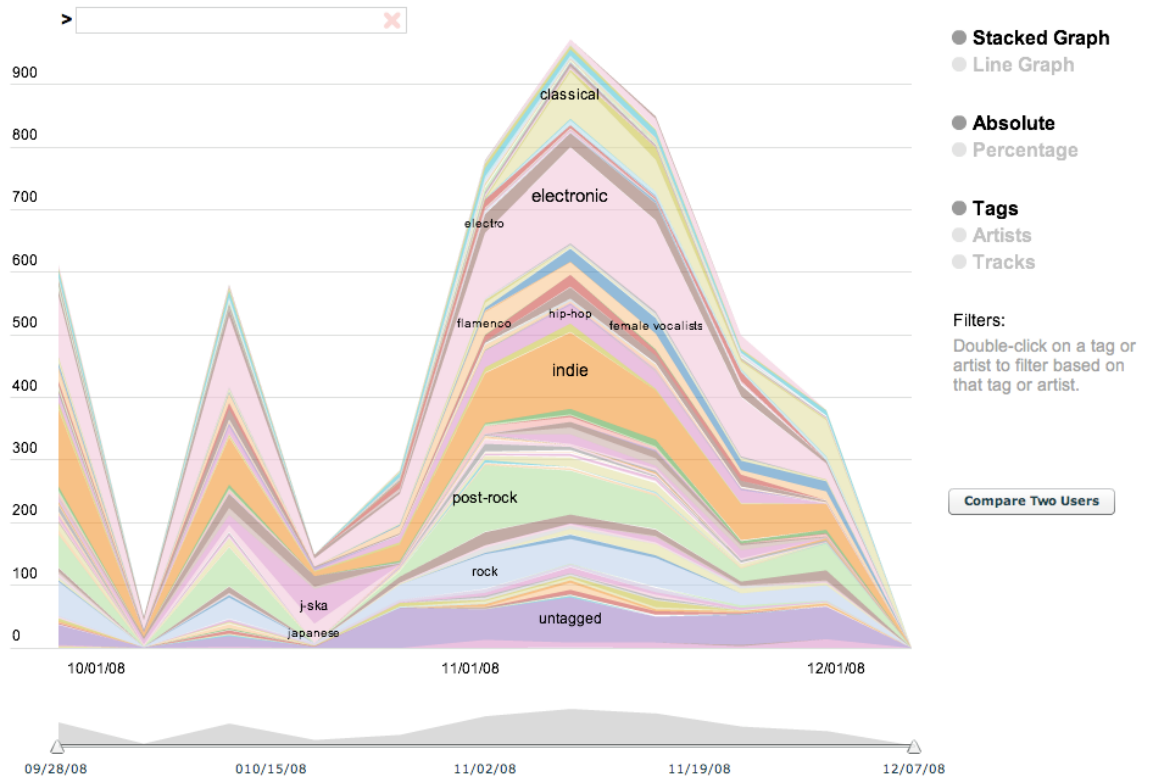

Figure 5. The Last.fm Explorer Interface Once Loaded

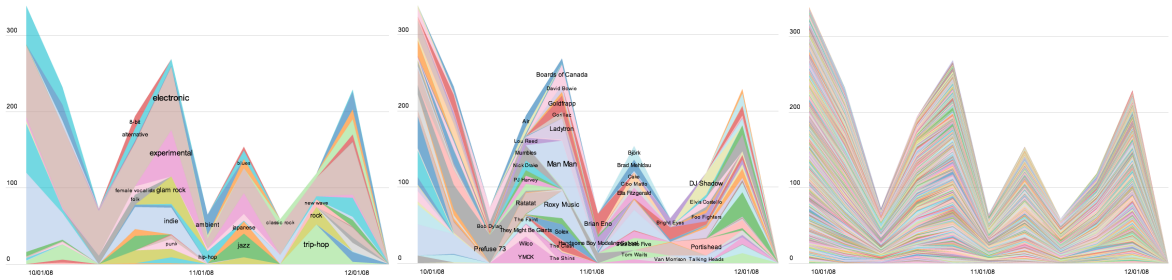

Figure 6. Three Views of Drill-Down: Tags, Artists, and Tracks

view. To make this time slider less confusing, a small version of the users' overall listening graph is drawn above the slider in the manner of a "scented widget" [22] so that overall context isn't lost when the display is zoomed on a specific region—while the large graph changes when the handles are moved, the small graph above the widget is fixed.

