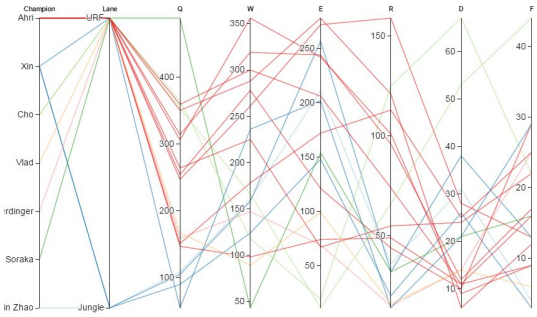
Final Report: League Analyzer

By Nathan Berton and Marshall Shortledge



Motivation / Purpose

One thing we noticed about League of Legends (LoL) players is that they click a lot. Their champion, the character the player controls in the game, is always on the move, and time standing still is time wasted. Being around both new and experienced players, there was an audible difference between them and we decided to visualize that difference.

Background on League of Legends

League of Legends is a Massive Online Battle Arena (MOBA) that has teams of 5 fight each other. Each player controls a champion in the arena and the end goal is to destroy the enemy's base. The arena has 3 lanes with jungle tiles between each lange. There are 5 different team roles: Top (top lane), Middle (Middle Lane), Jungle (roams between lanes helping out when needed), Attack Damage Carry (ADC, Bottom Lane), and Support (Bottom Lane to help the ADC). There is also a ranking system for players with the lowest tier being Bronze. The average skill of players increases as you move up in tier, Bronze < Silver < Gold < Platinum < Diamond < Master < Challenger. Within each tier are levels ranging from 1 to 5 with one being the highest and 5 being the lowest in terms of skill. For example, a player can move from Bronze 5 to Bronze 1, and then up a tier to Silver 5.

Central Research Question

We have chosen to investigate if it is possible to estimate a player's skill by their mouse and keyboard actions. Is there a noticeable difference between higher ranked, more skillful players, and less skilled players?

Hypothesis for data

We expect to see more experienced players click more and have more attack move commands (right clicks). We also expect to see a correlation between certain champions, the

heroes that players control, and key presses related to combinations of abilities and cooldowns on spells.

Target Audience

Our target audience is League of Legends players old and new. Older players can get an idea of their playing habits and newer players can look to see the habits of more experienced players to improve at the game.

Related Work, Background Material and Data Source

Related Work

One paper that provided some motivation for this project is “Rise of the Bots: Bot Prevalence and Its Impact on Match Outcomes in League of Legends”[1] This paper investigates the effects bots have on the matches and finds that bots are prevalent in all levels of the game, but are most common in the early levels. We wanted to see if there was a significant difference between a normal player and a bot’s actions. This paper is aimed at analyzing current games of league, with no access to the mouse or keyboard input to detect them, instead relying on aspects of the account such as the selected champion and items/masteries used.

In designing the parallel coordinates visualization we wanted to understand which features may be the most beneficial to this data. To do this we consulted the brushing paper from earlier in this semester[2]. While the main contribution of the paper, angular brushes, were not applicable to our visualization many of the other approaches they explored fit our needs. The paper also highlighted that the key for parallel coordinates is interactivity as that is where parallel coordinates perform best. One of the key features this paper highlighted to us was being certain the user can re-order their axes, a feature we previously had not considered as it seemed superfluous.

Data Source

To gather data Marshall built a python application using Pynput that logs the user’s mouse clicks and keystrokes over the course of a game. We then distributed it to a few friends to generate data from the games they played. Unfortunately we didn’t quite get as much data as would be ideal but it was functional for our purposes. It is also difficult to analyze trends in playstyle over the course of only a few weeks.

---

[1] C. S. Lee and I. Ramler, "Rise of the bots: Bot prevalence and its impact on match outcomes in league of Legends," Network and Systems Support for Games (NetGames), 2015 International Workshop on, Zagreb, 2015, pp. 1-6. doi: 10.1109/NetGames.2015.7382992

[2] H. Hauser, F. Ledermann and H. Doleisch, "Angular brushing of extended parallel coordinates," *Information Visualization, 2002. INFOVIS 2002. IEEE Symposium on*, 2002, pp. 127-130.

