**Building the bubble chart view**

To build the clustered bubble chart view, we had to learn about how to use force simulations. The first step was getting the circles to show up at all. We had always passed ‘cx’ and ‘cy’ values explicitly and never derived them from a simulation, so it was a new technique for us. We learned that in a simulation, if the nodes positions aren’t explicitly coded, then they we automatically be created. This was what the first example of creating the cluster looked like:

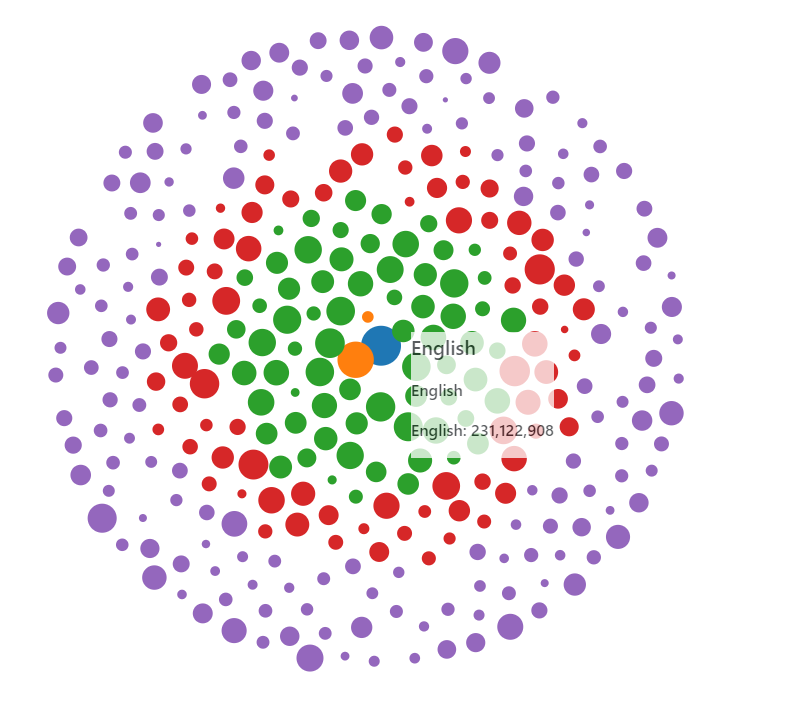


Figure 1: The default phyllotaxis arrangement

The guide that we used to help make this first visual was a combination of the tutorial for the course (<http://dataviscourse.net/tutorials/lectures/lecture-d3-layouts/>) and the d3 documentation on force layouts (<https://github.com/d3/d3-force/blob/master/README.md>). The problem with this first view was the circles overlapped. In addition to removing overlap, we wanted to cluster the circles instead of having them in rings. So we first got the collision detection working using this blocks page as reference: <https://bl.ocks.org/d3indepth/9d9f03a0016bc9df0f13b0d52978c02f>. See the next figure for a picture of what it looked like.

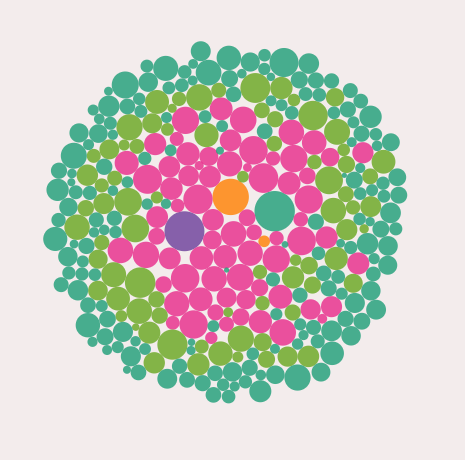


Figure 2: Collision detection was now working

Lastly, we wanted to get the clusters working. Before we got the clusters working, we changed from a log scale to an exponential scale since it was easier to control the sizing between the nodes. Linear scaling didn’t work since the smallest nodes were too small compared to the largest nodes, which is why we eventually decided to do exponential sizing as a happy medium between linear and log scales.

Part of getting the clusters working included using the force testing ground found here to fine-tune the simulation: <https://bl.ocks.org/steveharoz/8c3e2524079a8c440df60c1ab72b5d03>. The reference we used for clustering was found here: <https://bl.ocks.org/pbogden/854425acb57b4e5a4fdf4242c068a127>. Once we realized that we needed to have centroids that the circles clustered around and understood the code more clearly, we created this view:

