TOURVIZ: CS448B Final Project Report

Zachary Maurer
Stanford University
zmaurer at stanford

Santiago Seira
Stanford University
sseira at stanford

ABSTRACT

In this paper, we describe the design and implementation of "Tourviz" -- an application for visualizing the performance history of musicians. Tourviz seeks to help users visualize where an artist of interest has performed and who that artist has performed with at different points during their career. Many music fans are interested in this information, as demonstrated demonstrated by significant user activity at existing sites that aggregate this information like setlist.fm. Tourviz, however, addresses an unmet gap to provide users with interactive filtering features that present the data in such a way to facilitate discovery of interesting trends or previously unknown events. We use a set of mixed views for different information: a career timeline for context, a hexagonal heat map for geographic detail and a force-directed bubble chart to express relationships between artists. Following users' positive responses to our system, we assert that system described in this paper acts as a minimum-viable product to demonstrate the utility of Tourviz's design. Future work will be predicated upon increasing the speed of the application, providing functionality to compare artist's careers and creating a more integrated UI by allowing linking between views.

INTRODUCTION

Music fans want to learn about the performing histories of their favorite artists. However, there are no public applications that are effective at visualizing this information such that a user's original curiosity is satisfied. By creating an interactive visualization for artists' touring histories, we provide users with a way to learn more about their favorite artists. Specifically, we provide an application that helps answers the following questions for a user's favorite artists:

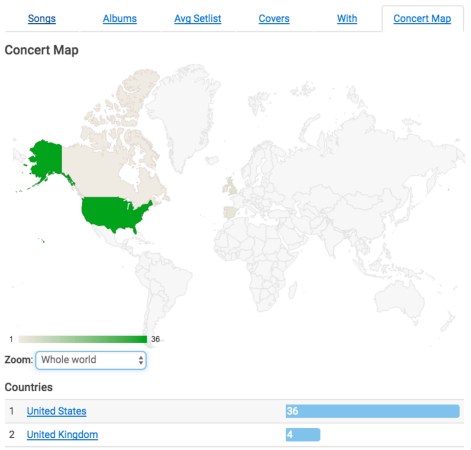
- **Where** did those artists play concerts during their career?
- **Who** did those artists perform alongside most frequently?
- What **types** of concerts does an artist tend to play? (e.g. festivals or smaller shows)
- What was the **trajectory** of an artist's career?

We hope our visualization will satiate users' curiosity. In addition, we hope that visualizing this information will facilitate their discovery of new artists, as users find unexpected performance combinations. Finally, our project will have experimented with a novel approach to visualizing aggregate trends in the music industry over time.

RELATED WORK

Existing work on this topic is very limited and generally ineffective.

Some visualizations cover reasonably sized datasets but lack any type of fluid interactivity. Views are static and the level of granularity in most of the views does not allow interesting trends to emerge.



The view above is from setlist.fm.[10] Mapping the frequency of charts purely at the national or state-levels does little to uncover interesting regional trends. Indeed, most popular artists of the past half-century will have toured extensively in the United States. As a result their visualizations will look incredibly similar, despite significant regional variation throughout their career.

Other visualizations focus on a hub-and-spoke type visualization.[11] Although geographically accurate, these visualizations emphasize an uninteresting part of the data. Users are not interested in the paths travelled by artists between venues. (This likely is more reflective of flight

paths and highway locations, than artist preferences anyway.) Users are much more interested in where, in a general sense, artists played concerts in some given period of time.

Furthermore, with many data points, a hub-and-spoke visualization becomes visually cluttered very quickly. While this can produce a visually appealing result[13], this obscures most of the map and makes regional trends hard to decipher. (Note: the following visualization[13] is representative of flight patterns in the US and is meant to reflect the density of information that accumulates with many data points.)

Finally, no visualizations that we found online attempt to visualize the amount of data that we aggregated and visualized in this project. Also, no visualizations online allowed users to filter by a time period in a fluid manner.

METHODS: DATA AGGREGATION

We started by researching appropriate data sources. We investigated manually web-scraping different websites for performance data, but realized these data were too sparse to be of use. For example, many wikipedia pages only included a couple of overall summary statistics for entire musical tours, as opposed to individual concert information.

