The Small Body Map for the Average Person

Colm Lang
University of San Francisco
San Francisco, CA 94117
cplang@dons.usfca.edu

***Abstract*—With this comprehensive Small Body Map for the Average Person (SBMAP), I provide a digestible, holistic representation of our solar system's many small bodies without sacrificing detail or accuracy. I have successfully created a lightweight interactive visualization for the web by prioritizing data-processing and sampling methods. Namely, with the use of Python's Pandas and Matplotlib along with an implementation of multivariate stratified sampling in JavaScript, I was able to reduce the data-set's bulk while maintaining it's characteristics and quirks. This ultimately allowed for a rich, *details on demand* approach for the Birds Eye View visualization and a more responsive Asteroid Dashboard.**

I. Introduction

In the recent years I have developed a keen interest in outer space and I want to bridge the gap between Astrophysicists and the rest of us. When data about outer space is collected it is often either displayed in ways that the average person cannot understand, or stripped of its intricacies to dumb down the content. It is my goal to provide a resource that keeps the depth of NASA's data without needing an advanced degree to understand.

A. Project Objectives

The goal of SBMAP is to produce a digestible holistic representation of our solar system's many small bodies that does not sacrifice detail or accuracy. To accomplish this, the project contains the following:

1) A **birds eye view** of our solar system and where most asteroids orbit
2) Analysis of the **distribution** of the small bodies' orbit distance
3) Analysis of small body **dimensions** and **size**

II. Related Work

The inspiration of this project came from the Small Body Database Visualizations by NASA/JPL. Therein, it had a visualization of every small body's location on 1/1/2018 by P. Chodas [2]. While this visualization was very nice, I thought that adding each body's full orbital path, rather than just showing a singular location, would be significant.

This created a necessity for large-scale aggregation, as one million ellipses would hardly be the best way to show a birds eye view of our solar system's many small bodies. It was at this stage that I came across Mike Bostock's D3 Hexbin Observable post [3].

When beginning to add transitions to the Asteroid Dashboard, I wanted to explore ways to interpolate the density plots and the contour plot. I imagined the best way to transition between states of contours would be to "lower" and "raise" the different contour sections to their new values. The best source I found to mimic this intuitive transition was done by Paul Murray. In an Observable post, he proposes the tweening of each point's weight between the previous state and the current state [5].

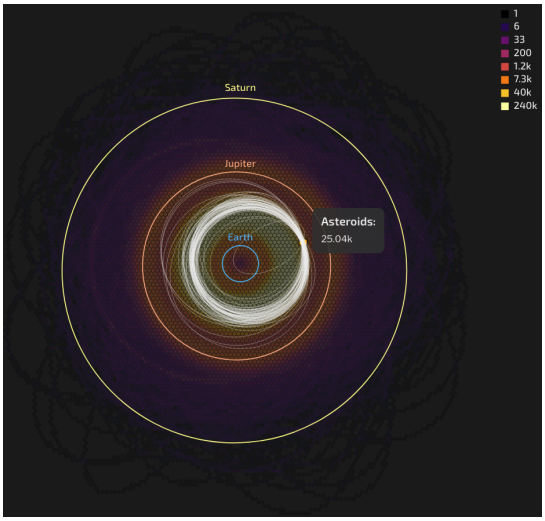

Fig. 1. The Birds Eye View visualization with a hovered cell revealing underlying ellipses.

III. Approach

Throughout the development process, I maintained the following strategy: Data first, Concise Data Visualization second, then UI/UX and Optimization last. This strategy ensured that the Data Visualization techniques I used would be aptly suited to the data I had. By thinking in this manner, I was able to reasonably tackle each project objective with limited need for big adjustments. A great example of this would be how I handled the data processing of the hexbin visualization.

A. Hexbins

The birds eye view visualization seen in figure 1 was ultimately developed using the process laid out above. I began with the data-processing by binning each ellipse into bins and aggregating the total count of intersections. I chose this approach to allow me to ship a much smaller file to the client. Rather than needing 1.2 million lines, I only needed to send the hexbins and their counts. To supplement user interaction,

I wanted to use the visualization technique that I originally planned: ellipses. Done in reasonably quantities, I found that displaying orbital ellipses when a user hovers on a hex was actually quite effective. Therefore, I knew I would need to implement the calculation of orbital intersections into the hexbin algorithm. Through a simple ID relationship, I added this feature to the data processing algorithm.