When a user hovers their cursor over an object in the visualization, such as a layer of the stacked graph or a line or node in the axis graph, the object is highlighted and a tooltip is shown next to the cursor with additional information regarding the particular object at the position the cursor points to (Figure 1). When a layer of the stacked graph is single-clicked, its position is swapped with the position of the bottommost layer; this allows any of the layers of the stacked graph to be viewed with a flat baseline, allowing better perception of changes in value (Figure 7).

When the user double-clicks on an artist or tag in the visualization, the display is filtered based on that artist and switched to the next level down of hierarchical data. For instance in Figure 9 the user has double-clicked on the tag indie and so sees all of the artists within the "indie" tag. In Figure 10 they have further double-clicked on the artist "Andrew Bird", and so they are shown all the individual tracks by "Andrew Bird". Clicking the red "x" next to the tag causes the filter to go away, showing all artists or tracks respectively. While the filter is in place the user can still move up and down in the hierarchy, but they will only see elements matching their filter criteria, so if "Artists" is clicked while a particular artist is selected to be filtered on, only that artist will be shown on the graph. This allows all state concerning what is being shown to be consistently presented and visible at all times.

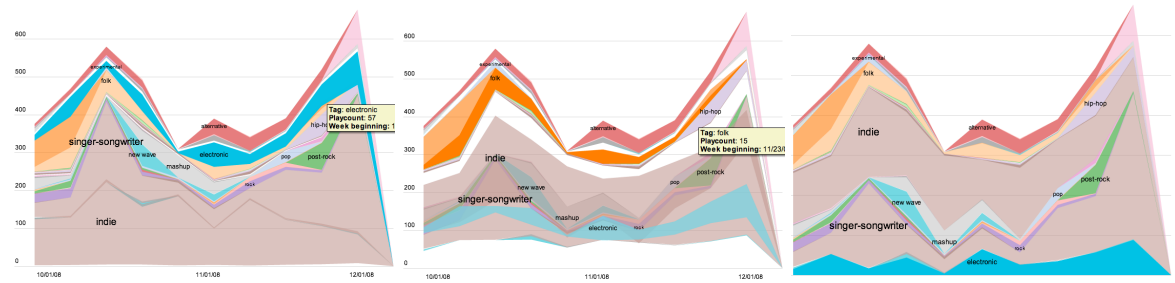

Figure 7. Animation of the Layer for "electronic" Moving to the Bottom

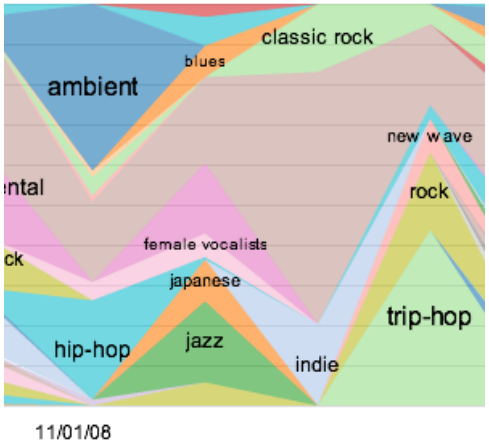

Figure 8. The Normalized Percentage Based Stacked Graph

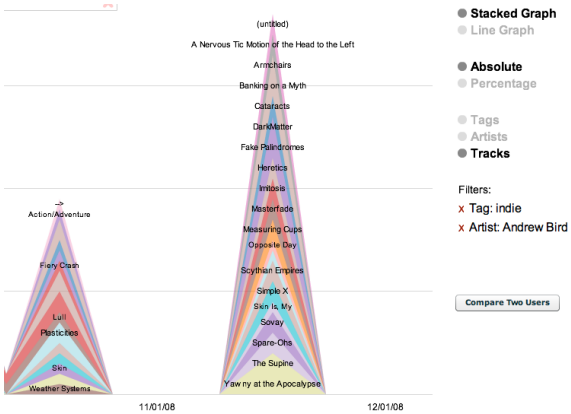

Figure 10. Further Drill-Down by Double-Clicking On the Artist "Andrew Bird"

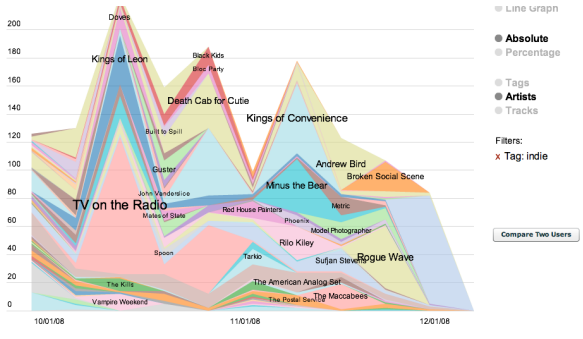

Figure 9. Drill-Down by Double-Clicking On the Tag "indie"