During our search, we discovered that Songkick has an open API that provides historical performance data.[12]

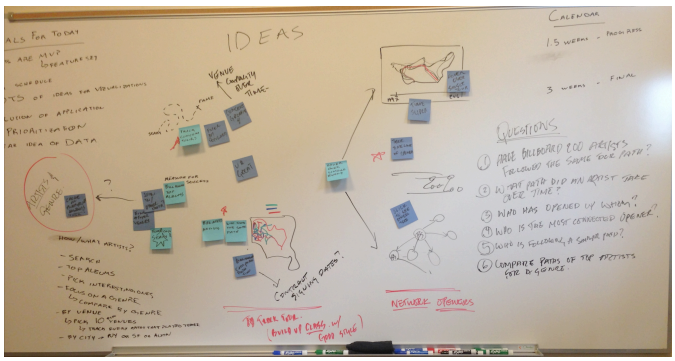
After confirming the functionality of Songkick's API, we searched for a broad list of artists of similar stature that would focus our requests from the API. We decided on focusing our visualization on Billboard's Annual Top Artists lists. To retrieve this data, we created a web spider with Scrapy[1] to crawl all Top Artist pages at billboard.com from 1958 to 2016. This effort yielded a list of roughly 2300 artist names.

To retrieve the data from Songkick, we set up a remote server on Digital Ocean that made requests to the Songkick API to retrieve (1) the artist's profile and (2) all events associated with that artist. This script had to be rate limited to comply with Songkick's policy, and consequently had to run for three days uninterrupted. We used a python snippet for rate limiting developed by Greg Burek[2]. These data were stored in a MongoDB database. We chose MongoDB because it would allow us to start implementing the project as quickly as possible, without having to design a SQL schema.

After three days of uninterrupted requests at (1 request per 10 seconds), we retrieved our entire dataset which comprised roughly 2k unique artist profiles and roughly 403k unique events.

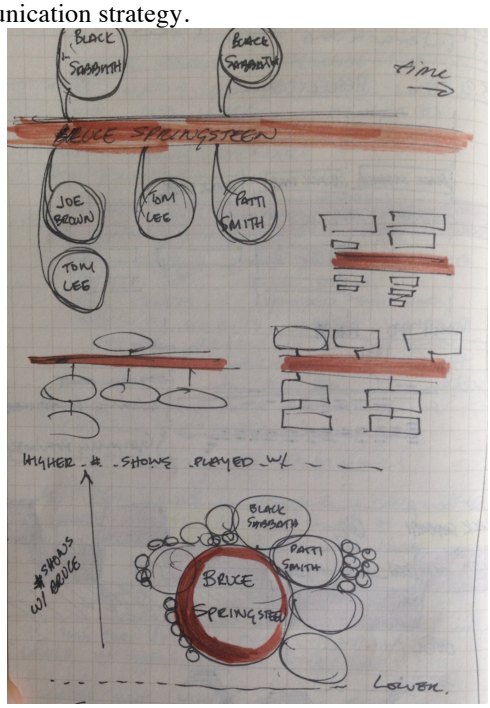
METHODS: DESIGN

Initial design attempts were done with paper and whiteboard mockups. This allowed us to quickly iterate on designs and make sure our team's collective understanding was consistent.



This process was incredibly useful for solidifying our understanding of why other visualizations had failed to convey interesting insights about musical performance data.

After initial brainstorming sessions, we communicated via sketches sent over instant message. Since we worked separately for most of the project, this was an invaluable communication strategy.



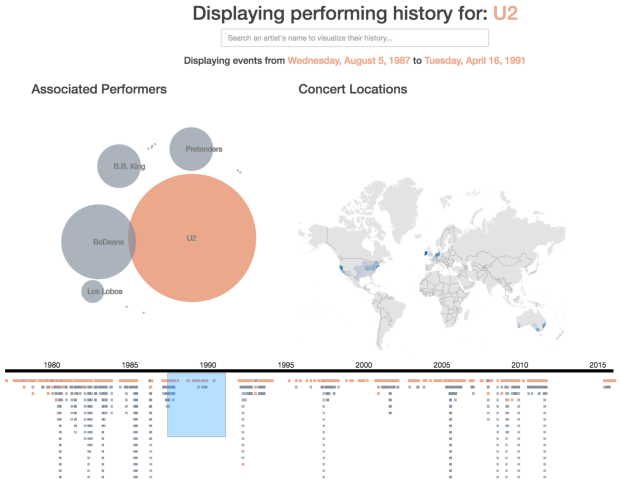
METHODS: IMPLEMENTATION

We used D3.js to create the visualizations.[3] All application services were run locally to minimize overhead associated with setting up a server and instead focus our development time on creating an effective visualization. MongoDB was again used as a local database to serve requests. Flask[4] is a Python web server framework that was used in conjunction with PyMongo[5] to query the database and serve data as JSON to the visualization application. The application itself was built with AngularJS[6]. Angular directives were used to compartmentalize the codebase and visualization views. A set of Angular services used GET requests to interact with the Flask server.