Here is a simple pseudo code demonstration of the data processing done in Python:

```
for body in dataset:
    # get all points based on
    # the body's orbital parameters
    points = get_ellipse_points(body)

    # get all bins based on the points
    bins = get_bins_by_points(points)

    for bin in bins:
        # if intersects
        if bin.count > 0:
            # add id to list
            bin.ids += body.id

    # update intersection totals
    total_bins += bins
```

Due to the intense run time for calculating orbital ellipses for 1.2 million small bodies, the algorithm takes roughly 75 hours to complete synchronously. By using Python's multiprocessing, I reduced the algorithm run time by 92% (down to 6 hours).

The user hover interaction was very fast and crisp on Opera but during the feedback session for the Beta Release, my group members brought browser issues to my attention. On Chrome, hover interactions were extremely slow and clunky but would seem to speed up after the first few minutes or so.

B. The Asteroid Dashboard

The Asteroid Dashboard seen in Figure 2 displays a Kernel Density Estimation (KDE) plot of orbit distance, a KDE plot of diameter, and a contour plot of orbit eccentricity to orbit distance. This dashboard displays four different asteroid groups with the ability to toggle any/all of them for data exploration.

Therefore, not only does the plot need to draw quickly, it must update quickly as well. Here I handled state using React's useState hook to track the current filtered groups to display. While this was a natural choice due to the React environment, filtering each plot using this state was not so straightforward. Using the paradigm set forth in Towards Reusable Charts [14] I abstracted the contour plot and KDE plot into their own files. This allowed one to call the data function with new filtered data and each plot would only update the changed information, rather than redrawing completely. Within these files I was also able to implement transition logic for smooth transitions between state.

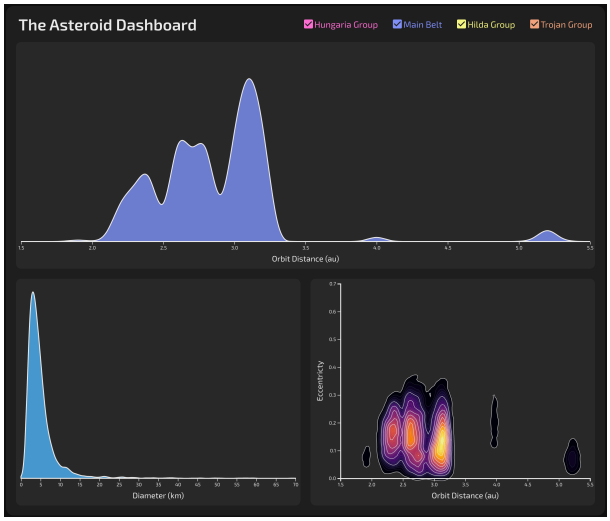

Fig. 2. The Asteroid Dashboard

All of this was only possible with a reasonably sized data set. I knew this before the visualization implementations described above, so I implemented sampling to reduce the data set from 140,000 entries to a more reasonable 10,000 entries. While a random sample would likely be fine, I wanted to ensure that the sample population accurately represented the distribution of all three variables (orbit distance, diameter, and eccentricity).

My solution resembles some sort of multivariate stratified sampling. I thought that by grouping the data points by orbit distance, then grouping each group by diameter, then again by eccentricity, one could be sure that each group is similar in all three dimensions. To produce a sample population of 10,000 points I simple take one in every 14 entries to reduce the population.

For the actual transitions in the dashboard I used interpolatePath [15] for the KDE plot to remove the unexpected behavior that often accompanies svg path transitions. As mentioned above, I used the weight tweening paradigm put forth by Paul Murray [5] to achieve contour transitions.

C. The Violin Plot

The implementation of the Violin Plot seen in Figure 3 was initially done through binning the bodies into 0.01au bins. This was effective yet limiting because the processing necessitated some hard-coding of values. I would render a curve that traces the exact count of each bin. After creating multivariate stratified sampling that retained the distributions of the data set, I could reuse the same data that is used for the dashboard for the Violin Plot and instead use a KDE line for the distribution.

IV. Results

The Hexbin Visualization very effectively shows a **birds eye view** of our solar system. I find that this coupled with the ability for details on demand makes this visualization very effective.

