

Final Project Report - Fortnite StatVis

Garret Premo



Introduction

Fortnite is a game that is currently taking the world by storm. With over 45 million players and 3 million concurrent users, it has a massive pool of data that is growing constantly. Amongst this mass of data, gameplay statistics will always be something of interest to the players. As a player myself, I know this to be especially true as it's a common area of comparison amongst friends who also play Fortnite. There are some websites that allow you to view statistics, however, none of them are very interactive nor do they allow comparisons between players. That was the main source of motivation for this project and, with it, provided a goal that would be appreciated by the Fortnite community: an interactive statistic comparison for the every-day Fortnite player.

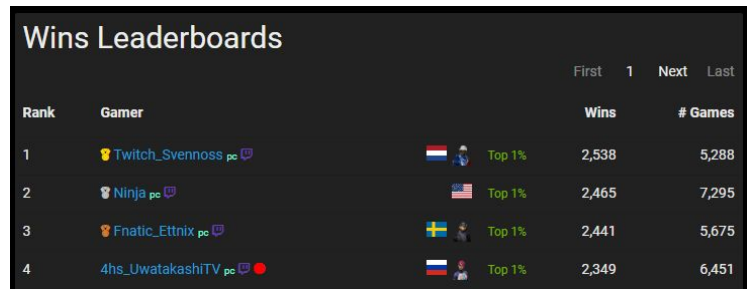
My original research question was to find whether or not I could map these statistics against location points on the Fortnite Map (see Figure 1). My hypothesis, at the time, was

that it could be done, provided location data could be found via an online API. After searching countless sites and forum posts, the conclusion was made that there was no such api. This caused a large shift in the overall focus of this project.

Data Collection

The main source of data for this project comes from it's main area of comparison, and is the current standard for players viewing statistics in Fortnite. It is a website called `fortnitetracker`, which uses Fortnite api calls to get the statistics for every player in every match that is played. Comparisons can be made between players, however, it requires multiple pages to be open, as there is no specific functionality for this.

Collecting the data was done in two main stages. The first stage consisted of gathering a list of usernames from the leaderboard section of `fortnite-tracker`. This was to be done for each of the individual PC, Xbox One, and Playstation 4 leaderboards. To do this, a web scraping application called ParseHub was used. The application would go through an arbitrary number of pages on the leaderboard and collect each player's username in order of rank.



Rank	Gamer	Wins	# Games
1	Twitch_Svenness pc	2,538	5,288
2	Ninja pc	2,465	7,295
3	Fnatic_Ettnix pc	2,441	5,675
4	4hs_UwataKashiTV pc	2,349	6,451

Figure 1: a small screenshot `fortnitetracker`'s leaderboard

The second stage was running each username through `fortnitetracker`'s public API. This API provided many useful statistics for each player, including statistics for eliminations, deaths, wins, matches played, win %, elimination:death ratio, eliminations/minute, average match time, and more. Furthermore, these statistics we split up into subsections of lifetime statistics, season statistics, solo games, duo games, and squad games. Overall, using this API as the main data source provided very thorough data. The biggest difficulty was a simple time constraint, as the API could be called a max of once every two seconds. The time constraint did limit the scope of my project, however. The original intention was to gather statistics for as many players as



Figure 2: a screenshot `fortnitetracker`'s public api

possible. With a player base of 45 million, it could have taken weeks to get data for every player, even using multiple API keys.

Visualization Design Evolution

The initial plan for this visualization was to overlay statistics onto the actual Fortnite Map (**Figure 3**). As the project progressed, it was becoming apparent that getting the necessary location data would not be a possibility, without an extremely biased dataset e.g. using *manually* gathered data from my own games and friends' games.

Gathering this data manually would mean jotting down the exact landing location at the beginning of each game, as well as other data points of interest throughout the game. This could have included: time survived; damage taken; damage dealt; etc.



Figure 3: the map of Fortnite: Battle Royale

Since the original plan was not seeming feasible to complete in the given time frame, this is where the project took a massive pivot towards visualizing and comparing statistics. Part of this process involved deciding how to visualize this data. After searching for different ways to show multi-dimensional datasets, some experimentation was done with a scatter plot matrix in d3. The idea in this visualization was to show comparisons for statistics gathered with the ability to brush over points of interest.

The first test of this visualization (**Figure 4**) shows 5 dimensions of data. Wins, eliminations, elimination:death ratio, and score are displayed against each other as scatter plots in the matrix. Each point is an individual player, and the fill color for each point shows the platform of the user. Green being PC, Xbox being orange, Playstation 4 being blue. Each point on one graph corresponds to one point on every other graph. The labeling for the test was not included, because the data used was entirely fake to produce a proof of concept (and sanity check).



Figure 4: The first experimental implementation of the visualization showing false data as a proof of concept.

This type of matrix made brushing a fairly simple process. Brushing is a technique that consists of simply clicking and dragging an area over any points in one graph. This causes those points to be highlighted, while graying out all points that are not selected. The purpose of brushing is to provide an intuitive, interactive way to identify the relationships between points across the entire matrix.

Most of the effort placed into the interactive part of this visualization was the brushing, as it provides the most clarity and understanding for larger datasets. Since the scope of this project was mainly large datasets, this became incredibly important, as the more intricate matrices were much harder to understand (**Figure 5**). Without the ability of brushing, it would be incredibly difficult to tell how points in some graphs correspond to other graphs.