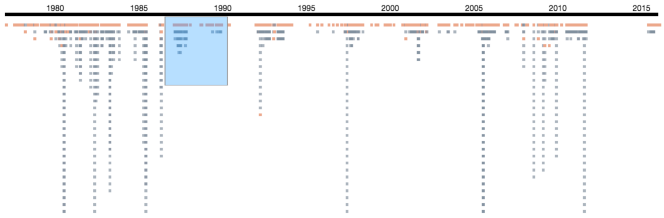
DESIGN STUDY

Our application has two views: the artist view and the aggregate view.

DESIGN STUDY: ARTIST VIEW



In the artist view, we wanted user to be able to see the entire context for an artist's performance history but also quickly drill down on specific periods of time. Additionally, we wanted the user to be able to simulate the passing of time relatively fluidly and visualizing the changes to the top-most detail panes in step. We believe that users are more interested in filtering by some "ballpark" time segment instead of limiting the search to very specific dates.

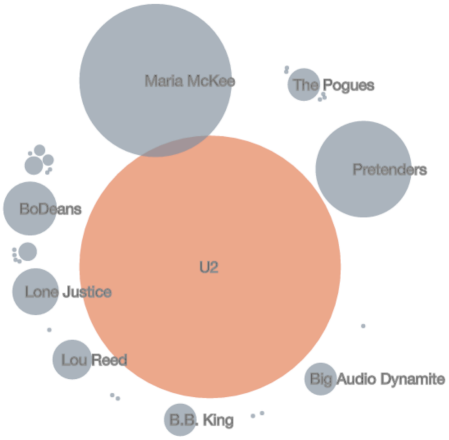


We portrayed the overall context of the artist's performing history with a minimized timeline. We drew inspiration from a d3 Block made by the user "bunkat".[7] Each column in the timeline represents a concert. The squares that make up the column represent individual artists that played at that concert. The red squares represent the artist currently selected by the user and the gray squares represent other artists. The closer a square is to the top of the column, the closer that artist is to headlining that event. Squares lower down the column represent supporting artists, lower on the show billing. The timeline allows users to brush over it to select a period of time which to limit the search that renders the upper two views.

(Note: For some artists, columns are actually very tightly packed and may appear as "one" column. This was a trade off we justified by providing the user with a consistently sized UI for each artist, regardless of career length.)

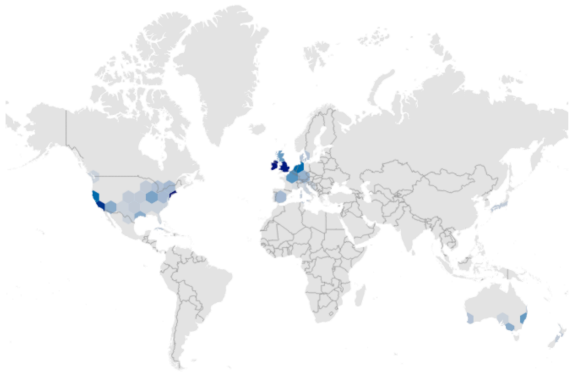
The leftmost view is a force-directed bubble chart[8] that portrays the artists that performed with the selected artist, during the selected period of time. A larger bubble indicates

a greater number of performances. Hovering on each of the bubbles provides the name of the artist and number of shows that occurred during the selected period of time. Forces were modeled with a gravity-like simulation, based on the radius of the bubble. All bubbles animated towards the center of the pane. This animation was chosen to give the display some dynamism and indicate to the user that the view had refreshed to reflect the new timeline brushing.



Although area is not the most perceptually accurate model for visualizing the number of shows, our intent was merely to quickly communicate which artists performed most frequently with the selected artist. We felt that a bubble chart provided a very quick way of visualizing this information and captured the user's attention with some element of dynamism.