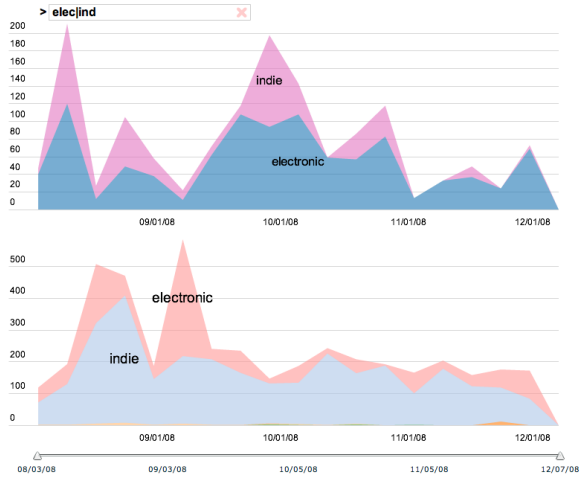

Figure 11. Comparison of the Tags "electronic" and "indie" For Two Users

The final feature of Last.fm Explorer is user comparison. Once a user's data has been loaded and visualized, a second graph can be created with a different user's data. The two graphs are linked to a single interface, allowing browsing and exploration of the two data sets in parallel—filtering, searching, and drilling-down is done on both graphs concurrently. This allows easy direct comparison between features of two users' listening habits (Figure 11).

Notes on Implementation

Last.fm Explorer is implemented as an Adobe Flash [2] application, written in the Actionscript 3 language, and utilizing the Flare [10] and Flex [3] APIs. It loads users' data on demand via Last.fm's REST API [4] in XML and processes the data for use in Flare's visualizations. A major hurdle of this project was reasonable loading times. Last.fm's API requires a separate HTTP GET request for every piece of data needed. For this reason Last.fm explorer only loads the last 20 or so weeks of a users' Last.fm data: loading the potentially hundreds of weeks of a users' full data would take longer than most people are willing to wait. A useful future feature would be allowing a user to subsequently request data further in the past.

The biggest challenge this limitation posed was retrieving the top tags for artists, to allow browsing data by tag. The hundreds of requests necessary to download the tag for every artist listened to in even a few weeks was causing exceptionally long load times. To mitigate this, Last.fm Explorer connects to a server-side script backed by an SQL database which caches the artist to tag mappings. This allows the mappings to be retrieved in a single request, provided the artists needed are already cached. If not, the script downloads the tags from Last.fm and inserts them into the database before sending its results back to Last.fm Explorer.

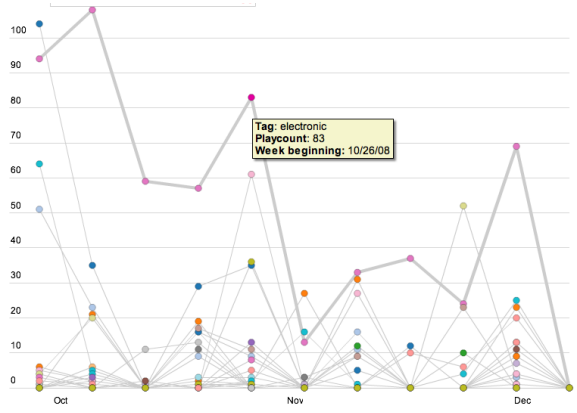

Figure 12. The Line Graph View, with Mouseover Highlighting

RESULTS AND DISCUSSION

While no formal studies have been conducted to quantify Last.fm Explorer's efficacy, preliminary user testing and discussion have shown that people familiar with Last.fm's data find the visualization exciting and interesting. Already the author has found interesting patterns that would not have been easy to find through existing means (the inversion of two users' preference for indie and electronic music in Figure 11 is a good example). Access to sophisticated visualizations of data relating to a person's personal life and habits appears to be very appealing, as evidenced by the popularity of sites like RescueTime [11] and ManyEyes [20], and the proliferation of popular visualization tools for Last.fm.

The stacked graph display seems to be a popular and accessible visualization for data of this sort. The biggest drawback of a stacked graph is the difficulty of comparing two different layers: as the graph stacks up, changes in multiple layers alter the display of higher layers, making changes in a single layer hard to view. While a line graph doesn't suffer from this problem, as explained below there are several other issues which make a line graph in this context difficult to read. The ability to reposition single layers of the stacked graph along the bottom of the display was an attempt to mitigate these problems, and several users commented on the effectiveness of this technique (Figure 7). The author has not seen any mention of a technique like this in previous literature or software.