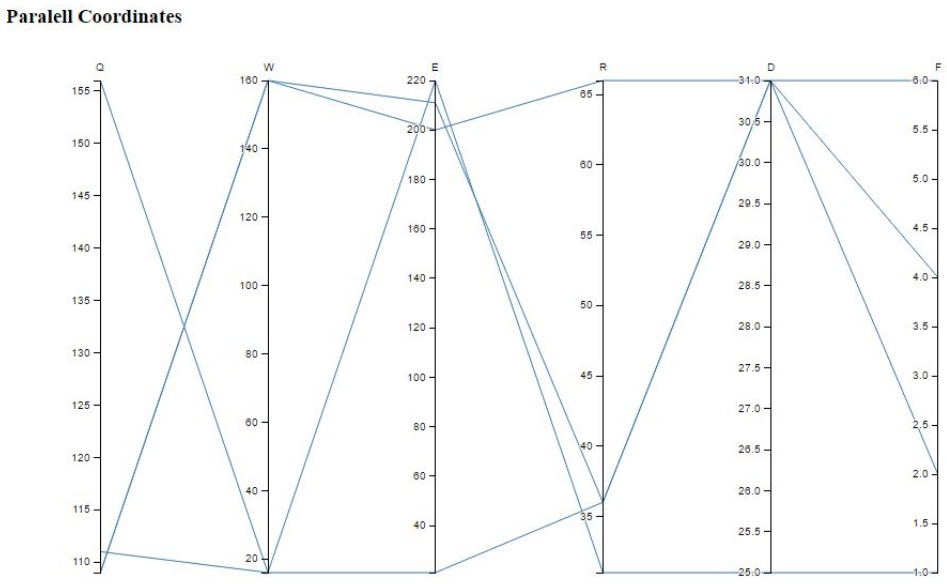
Visualization Design Evolution

The initial idea for our project was to create a set of visualizations from our data that players could use to discover trends in their play. To achieve this we brainstormed possible visualizations before settling on the most achievable group that we felt would be useful. Each visualization went through its own design iterations along the way. Because the data we gathered is very flexible there were endless visualization options to chose from and in the end we decided on three, Parallel Coordinates, 3D Scatter Plots, and Streamgraphs, each of which are explained in detail in the following sections.

Parallel Coordinates

The goal of the parallel coordinates chart is to allow players to explore trends in their play. In particular which abilities they use more frequently on champions or sets of champions. The first design choice we made was around which axes to put on the chart. We chose to put key presses as the keys represent various different abilities which could easily be explored using parallel coordinates. We did not put actions per minute (APM) or other statistics on the chart to keep it simple and focused.

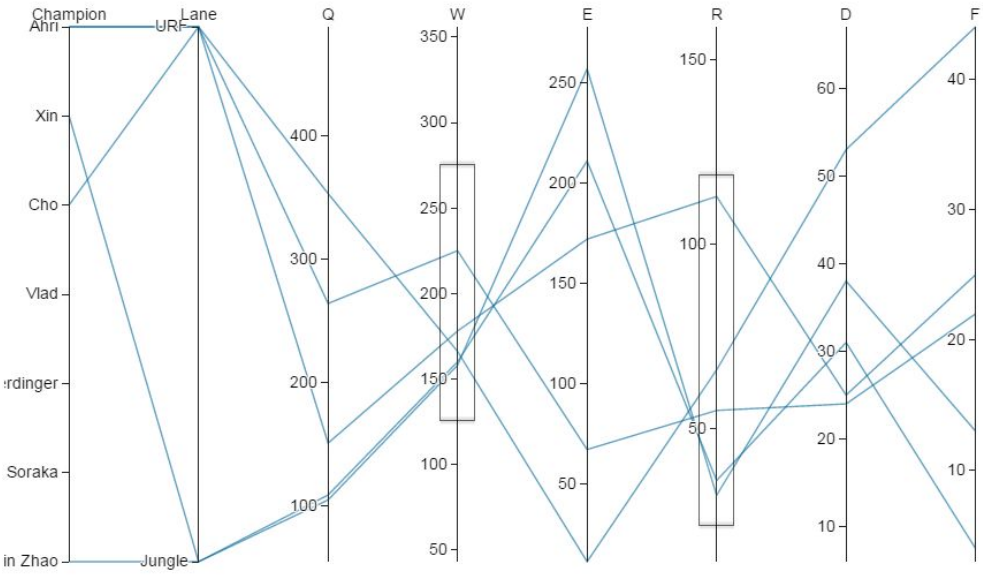
The first challenge was choosing how to get the chart functional. At first we used a raw D3.js implementation based on an implementation by Jason Davies[3]. It had a lot of features that were desirable but it lacked an easy mechanism to color the chart. However it served as a good proof of concept.