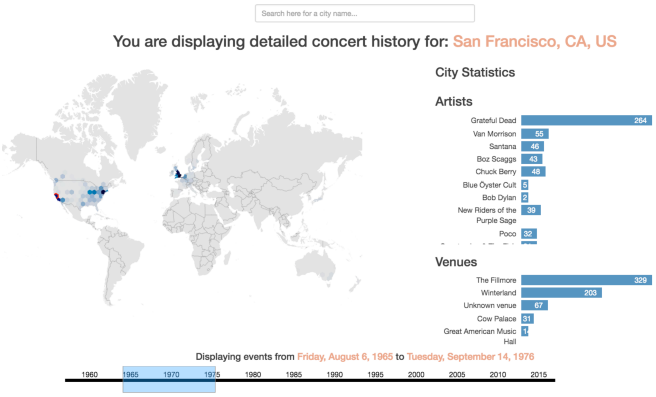
Finally, the rightmost pane visualizes the geographic location of the selected concerts. Again, perceptual accuracy was less of a priority, since were were merely trying to give users an intuitive, general sense of where these concerts occurred. The sizes of the hexagons were tuned manually to find a balance between geographic accuracy and visual impact, since smaller hexagons rendered a very sparse looking map where changes were not as noticeable.



Hexagonal bins[9] were chosen because they stacked on the borders of the SVG map most appealingly. We experimented with individual circles as marks for concerts that took place at a given location, but this led to a significant amount of visual occlusion. Moreover, users did not necessarily appreciate the specificity of information, as users were generally knowledgeable about major cities locations and were more interested in regional trends.

When users moved the timeline brush, these views update automatically, providing a very quick Query-View-Reflect cycle. This allowed users to manually "animate" the visualization and see the entire history of the selected artist in detail.

DESIGN STUDY: AGGREGATE VIEW



The aggregate view was meant to visualize some of the same geographic trends in the artist view, except for all 403k performances. This view also allowed users to search and select a city. Summary statistics were displayed in the rightmost pane. The upper chart labelled "Artists" charts the number of times different artists played in the selected city, during the period of time selected on the timeline. The lower chart labelled "Venues" charts the number of concerts that happened at different venues in the selected city, during the period of time selected on the timeline. Unfortunately, repeatedly sorting these views were not possible due to a limit within AngularJS. Sorting repeatedly caused a call stack overflow in the browser or would cause Angular's digest cycles to abort.

RESULTS

We feel that this project was successful as a minimum-viable-product and have plans for a more refined visualization in the future as a side project. The visualizations we created directly addressed the interests of users and from observing users interact with our software. We have found Songkick's data to be sufficient to provide answers to their questions. We will now break down our results based on our initial goals we described at the beginning of the paper.

Where geographically did artists play concerts during their career?

We addressed this question with two hexagonal heat maps that highlighted areas of the world where artists played their shows, one with aggregate data of all artists during a certain time period and the other for a specific artist of the user's choosing. The aggregate map provided the user an excellent way to quickly understand trends in the live music industry. Our one shortcoming in this visualization were our slow initial load times given that we had over 400k data points. The nature of Angular's digest cycle also limited us in regards to being able to sort the aggregate city data displayed next to the world hexagonal heat map with slow rendering times. We mitigated this by sorting the city data independently of the time filter which resulted in semi-sorted results but better than random. The artist specific hexagonal heat map had similar success in allowing the user to understand the overall tendencies of the artist's concert locations. However, we found that in this view, users wanted more specific concert data and expected to be able to hover over this map for such information. Users also mentioned that they wanted the ability to use the aggregate view as an initial search tool to then index the artist-level visualizations. For example, after searching "Los Angeles" in the Aggregate view, users would've liked to then be able to click on an artist's name and be redirected to the Artist view. We had hoped to implement this type of dynamic linking, but simply ran out of time.

Who did those artists perform alongside most frequently?

We aimed to answer this question through the bubble chart in the second view. After observing users interact with this visualization it became clear that what they were interested was strong partnerships between artists and they did not care about one-time set relationships. Luckily, by only displaying the name of the most frequent supporting artists our users were able to get an accurate sense of this very quickly. We also provided a hover over functionality to discover less frequent relationships but found that it was rarely used.

What types of concerns does an artist tend to play? (e.g. festivals or smaller shows)

We created a customized timeline with columns for events that displayed the number of artists that participated in that respective event. These columns had no labels which allowed the user to focus on the map and bubble chart yet allowed her to distinguish when events had 20+ artists, indicating a festival, or just a couple, indicating a smaller show at a glance. Once this feature was explain, users thought it was a great way to visualize this information quickly. The possibility of occlusion was not mentioned as a concern by users who demo'd the product, but it was something that we would want to address in future iterations.