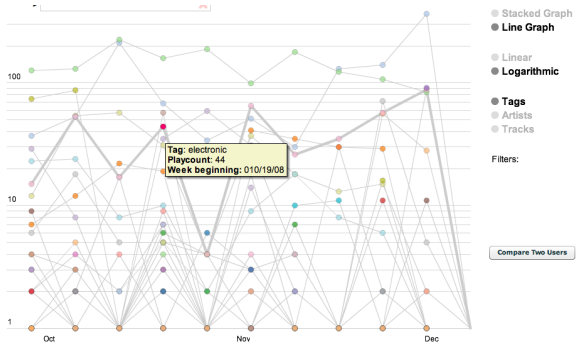

Figure 13. The Line Graph with a Logarithmic Vertical Scale

The line graph ran into several problems which limited its usefulness. As is visible in Figure 12, many users have a small number of artists or genres they listen to a lot of, and many they listen to a few of. This causes clumping of data in the bottom of the graph, with a few outliers higher up. The logarithmic display (Figure 13) mitigates this issue considerably, but at the cost of a very complex and hard to view graph. Additionally, the line graph is not directly labelled, so individual lines can only be identified by hovering the cursor over the graph to view the contextual pop-up. Lastly, as playcounts are integer numbers, it is not uncommon to have more than one node at the same point, particularly with low numbers—viewing all tracks in the line graph is nearly useless, as generally few tracks are listened to more than a few times in a given week (Figure 14). This makes the graph difficult to read, as only one node can be hovered over at time—to see the multiple artists etc at a node, the edges leading into it must be hovered over. This problem simply does not exist for the stacked graph.

Interactive performance was sometimes difficult to maintain. Particularly when showing data sets with many different nodes, such as all tracks, Last.fm Explorer sometimes momentarily hangs the browser it is running in. This is due

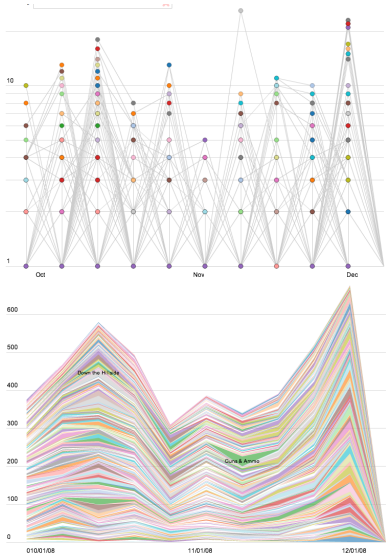

Figure 14. Top: a logarithmic display of all of a users' tracks shows the discrete nature of track counts. Bottom: the same data in a stacked view shows there are many more tracks than visible above.

to limitations of the Flash platform; Flash's performance drops dramatically when many individual objects are displayed simultaneously. This is exacerbated by Flash's lack of an interface for writing threaded processes as there is no way to show feedback while data is being processed and the display is being drawn.

FUTURE WORK

Last.fm Explorer attempts to be a fully functional and useful visualization tool. However, there are several features the author wanted to add but was unable due to technical and/or time constraints. The MusicBrainz [1] database is a publicly accessibly database of music metadata, including track lengths. By combining track length data with Last.fm's listening data, actual time spent listening to specific tracks, artists, and tags could be visualized. This would be particularly relevant with users who listen to music with small numbers of long tracks, like classical or post-rock, as visualizing play counts alone gives a skewed view in these instances. Gathering this data proved too time-consuming; as far as the author could tell MusicBrainz requires an separate network request for every single track a user listens to. If a sufficiently large cache could be built, then this might become feasible.

Currently Last.fm Explorer does not use the same selection of colors for the same elements in different visualizations: the color of the stack for "indie" in the stacked view does not match the color of the nodes for "indie" in the line view. Additionally, as is visible in Figure 11, colors are not connected when comparing multiple users. This is a significant shortcoming, as having the same elements colored differently is confusing to say the least, and significantly reduces the effectiveness of using colors to label categorical data.

Last.fm Explorer organizes the layers in its stacked view alphabetically. In [14], Byron and Wattenberg outline a number of sophisticated algorithms for sorting and coloring stacks in a streamgraph, which is closely related to a stacked graph. It is not clear how effective these algorithms would be in an interactive visualization where the stacks can be re-ordered, but an examination of these algorithms in this context could be beneficial.

