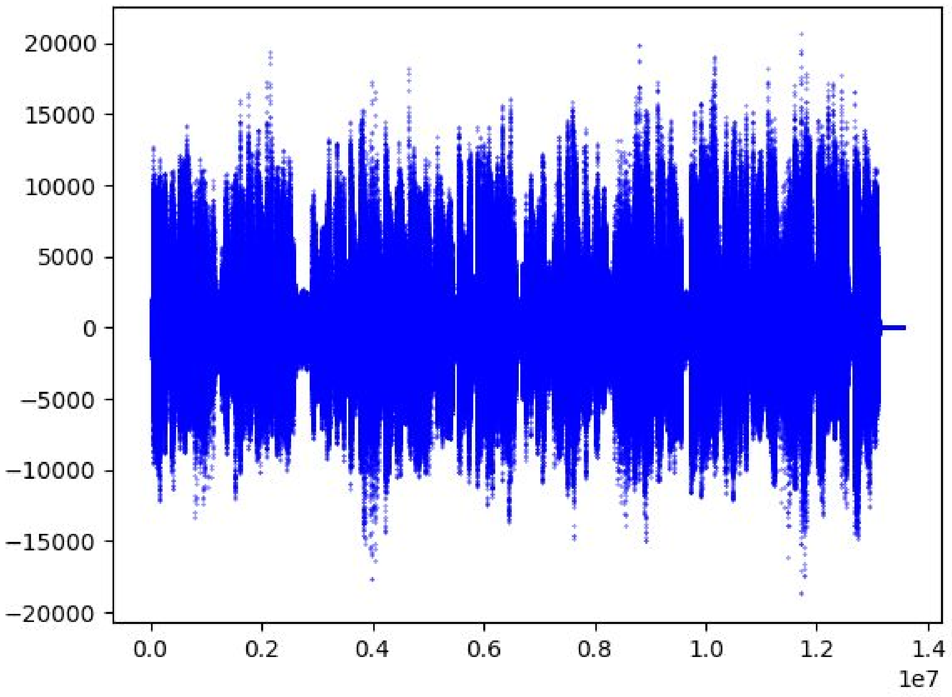
Introduction:
Our project aims to discover whether there are patterns in comedic timing and audience laughter during stand-up comedy sets. The target audience for this visualization is the general public and the visualization would best be used as part of an explanatory article. Our hypothesis is that there is a narrow range of time between the punchline and the beginning of audience laughter.

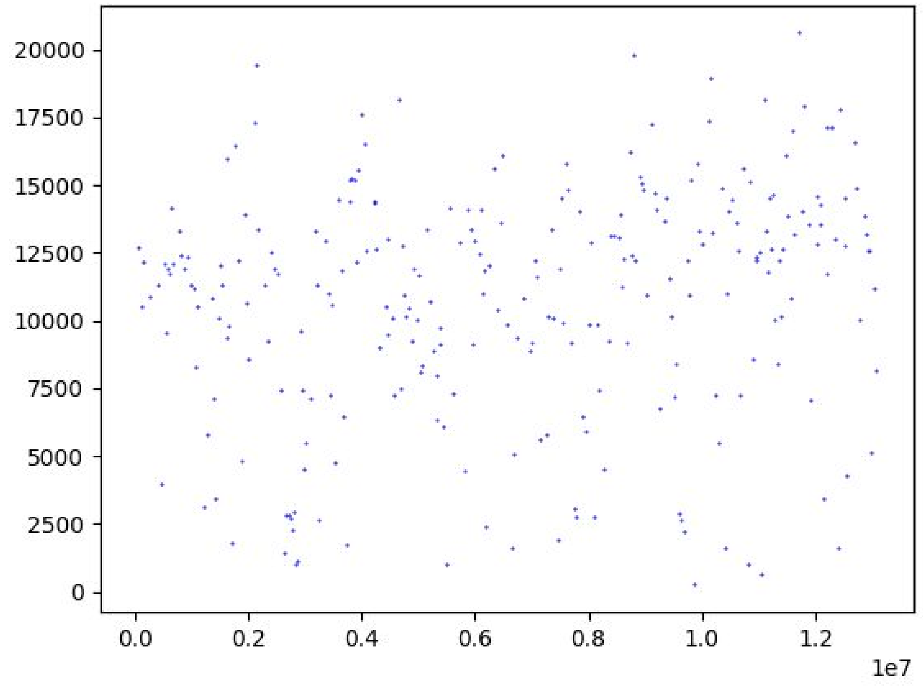
As well as being of interest to people in the general public, this information will be helpful for my (Valerie) undergraduate thesis, which involves comedic delivery. Other researches in the field of humor production might also find these results interesting and stand-up comedians could use this information to perform better. Although we focused on comedic sources, our process and algorithms could be used for the study of other speech recordings or even non-speech based sounds.

