# INFO 3300 Project 2 Write Up

Karen Jan (kj258), Jean-Charles Turban Davila (jt665), Samuel Williams (siw25).

**DESCRIPTION**

Our dataset, final_dataset.csv, is a combination of the super_hero_powers.csv and SuperheroDataset.csv, pulled from the kaggle competitions, https://www.kaggle.com/claudiodavi/superhero-set and https://www.kaggle.com/thec03u5/complete-superhero-dataset, respectively. super_hero_powers.csv has one column dedicated to the hero's name and 166 columns dedicated to a specific superpower (e.g: intelligence, invisibility, omnipotence). The values of those 166 columns are True/False. SuperheroDataset.csv has additional statistics about heroes such as Combat, Power, Strength, Durability, Intelligence, Speed, Height and Weight. With the exception of height and weight, all the values are between 0-100. We merged both data sets on the name of the heroes to have more data points for each hero (joined_data.csv).

Several non-Marvel, non-DC, and non-Dark Horse Comic characters were missing height and weight values, so we filled those in with the medians of those columns. To grab those values, we created a temporary dataset (tmp.csv) that only had hero names, weight, and height so that we could more easily use the pandas function, pd.describe(). Additionally, for the purpose of our brackets, we reassigned any character not part of Marvel, DC, or Dark Horse Comics to a new company, Miscellaneous, to cut down on the number of organizations. For example, Harry Potter, Katniss Everdeen, and Yoda all belong to

Miscellaneous. The height and weight were also stripped to just contain the cm and kg values. Furthermore, we added a tier column for each character that calculates the tier of character based on which specific superpowers they have from the true and false values using a weight and ranking system for each of the 166 abilities.

The intermediary python scripts and datasets can be found in the data_preprocessing folder.

**OVERVIEW**

The rationale behind our design was inspired heavily from the march madness brackets visualization that already exists. One of these sources of inspiration came from march madness brackets for basketball games(found at https://www.ncaa.com/brackets/basketball-men/d1). We thought of ways that we could represent this same concept, and how we can mimic the competitive theme. Because people naturally argue about which heroes/villains are the strongest/weakest, we decided to have popular/famous characters fight each other March-Madness-style until a winner is crowned.  March Madness brackets were the perfect way to represent the simulation since the brackets are commonly used to convey competitions and winners-losers scenarios.

Our visualization requires user interaction--the user has to click on the right button to display the simulation. This is one trade-off as opposed to having the visualization be automatically generated and then having the user interact with it. So, if a user does not interact with the button the visualization will remain incomplete. Therefore we had to make the button clicking interactivity very obvious.

To display the data, 16th characters are pulled from our dataset in the first round of the simulations, and those characters are passed into an algorithm that uses their characteristics such as defense and accuracy to determine who would win each round of the simulation. We designed the algorithm to pre-determine who wins for each round because it would save time having to calculate per round, and also this helped us to perform the simulation animation/transitions faster. The data of winners for each round is displayed within the brackets of each round using text elements.

Some design choices we made in displaying this data was that winners of a round would be bolded and red. We decided to do this because it will allow viewers to easily conclude who the winner is and can easily follow the path of winners until the final round. Another design choice was to display the winners and round one by one using transitions and animation instead of rendering the data immediately because we wanted the user to follow and anticipate the outcome of the round.

Finally in the styling our visualization we chose the background image because it coincided with the competitive theme of our visualization. The overall font and colors were chosen to convey a comic-like feel and lighthearted tone.

**INTERACTIVE ELEMENTS AND THEIR DESIGN RATIONALE.**

The main purpose of the interactive elements within our visualization is to give our users more control over the scenarios that are generated programmatically. When we initially came up our idea, we wanted to have a way that a user can first generate a battle simulation between fantasy superhero villains and other

miscellaneous characters, while giving them multiple options to influence its outcome. Our process to choose the interactions started off with the brainstorming the general concept of our visualization, and writing down areas of interactivity. Then we started by making a non-interactive visualization then adding the various elements of interactivity that we discussed. We tested the various interactive elements to make sure they serve a purpose and aided with the user experience, otherwise, we would remove that element. Within our project there are multiple elements that a user can interact with:

**Buttons**: The fight button on our visualization plays a simulation by randomly generating characters based on the user's selected parameters, and then following the battle overtime until a winner is defined. We choose to include the fight button instead of having the simulation automatically generated because we wanted the user easily generated multiple scenarios just by clicking a button. We also included a fast forward and rewind button to allow the user to manage the pace of the simulation. We choose to include the fast forward and rewind buttons to accommodate both fast-paced users that would not want to wait for the simulation to play as well as users that would enjoy viewing the overall process.