What was the trajectory of an artist's career?

We approached this question by incorporating timeline filters for all of our visualizations. This way, the users were able to brush through an artist's concert history a couple years at a time and gain a rough understanding of the changes in concert location and supporting artists throughout time. However, given that load times were an issue, we only rendered results after the user selected a timeframe (i.e. on 'mouseup') and not while they were continuously scanning the timeline that resulted in us only partially achieving this goal.

DISCUSSION

After talking with users who interacted with our visualization, we noted two main, repeatedly vocalized impacts.

The first effect was that users learned new information the history of the live music industry and artist specific trends. After interacting with the timeline that filtered the events displayed on the aggregate hexagonal heat map, users quickly noticed how musician touring has intensified and globalized significantly, since 1958 and even more recently throughout the past 15 years. Users also appreciated the context provided by the timeline visualization. This allowed users to see the "shape" of an artist's career. Furthermore, although some users were very confident in their knowledge of artists' performing histories, our visualization cemented their knowledge or reminded them about facts they had forgotten.

Next, users really enjoyed the sense of discovery created by the artist-specific view. The bubble chart visualization of performing partners led to interesting discoveries for users on a frequent basis. Specifically, users were often surprised by who had performed alongside their favorite artists. Being able to filter by time in a very fluid manner facilitated the creative question development process. This discovery process created questions beyond the bounds of our application. After using our visualization, multiple users expressed an interest in using information about their favorite artist's co-performers to guide their listening experiences in the search for new music.

In addition to discovering this information individually, we also found that our visualization acted as a tool that could help users explain their favorite band's history to others. This was a very pleasant surprise. In particular, one student spent a significant amount of time explaining the history of Blink-182 to the authors. During this interaction, periods of inactivity displayed on the timeline served as a way of framing a discussion about dysfunction within the band and interesting pieces of trivia related to their touring patterns.

FUTURE WORK

We believe that the most desirable future extensions would further explore the similarity between artist's performing histories. One of the most exciting applications of similarity ratings would be for prediction and recommendation features. We could compare the initial concert paths of famous artists and search for new artists with similar paths to be able to predict which new artists are on track towards successful careers. We could also generate recommendations of similar artists based on their concert venues they played and the popularity of their shows at respective venues. By grouping successful artists, we would also be able to identify historical venues where a popular show has a strong correlation with future success and length of career.

Visually we would incorporate the ability to view multiple artist's concert history side by side and create a hover over feature for the hexagonal heat map to display detail information about specific concerts. In general, we would hope to integrate the aggregate view with the artist view through more dynamic linking.

To improve the speed and usability of the application, we hope to re-implement this project using Crossfilter[14] or dc.js[15] in order to overcome speed and performance issues.

CONCLUSION

Tourviz serves as a minimum-viable product to demonstrate the utility and demand for visualizing musical performance data in an interactive, multi-faceted manner. Having an integrated visualization that provides high-level context and more detailed views with the opportunity to focus on specific areas or times of interest is valuable to users and facilitates discovery. For such a tool, it is important to note the flexibility that users desire for creative discovery. For this reason, future design decisions must not over-emphasize the importance very granular, hyper "accurate" visualizations. Instead, future design choices should seriously consider the intuitive understanding that motivates and satiates users' curiosity.

ACKNOWLEDGEMENTS

We would like to thank the teaching team for developing an interesting course and providing support throughout the quarter. Also, We would like to thank Songkick for opening their API, and Mike Bostock and many others for developing D3.js.

REFERENCES

1. http://scrapy.org/
2. https://gist.github.com/gregburek/1441055
3. https://d3js.org/
4. http://flask.pocoo.org/
5. https://api.mongodb.com/python/current/
6. https://angularjs.org/
7. http://bl.ocks.org/bunkat/2338034
8. http://vallandingham.me/bubble_charts_in_js.html
9. https://bl.ocks.org/mbostock/4248145
10. http://www.setlist.fm/stats/concert-map/bruce-springsteen-2bd6dcce.html
11. http://vizzuality.github.io/rollingstonesmap/#/8
12. http://www.songkick.com/developer
13. http://www.aaronkoblin.com/project/flight-patterns/
14. http://square.github.io/crossfilter/
15. https://dc-js.github.io/dc.js/