Related Works:
We found several related works. Attlardo, S., & Pickering (2011) is a study on the length of pauses during the performance of jokes. It closely relates to our work as this paper is about pause length in reference to the punchline, while our study will look comedic timing and rhythm more generally. What we can gather from this paper is the length of pause does not indicate a temporal similarity with a punchline. Leong V., Stone, M. A., Turner R. E., & Goswami U. (2015) looks at amplitude to determine speech rhythm. This is important for our research as we need to be able to determine the rhythm of the comedy being analyzed. We will look towards the techniques offered in this paper as examples to follow. Salamin H., Polychroniou A, & Vinciarelli A, (2013) is a paper about techniques to identify laughter in conversations. We will attempt to use their technique to identify audience laughter, though as their paper outlines this can be a difficult task. Petrovic S., & Matthews D., (2013) is a paper about an unsupervised neural net that attempted to generate jokes. This relates to our project as knowing what makes a joke, can help us find what to look for in our analysis of speech.

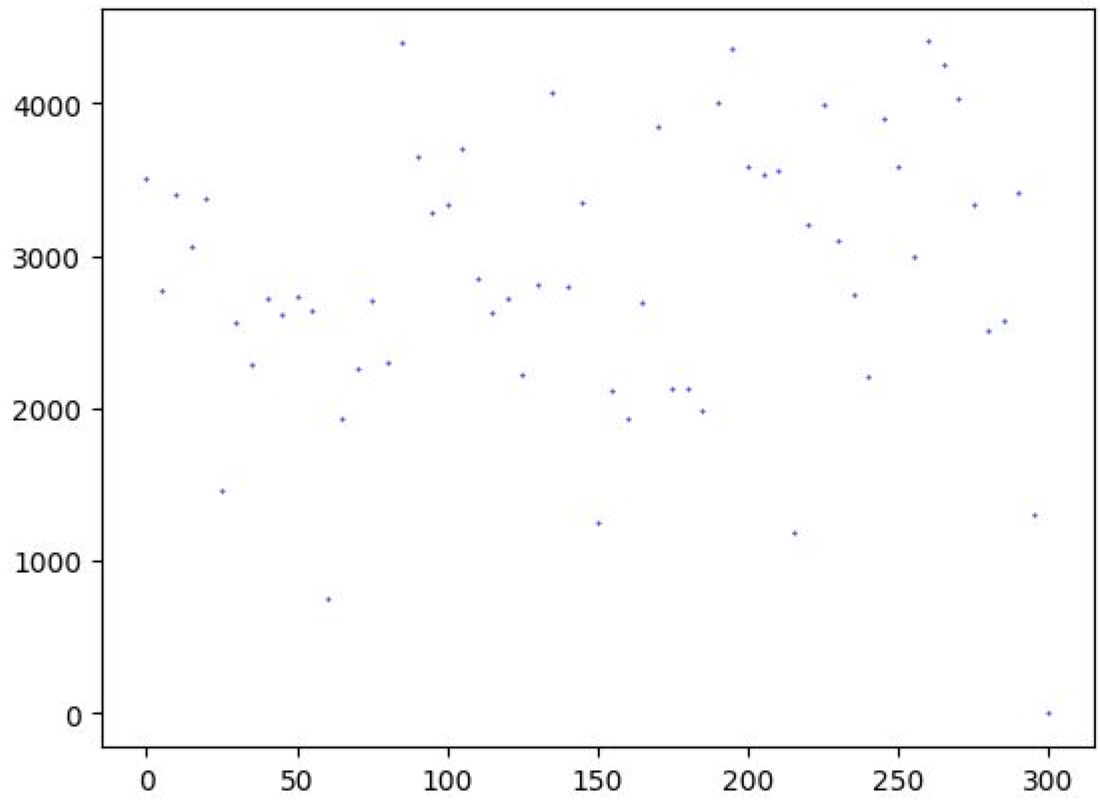
Process:
Initially we were to use a two split streamgraphs which will show the comedian's volume, the audience's volume, and have key moments in the jokes highlighted (e.g. the start and end of punchlines). As we worked on the data parsing we realized that getting the audience volume would be quite difficult and use sound techniques beyond our capability so we dropped that idea. Additionally, we ended up using a bar chart instead of a streamgraph as for debugging purposes we started with a bar chart and really liked the look.