The first implementation of parallel coordinates with fake data

[3]"Parallel Coordinates." Popular Blocks. N.p., 5 Apr. 2016. Web. 05 May 2016. <http://bl.ocks.org/jasondavies/1341281>.

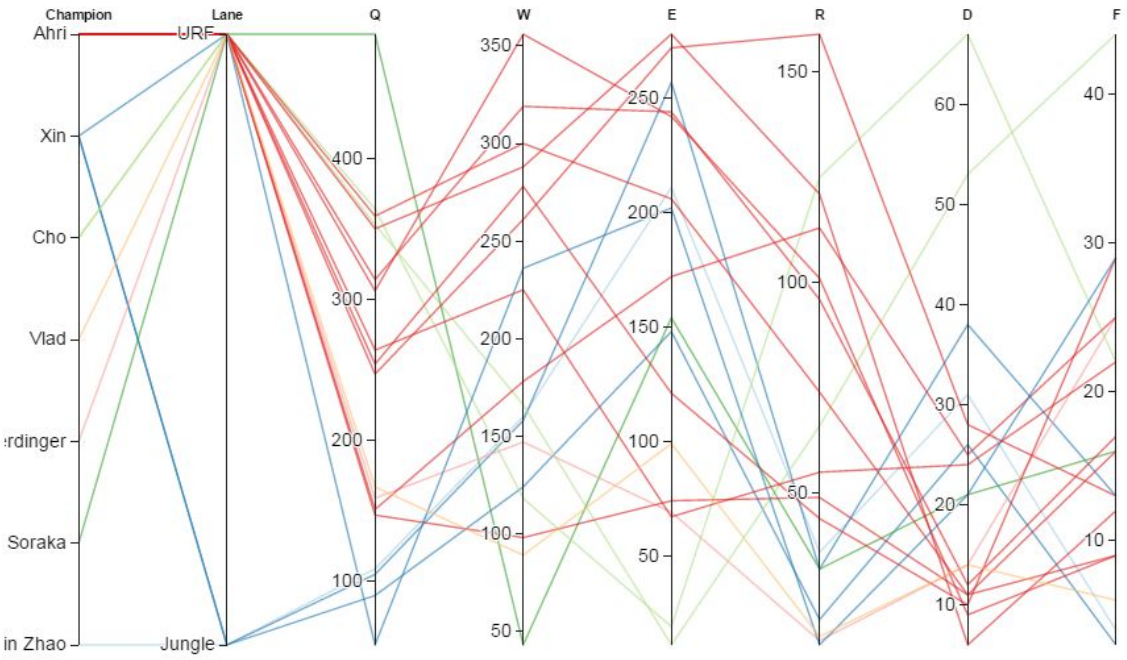
In order to facilitate improvements for the parallel coordinates we switched to a d3 library for parallel coordinates[4]. This library had some issues and we initially lost some functionality, like shadows behind brushing, but using the features we were able to regain that functionality and more. It was also at this point we added two more axes, champion and lane, to make the data more explorable.



This is the prototype we used for in-class demo day

Going into demo day we knew that we would need to address color so a big goal was discovering where color would be most useful. The feedback we received indicated that it would be most useful to have color to categorize the lines by either champion or lane. A few users indicated they may want a coloring based on how much a certain key was pressed so we created a scale for the most powerful ability on every champion which is bound to the R key. Once we implemented these three colorings we needed a way to switch between them so we added a control bar to the bottom of the visualization. This method also preserves brushes on color switching so multiple colorings can be used to explore a brushing.

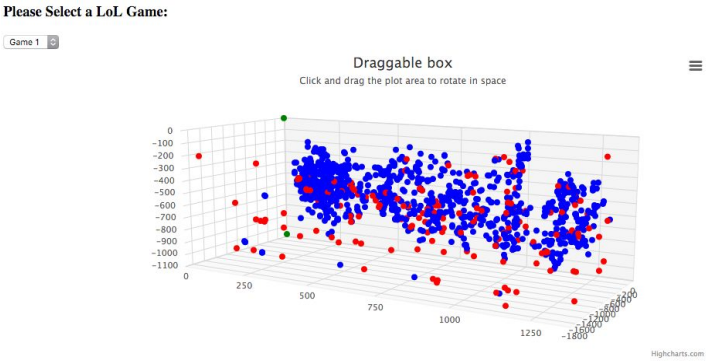
[4] "Parallel Coordinates (0.7.0)." Parallel Coordinates. N.p., n.d. Web. 05 May 2016. <https://syntagmatic.github.io/parallel-coordinates/>.