As previously mentioned, the line graph suffers from a number of shortcomings. A non-obstructive direct labeling algorithm would greatly help interpretation of the graph when it is not too dense, however as the graph reaches densities close to that shown in Figure 13 its effectiveness would diminish. [16] describes an interactive technique for labeling very dense datapoints; an examination of techniques like this for viewing the line graph's data would most likely be beneficial.

Sense.us [19] uses a sophisticated technique for synchronizing application state with unique arguments appended to the application's URL. This allows direct integration of a browser's back and forward buttons, as well as preserves the steps of a user's interactive exploration in the browser's history. By storing the state of the visualization in the URL, users can email or otherwise send links of the visualization in a specific state, facilitating social exploration and discussion. The open-source library SWFAddress [12] is designed specifically to facilitate this technique; integrating it into Last.fm Explorer would add a new dimension of social interactivity.

Acknowledgements

An initial version of this application was co-developed with Nicholas Kong; a fair amount of his code remains in Last.fm Explorer. Thanks to Nick Kong, Maneesh Agrawala, Kealie Goodwin, and Scott Murray for useful discussion and feedback on aspects of Last.fm Explorer.

REFERENCES

1. About MusicBrainz. `http://musicbrainz.org/doc/AboutMusicBrainz` retrieved 12-14-2008.

2. Adobe Flash Platform. `http://www.adobe.com/flashplatform/` retrieved 12-13-2008.

3. Adobe Flex. `http://www.adobe.com/products/flex/` retrieved 12-13-2008.

4. Last.fm API. `http://www.last.fm/api/rest` retrieved 12-14-2008.

5. Last.fm Extra Stats. `http://www.lastfm.de/user/C26000/journal/2006/07/30/383m_last.fm_extra_stats` retrieved 12-13-2008.

6. Last.fm Homepage. `http://www.last.fm/` retrieved 12-13-2008.

7. Last.fm Spiral. `http://www.diametunim.com/muse/` retrieved 12-13-2008.

8. Lastgraph. `http://lastgraph3.aeracode.org/` retrieved

12-13-2008.

9. Listening History. http://www.leebyron.com/what/lastfm/ retrieved 12-13-2008.

10. Prefuse Flare. http://flare.prefuse.org/ retrieved 12-13-2008.

11. RescueTime. http://www.rescuetime.com/ retrieved 12-13-2008.

12. SWFAddress—Deep linking for Flash and Ajax. http://www.asual.com/swfaddress/ retrieved 12-14-2008.

13. Website offers new view of music. http://news.bbc.co.uk/1/hi/technology/2888431.stm retrieved 12-13-2008, March 2003.

14. L. Byron and M. Wattenberg. Stacked Graphs – Geometry & Aesthetics. http://www.leebyron.com/else/streamgraph/ retrieved 12-13-2008, Dec. 2006.

15. W. S. Cleveland and R. McGill. Graphical Perception: Theory, Experimentation, and Application to the Development of Graphical Methods. *Journal of the American Statistical Association*, 79(387):531–554, 1984.

16. J.-D. Fekete and C. Plaisant. Excentric labeling: dynamic neighborhood labeling for data visualization. In *CHI '99: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 512–519, New York, NY, USA, 1999. ACM.

17. J. Heer, S. K. Card, and J. A. Landay. prefuse: a toolkit for interactive information visualization. In *CHI '05: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 421–430, New York, NY, USA, 2005. ACM.

18. J. Heer and G. Robertson. Animated Transitions in Statistical Data Graphics. *Visualization and Computer Graphics, IEEE Transactions on*, 13(6):1240–1247, Nov.-Dec. 2007.

19. J. Heer, F. B. Viégas, and M. Wattenberg. Voyagers and Voyeurs: Supporting Asynchronous Collaborative Information Visualization. In *CHI '07: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 1029–1038, New York, NY, USA, 2007. ACM.

20. F. B. Vigas, M. Wattenberg, F. van Ham, J. Kriss, and M. McKeon. *Many Eyes:* A Site for Visualization at Internet Scale. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1121–1128, 2007.

21. M. Wattenberg. Baby Names, Visualization, and Social Data Analysis. *Information Visualization, 2005. INFOVIS 2005. IEEE Symposium on*, pages 1–7, Oct. 2005.

22. W. Willett, J. Heer, and M. Agrawala. Scented Widgets: Improving Navigation Cues with Embedded Visualizations. *Visualization and Computer Graphics, IEEE Transactions on*, 13(6):1129–1136, Nov.-Dec. 2007.