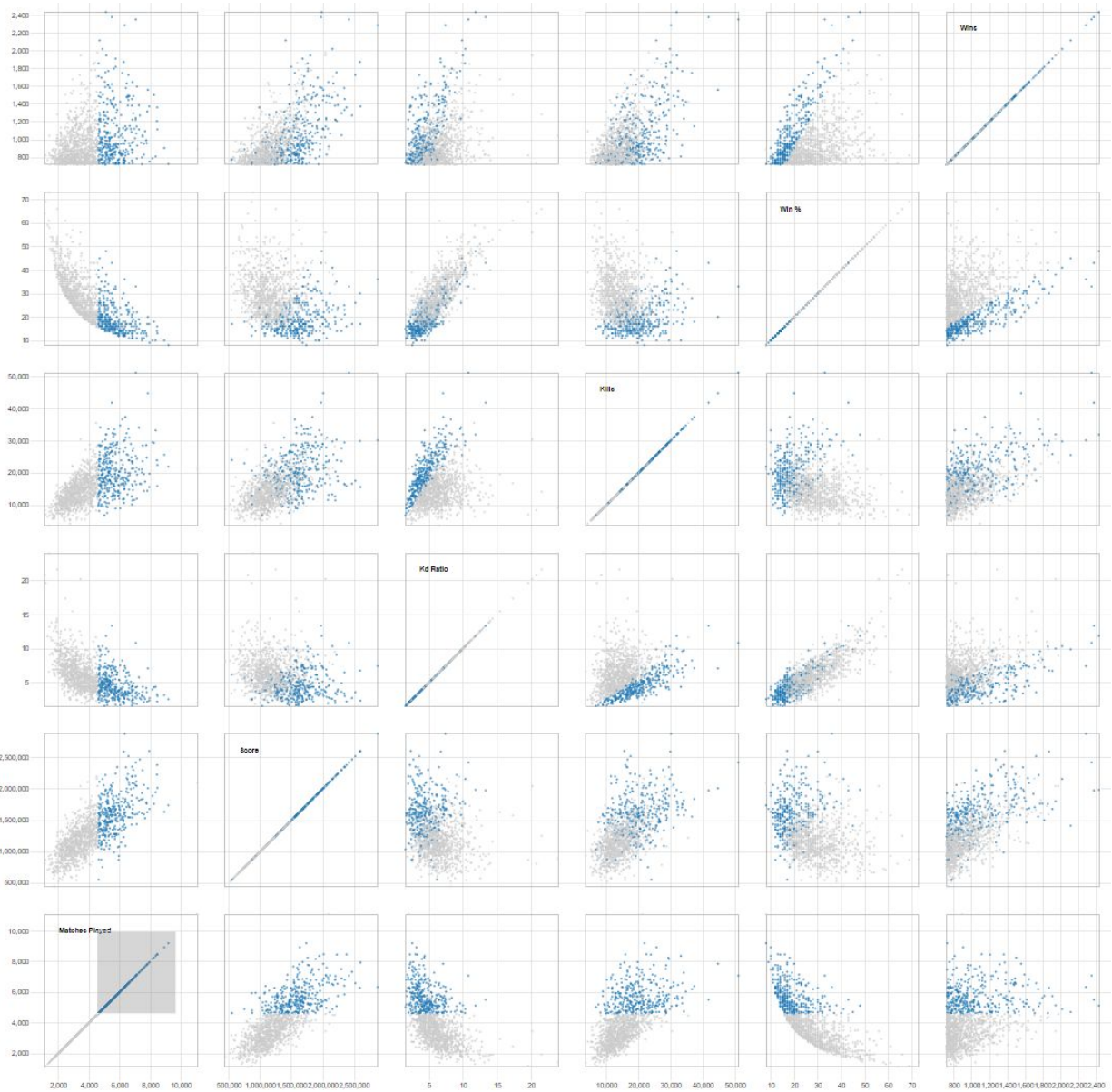


Figure 5: The Final (complete) implementation of the visualization showing A 6x6 scatterplot matrix with ~1250 top PC players (globally). The bottom left-hand corner (Total Matches Played) is brushed over, and all corresponding points are highlighted blue. The dimensions of data (from upper-right to bottom-left) are: wins; win%; kills; kill/death ratio; score; matches played.

Feedback

The area of feedback that I took the most from were my peers who actively played Fortnite. A lot of friends were intrigued by the original map overlay idea, however, pessimistic. The

newer iterations of the project brought in a lot of new feedback. Most of which was adding more customizability to the statistics that are being shown in the matrix. One of the biggest points of feedback was adding dropdown menus to select certain data points for each row and column. This would be the best way of interactively comparing statistics on a broad level, as it gives the user direct control of the statistics. Other points of feedback included choosing the size of the dataset, toggling which platform is displayed, zooming into portions of interest, draggable rows/columns, and, of course, support for datasets by specific region.

The feedback I gained was directly from a subset of my main target audience, and was incredibly constructive. The potential for features that could be implemented was near endless.

Features

The main layout of this project was a scatter plot matrix. The matrix was made using d3, and utilized source code found in the d3 examples section. This matrix could accept any number of dimensions of data via a csv file. The csv file was formatted by a python script which parsed through the raw JSON data received by the fortnitracker API.

The main feature of this visualization was brushing. It could be applied to any graph in the matrix, and would in turn update all other graphs by highlighting the points corresponding to the points selected. After a brush area has been established, it could be freely dragged within the graph to highlight other points without creating a new brushing. A brushing could be removed by simply clicking anywhere on the graph.

The main challenges faced in this project was implementing new ideas for features that were gained through feedback. The nature of d3 is not one that allows much room for quick implementation. Trying to make a legend, dropdown menus for rows/columns, and changing color scheme were examples of some challenges that were faced that could not be solved.

References

1. fortniteTracker fortnitetracker.com/leaderboards
2. fortniteTracker API <https://fortnitetracker.com/site-api>
3. Scatterplot Matrix D3 example <https://bl.ocks.org/mbostock/4063663>