Final Visualization with champion based coloring

3D Scatter Plot

The largest dataset we gathered from games is the mouse action log, a recording of time and position on the screen for every mouse click. The scatter plot would allow the user to visualize where on the screen they click the most (the X-Y plane) or how often they click and if there are any large gaps (the Y-Z plane). We had to differentiate between left and right clicks because they have very different uses in the game so we have each type be a different color.



The first challenge was getting an operational scatter plot that the user could interact with by clicking and dragging to rotate the plot. This was accomplished using a third party library for Javascript called HighCharts[5]. This proved to be a viable solution for smaller data sets but once we imported the average 4,000 point file the plot became sluggish and unusable from a user standpoint.

We went into the demo day with the HighCharts plot because our time had been spent building and perfecting the data collection scripts. The plot would give users an idea of what we were trying to accomplish and the feedback represented that. Users were able to overlook the

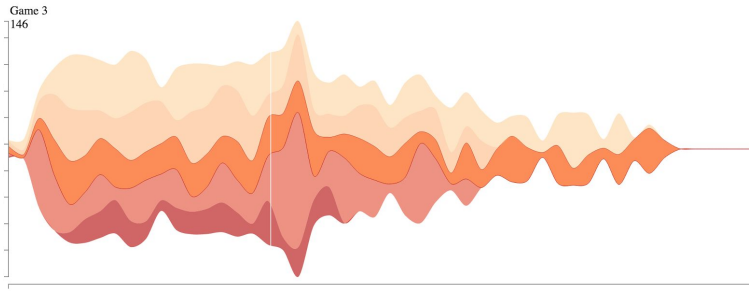
[5] "3D scatter chart." HighCharts. N.p., n.d. Fri. 08 April 2016. <http://www.highcharts.com/demo/3d-scatter-draggable>

laggy behavior (even though fixing it was a common suggestion) and give advice on how to improve it.

Our solution to the poor performance was to change libraries. We looked for solutions using D3 and Three.JS but eventually settled on Plotly[6], a high-level plotting library build over D3. This was a huge improvement over Highchart in both performance and number of lines of code needed.

Streamgraph

One fun way to visualize data is to use a streamgraph, it doesn't provide the in depth detail of other visualizations but it allows the user to differentiate between datasets quickly. The data we collected would not provide and useful information in a stream graph so we created a parser that combined mouse and keyboard actions into a single measurement of Actions Per Minute (APM). We were then able to graph the APM values so there would be a single point for every minute of the game. At first we used D3[7] to implement our streamgraph. There was not a whole lot of interactivity between the graph and the user which we wanted to improve upon.



First implementation of our streamgraph

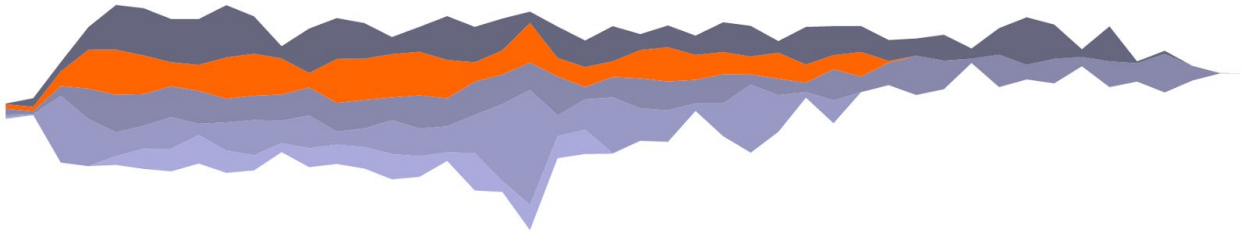
After some searching we found a second implementation[8] that provides more interaction, allowing the user to highlight streams and switch between different datasets. We hope to find a medium between the two graphs, keeping the smoothness of the first one but allowing for the interactivity from the second.