**Inputs**: The inputs on the page allows the user to input weights to Damage, Health, Defense and Accuracy statistics to see how the different weights affect fights and who wins that battle. That way, if a user believes that large height and weights are actually detrimental, they can modify the Health weight to reflect their beliefs and see who would win based on their definition of what makes a character powerful. We decided to use inputs of type numbers to ensure that users would give meaningful weights (e.g: inputs won't allow "fire" as a weight).

**Checkboxes & dropdown menu**: These dropdown menus and checkboxes are other forms of interactivity that allows the user to apply filters to the generated character. We included the checkboxes so

a user can be selective of what traits they would like for the simulation to have. Building on that further, checkboxes were chosen in lieu of radio buttons because we wanted the user to be able to deselect filters, which is not a functionality allowed by radio buttons. The dropdown menus also help with allowing the user to have more filtering power. The dropdown menu also gives the user more options outside just the Marvel Universe, so if they would like DC characters, they can apply this filter.

We made these interactive elements discoverable by moving them to the top of the page. We wanted to first thing our user sees is the interactive elements and that they are aware of these controls. The design of these buttons, input, checkboxes and dropdown menus are what the user would typically see on the web or on their computers so they fit the mental models that they already have. As a result, a user could easily deduce that they afford clicking, or they can enter a value.

**THE STORY**

Our visualization does the impossible -- or highly improbable. It does what every hardcore comic book/pop culture fanatic wants to see: which characters are the strongest of their respective universe and which characters are the strongest across many different universes. Thus the algorithm compares all traits that can be paralleled between characters to create a single metric from which to see who truly reigns supreme in the fictional world.

What is surprising about our visualization is that certain characters have very high chances of winning the entire bracket. Some of those characters are not surprising, like Thanos, but others were, like Groot and King Kong. What weighed heavily in the respective favors were overall size and mass. Being able to offset this using the variable weights created a more level playing field that would visualize how other

components would impact. For example, when health and tier became negligible massive, God-like heroes lost their considerate advantage, letting ordinary folk, like Katniss Everdeen, win it all.

We want to convey to our viewers that some characters that get overlooked or are not part of the Marvel Cinematic Universe or DC Cinematic Universe are actually quite strong and that you should not discount them. Examples of these players include: Aquaman and Beyonder. Furthermore, while females may not have the same strength or health statistics, they should not be discounted. In fact, Captain Marvel is a character with very high chances of winning based on default weights.

**TEAM CONTRIBUTIONS:**

**Samuel Williams**: I initially worked on the front-end portion of the visualization which included rendering the brackets. Afterwards I worked on animating the simulation which entailed incorporating the text transitions, and coloring. After more of our visualization came together I work on styling and adding more to the animation and transitions.

**Jean**: I worked at the beginning of a lot of the sketch work and did some preliminary implementation for a different idea we had earlier. I was very interested in the design and back end calculation so I ended sketching and make mock UI's. After we agreed what the visualization should look like I worked on first creating the algorithm that determines the winners and then implementing it javascript. After the algorithm was implemented in the front end we had I also worked on making the bracket more aesthetic and responsive to changes in the SVG height and width as well as the box heights and box widths.

**Karen Jan**: I wrote python scripts and csv files to combine our two datasets together and format it so that it was more easily usable (i.e the data preprocessing stage). After that, I worked on changing functionality based on user data. This included: displaying the interactive options (drop down menus and buttons), ensuring that users can only select valid combinations of filters (e.g: a hero can't be both female and male at the same time), filtering the data based on user input, choosing 32 characters from the filtered data, displaying the 32 characters (and removing the previous 32 if needed), altering the algorithm to use user defined weights, changing the transition speed, and limiting just how fast/slow a user can make it by making the buttons disappear). I spent roughly 48 hours on the project.

**BREAKDOWN:**

The part of the project that took the most amount of time was filtering the data based on the user input. This is because this "one" step required many substeps to be accomplished. Firstly, we had to grab the user's inputs. Secondly, we had to ensure that the filter combinations on the left side of the versus was kept independently of the filter combinations of the right side (function check_sat()). We then had to select 32 unique random characters from the filtered list. However, we also had to handle the case where the size of the filtered list was not large enough to grab 32 unique random characters. We fixed this problem by randomly grabbing the remaining characters from the total dataset, provided that that chosen character was already not in the list. Because of the way it was implemented, we also had to handle the case where the filtered dataset from the left and right filters were exactly the same and their combined length was exactly equal to 32. Overall, this part of the project took the most amount of time because it had many moving parts within it with a lot of places for small bugs to occur (which they did), so a decent amount of time was spent writing the function and a lot of time was spent testing to discover and fix bugs (like infinite loops).