Our codebase was separated into two different parts, the processing of the wav file into data done in python and the visualization done in Javascript. Below you can see some first visualizations for debugging purposes done using pyplot.

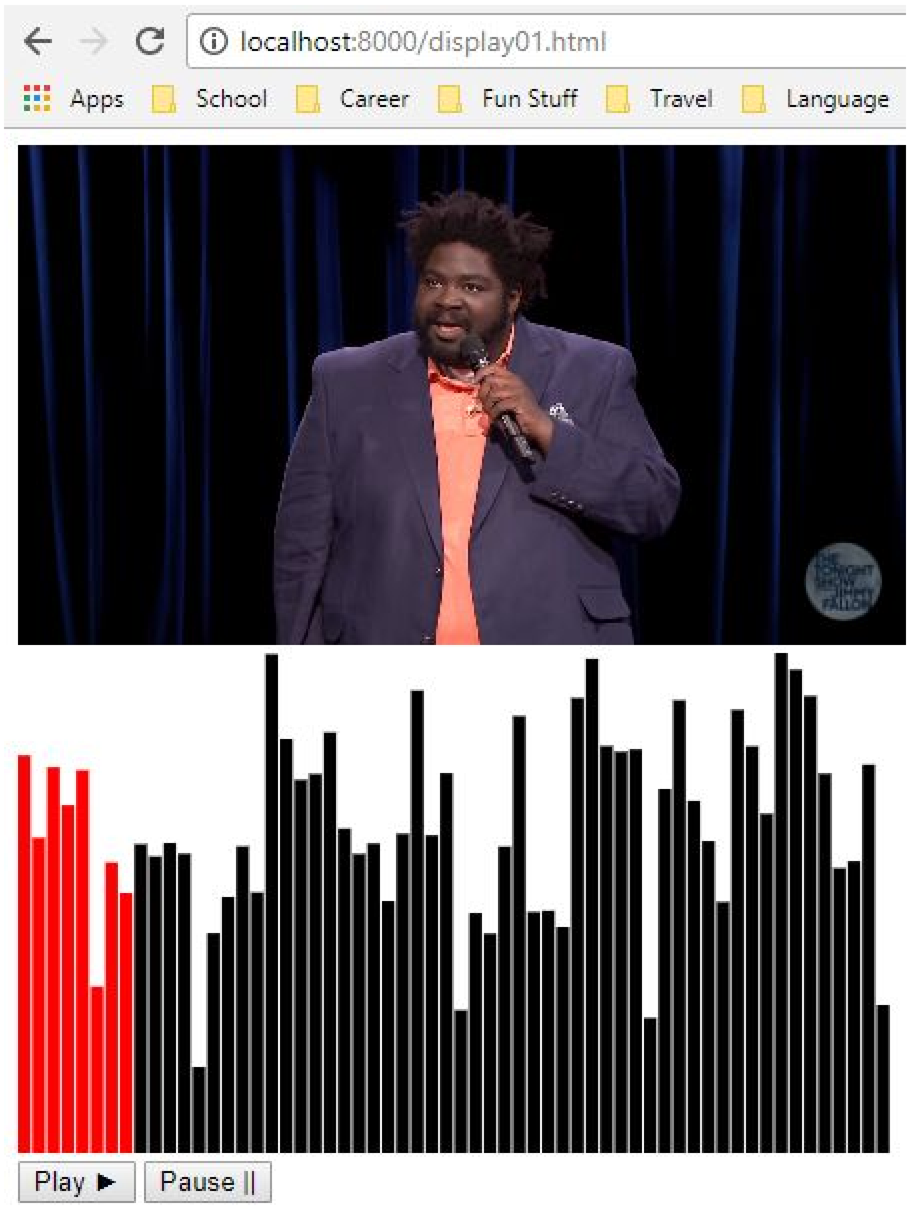




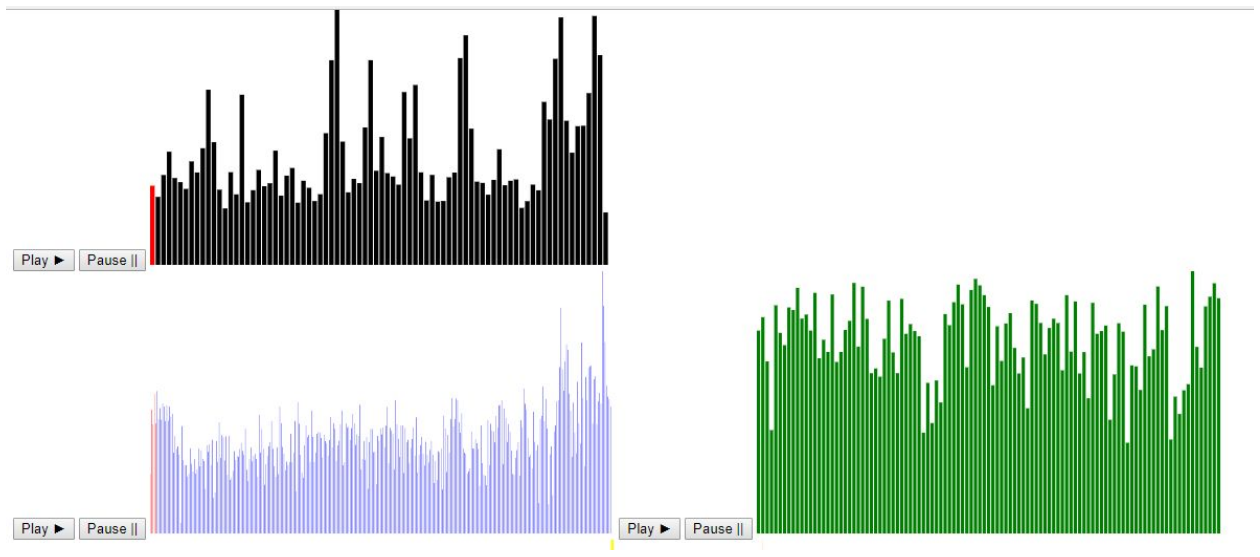
The top visualization is a visualization of the raw ~4500 Hz wav file. The bottom visualization is a 1Hz attempt to capture the amplitude of file.


4000
3000
2000
1000
0
0
50
100
150
200
250
300

Here is the file further compressed to .2Hz.



Here is that same data visualized using a bar graph from d3 with the corresponding YouTube clip.



Here is the data from three separate speakers. Comedian Ron Funches is in black, former president Barack Obama is in blue, and comedian Jerry Seinfeld is in green. Funches' and Obama's audio clips are currently being played.

Feedback incorporated:
For our visualizations, the majority of classmates polled preferred uniform joke widths over uniform bar widths, which was in line with what we were already creating. However, a few pointed out that it would be helpful to also compare overall joke lengths, which led to the drag-and-drop comparison pan. Here, each bar is a uniform width, so that a longer joke (more bars) would be a wider on the screen.
We also heard back that highlighting the bar that corresponds to the current playback time would be helpful to correlate what is being heard with what is being seen. This was incorporated into our final visualization. Another piece of feedback we did incorporate was for each comedian to be their own color. This was a simple way to incorporate color and help keep the different comedians separate when only their bars were visible (originally, as seen above, there were accompanying YouTube clips, but these proved to be too time-consuming to load).

Technical challenges:
For the processing of the wav file there were many technical challenges in developing a processing algorithm to capture the amplitude present in the raw data. To find the amplitude we needed to find the the local minima and maxima in the data. Due to the large quantity of points ~14 million per clip the python numpy algorithm was too slow so we developed our own algorithm to approximate the local maxima. This approximation developed over many iterations and the result is an O(n) algorithm that approximates the amplitude fairly well.

As mentioned in my original proposal blog post, implementation was a frickle-fracking challenge. The biggest challenge was writing it in such a way that I wouldn't have to hand-make every single button for every joke for every comedian. Aside from that, some changes wouldn't take effect for several minutes with seemingly no reason.

Serving the audio files takes a while and once one is served, the whole page has to be reloaded for another audio file to be played. This is by far the biggest hindrance in testing so far, and I hope to have this fixed by tomorrow.
As for other issues, it was difficult to balance a visualization that would be meaningful with something I could feasibly complete in the alloted time. People seem to be interested, and it has been helpful to work with this data, but overall I think the next helpful step would have been to have fit lines that could be stretched to more easily compare across different joke lengths.

Who did what:
Valerie did:

- Collected the majority of the clips and turned them from youtube videos to wav files.
- Worked on the javascript part of the visualization
- Wrote intro, feedback, and half of technical challenges portions of final report
- Wrote the majority of the project proposal

Alex did:

- Wrote the python wav file processor and python data plots
- Wrote the majority of the user study
- Made the final presentation
- Wrote related works, process, and half of technical challenges portions of final report