---

[6] "3D Scatter Plots in plotly.js" Plotly. N.p., n.d. Tue. 03 May 2016. <https://plot.ly/javascript/3d-scatter-plots/>

[7] "D3 Interactive Streamgraph" William Turman. N.p., n.d. Tue. 08 March 2016. <http://bl.ocks.org/WillTurman/4631136>

[8] "Interactive Streamgraph D3" Ger Hobbelt. N.p., n.d. Thu. 28 April 2016. <http://bl.ocks.org/GerHobbelt/3480186>



Second implementation of our stream graph with game 4 highlighted in orange

Feedback

For feedback our team split up with Marshall getting feedback on the Streamgraph and 3D scatter plot and Nathan gathering feedback on parallel coordinates.

Parallel Coordinates

Nathan talked to six different users about the parallel coordinates visualization. Only two of the users fit the target demographic of League of Legends players but all the feedback was valuable as the others were able to use their general knowledge of visualizations to find problems and suggest possible improvements.

The first issue the users encountered was understanding what was being visualized. This led Nathan to the conclusion that a description of the visualization was necessary to help users understand what data they are interacting with. In addition there needed to be instructions for what the user can do to interact with the visualization so they are aware of their options. Nathan asked the users where they would use color in the visualization. Half of the users suggested coloring to categorize the champions into classes like mages or tanks and the other half would have liked color to categorize the data by key presses in a type of clustering. One user who was unfamiliar with league noted the way he would approach the data was by looking for clusters so the visualization should do its best to facilitate the finding of those groupings.The users also found a few bugs with the visualization including a few unbrushable areas of the chart.We took this feedback into account while finishing up development and the results drastically improved the parallel coordinates overall quality.

3D Scatter Plot

Marshall talked to five individuals, of which three were familiar with league, two of which are players. Feedback on the scatterplot was limited, with the most suggested item to be a view snap feature that will orient the camera to one of three views, the X-Y plane for location of clicks, the Y-Z plane for activity, or a more isometric view. It was also suggested to use a different library which we were already planning on doing. The user's were unable to switch between games, and they suggested having this feature along with some way to compare between graphs. A heatmap could fix this issue because the heatmap would show the density

of clicks over the screen which the user could use to compare with his own clicks. For example, if the user is lacking a large number of clicks over the minimap but someone of a higher skill level has a higher density of clicks in that region, the user might infer that they should use the minimap more often.

Streamgraph

The general consensus was to have some sort of filter with the streamgraph that would allow the user to sort by champion or lane position. Our new implementation of the graph will be able to support this feature in the future. One individual suggested having the most recent game played be overlaid with average games of either lane position or champion. One individual suggested having the streams overlap each other instead of stacking to better compare between games. A different individual suggested to change the time range of the graph because the tail end of the graph can be uneventful. This could be done in a slider bar at the bottom or even dragging a rectangle in the graph to zoom in on a certain time.

Core features/ and technical details

Core Features

Parallel Coordinates

The parallel coordinates are highly interactive making it more of an exploration tool than a visualization of any conclusion. To achieve this it is as customizable as possible. All of the axes are reorderable so the user can look into a side by side relationship of any two variables or an odd ordering that they want to observe. In addition every axis can be brushed at the same time so that the graph can be narrowed down appropriately. To keep the user aware of the wider implications of the data there are shadows of the un-selected data in the background when a subset is being highlighted by a brush. There are 3 color modes available to the user, one that categorizes data by champion, one that categorizes by type and a third that is a gradient from blue to red based on how many key presses of R there are for that data.

3D Scatter Plot

The scatterplot has limited interactivity, the user is able to drag and rotate the plot, click a button to reset the view, and switch to a different game. Its primary use was to help the user observe their habits throughout the entire game. Overlaying multiple games would produce too many data points and the plot would become too dense and its usefulness would decrease. The left and right click events are each represented as a single point, with blue being right click and grey representing left clicks. The largest issue we faced was ensuring the plot had little to no lag while the user was interacting with it which we fixed by switching libraries.

Streamgraph

This is very difficult to implement correctly with all of the wanted features. Currently it just displays a stacked streamgraph of all games played and their APM. Eventually we want to add the filter functionality to select between different lane positions, champions, or skill level. The most challenging part of the streamgraph is manipulating the data and organizing it by game and time.