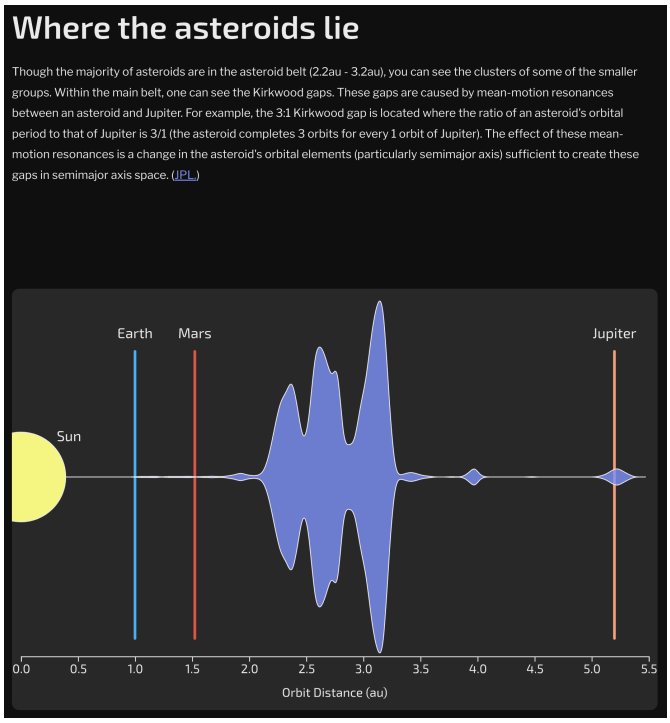

Fig. 3. The Violin Plot and its accompanying explanation.

The Violin Plot utilized the data sampling and KDE implemented for the Asteroid Dashboard lent nicely to visualizing the **distribution** of orbit distance.

The Asteroid Dashboard allows the user to investigate the **dimensions** and **size** of each group. The transitions between each group is done effectively and smoothly, allowing for seamless user interactions.

V. Discussion

Overall, I found the approach I took to be very promising. More specifically, I liked the strategy I used for development. Each stage I came to was very manageable and I think that additional feature implementation would be very reasonable using this strategy.

I think that the Towards Reusable Charts paradigm that I implemented in the dashboard was extremely effective. I would definitely like to adopt this approach to the rest of the project.

There was much to learn from this project but I found that data processing and the importance of chart re-usability served to be the most significant learning points. React's state management coupled very nicely with the reusable charts. Moreover, I learned so much about the delicate balance between React and D3's manipulation of the DOM.

VI. Future Work

While I found the visualizations to be quite effective, I would like to add supplementary context features. Additions like a glossary, more annotations, and more tooltips, would be very useful to make SBMAP as accessible for beginners as possible.

Full browser support would be ideal as Opera is not a very common browser. Ensuring that Chrome and Safari users have a fast interactive experience on the hexmap would make SBMAP's experience the best for much more users.

References

[1] Eleanor Lutz (Visualization), Nicholas LePan (Author), A Map of Every Object in our Solar System, https://www.visualcapitalist.com/mapping-every-object-in-our-solar-system/
[2] P. Chodas NASA/JPL, Inner Solar System, https://ssd.jpl.nasa.gov/diagrams/
[3] Mike Bostock, D3 Hexbin, https://observablehq.com/@d3/hexbin
[4] Mike Bostock, D3 Density Contours, https://observablehq.com/@d3/density-contours
[5] Paul Murray, Animated Contours, https://observablehq.com/@plmrry/animated-contours
[6] Natalia Andrienko, Gennady Andrienko, Visual analytics of movement: An overview of methods, tools and procedures, https://journals.sagepub.com/doi/pdf/10.1177/1473871612457601
[7] Yan Holtz, Data to Viz Ridgeline Plot, https://www.data-to-viz.com/graph/ridgeline.html
[8] React, https://reactjs.org
[9] D3, https://d3js.org
[10] Sass, https://sass-lang.com
[11] Pandas, https://pandas.pydata.org
[12] Matplotlib, https://matplotlib.org/stable/index.html
[13] Multiprocessing, https://docs.python.org/3/library/multiprocessing.html
[14] Mike Bostock, Towards Reusable Charts, https://bost.ocks.org/mike/chart/
[15] Peter Beshai, D3 Interpolate Path, https://github.com/pbeshai/d3-interpolate-path