Technical Details

Parallel Coordinates

As discussed in the design evolution the implementation of the parallel coordinates uses a library on top of D3.js. This simplified the creation of the parallel coordinates but added some challenges. The library had some issues with label placement so it was necessary to use some extra D3 to place them appropriately. The biggest problem with the library was it does not feature dynamic sizing and that feature is beyond our technical abilities with D3.js Other than those few issues the library entirely simplified the process. Making color sets was straightforward after viewing a few examples and reading their documentation. The brushing and reorderable axes were easy options to turn on and the shadows were very customizable. Overall the implementation offered the right level of complexity and customization without becoming bogged down with excessively confusing options.

The other part of the implementation was creating a parser in Python for the raw data files. Our data is stored with each game being a folder. Inside that folder is a ReadMe text file with basic information about the game including champion and lane. Then there are two data files with key and mouse logs respectively. The parser creates a header line with a category then asks to be pointed at a game directory. It works through that directory to create a data line then adds it to the csv. This creates a properly formatted csv for the parallel coordinates library.

Format for the keyboard input:

```
Time - key
13.53817 - q
```

Formated csv:

```
Champion,Lane,Q,W,E,R,D,F
Xin Zhao,Jungle,109,160,211,36,31,4
```

3D Scatter Plot

Our 3D scatter plot uses the Plotly library to create the graph. A lot of the work is done by plotly so there is relatively little work to make the scatter plot. After parsing the data we

import it using the d3.csv built in method. We then add the x, y, and z points into respective arrays that are within one of two objects, one containing data for left clicks and the other for right clicks. After that is finished the only thing left is to format the layout of the scatter plot.

The parser for the scatter plot takes in the mouse input, which is of the following format:

```
Time - X, Y - Button
2.03314 - 870,1079 - Button.left
```

We then parse it into a csv file with a similar format. Time becomes the z coordinate:

```
z,y,x,button
2.033,-1079,870,left
```

Streamgraph

For our streamgraph to work we had to write two different parsers in Python to generate the proper csv file. Our first file parsed through each game folder individually creating an APM csv with the format

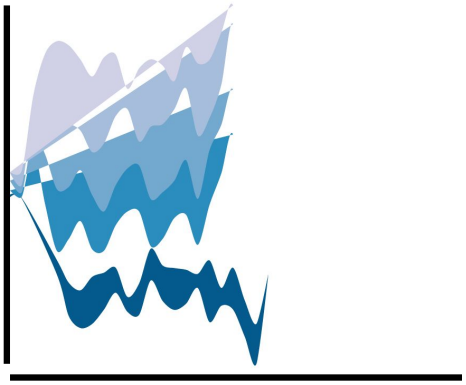
```
time,actions
0,60
```

Once this was done for all games, the second parser was run which compiled all of the individual APM csv files into one csv which is then read by the streamgraph code.

```
game,time,actions
1,0,60
```

The streamgraph code takes this data and creates one large flat array with each index containing the game, time, and number of actions. We use the built in functions Nest and Map to build a hierarchical data structure with the game at the highest level, time for each game at the middle level, and number of actions within each time.

One issue we encountered when creating the streamgraph is that the data structure within each game must have the same number of indices, if one game has a value for the 37th minute, they all have to have that value. Otherwise we got a graph resembling the image to the right. To fix this issue we had a default value of 0 if there was no value for a specific minute.



Conclusions

We were unable to accept or reject our hypothesis because we were unable to collect enough data. We did observe patterns in the location the user clicks on the screen. There is a cluster of left clicks where the item bar is, as well as left clicks slightly above the skill bar which would be from leveling up skills. There is also a group of both left and right clicks centered

around the minimap. The clicks also follow the diagonal shape of the arena, bottom left to top right. We also saw some of the correlation we expected in the parallel coordinates with champions like Ahri, a mage, using her abilities more than Xin Zhao, a melee champion. Again there was too little data to draw real conclusions.

Division of Work

Nathan Berton

Nathan focused primarily on the creation of the parallel coordinates graph and building parsers to work with the parallel coordinates library.

Marshall Shortledge

Marshall worked on the Python parsing and data collection scripts as well as the 3D scatter plot and the Streamgraph.

Submission Details:

Live Link: http://nberton.github.io/LeagueAnalyzer/

To run the visualizations locally just use a simple server pointed at the unzipped directory and access index.html. The visualizations are under their properly labeled folders and the data parsers are under dataCreation.