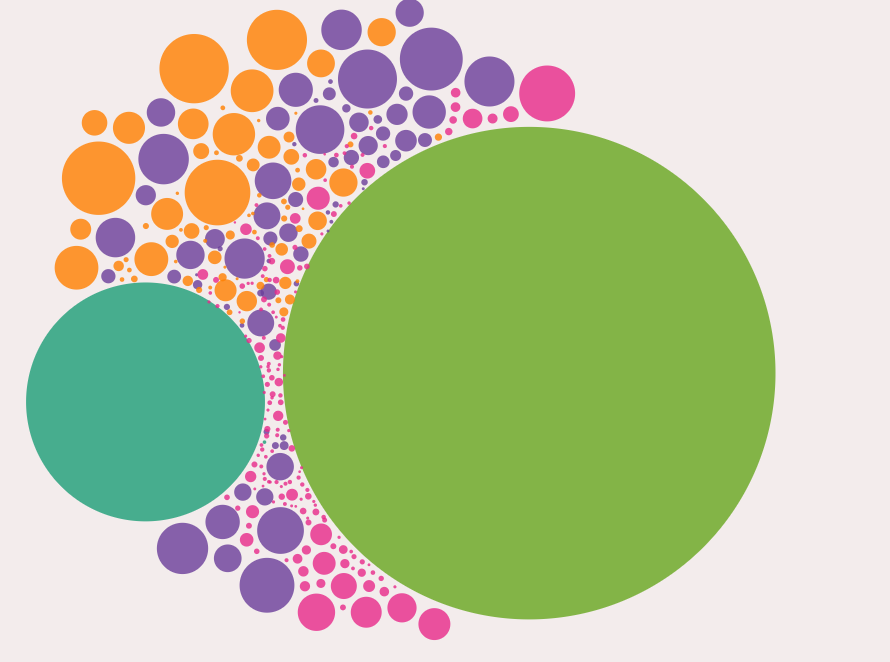


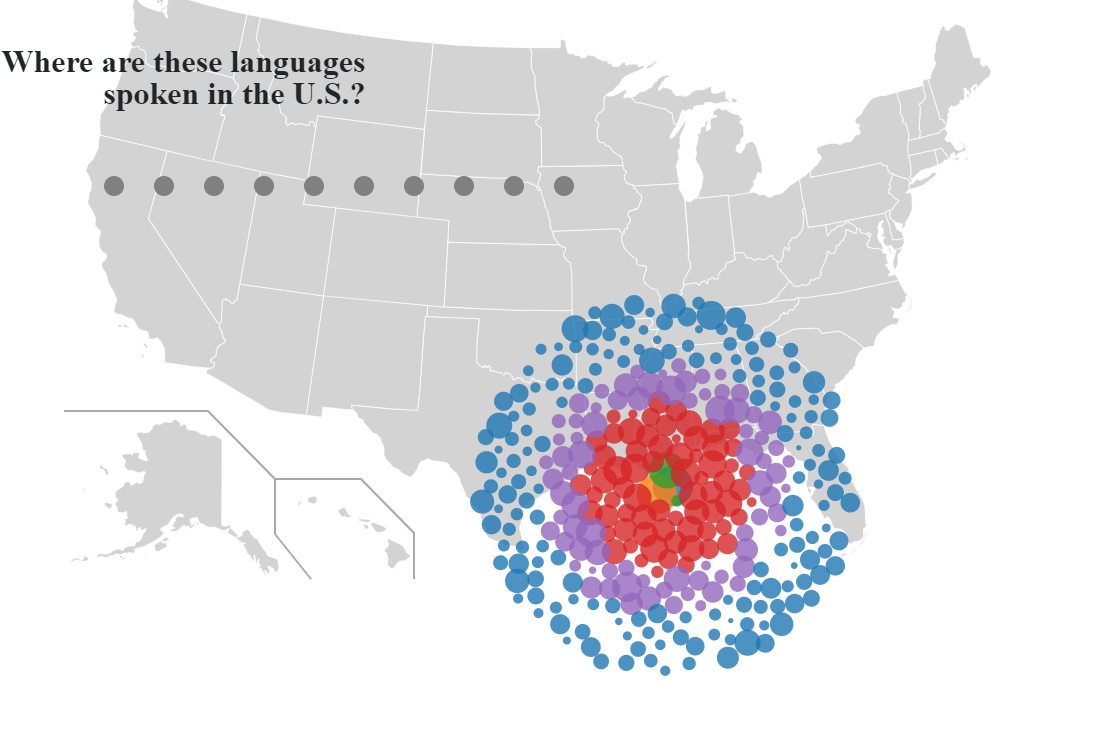
Figure 3: Initial clustering attempt

Finally, after fine-tuning some more, we ended up with the final result for our first visualization in our scrolling visualization.

**Scrolling**

Figuring out how to create a scrolling visualization seemed complicated at first, but became easier to understand once I got through reading the articles and going through the code myself. The initial inspiration for building a scroller came from this medium post: <https://towardsdatascience.com/how-i-created-an-interactive-scrolling-visualisation-with-d3-js-and-how-you-can-too-e116372e2c73>. The visualization that was created can be found here: <https://cuthchow.github.io/college-majors-visualisation/>. We read the medium article first and then consulted Cuthbert’s reference for building a scroller: <https://vallandingham.me/scroller.html>

The challenges with scrolling was difficult to get right initially since we were trying to coordinate 5 different views. An initial struggle was getting simulations to start in the middle of the page instead of being created in the top left and then floating to the middle. Another struggle was getting the visualizations to disappear by changing the opacity, so we would have errors like the following:



Eventually we were able to control the opacity of the visualizations. The next challenge was sharing one tooltip div between the visualizations. The issue was the order that the SVG’s were drawn was causing the tooltip to only work for the first visual and not the rest. In order to fix this, we learned to use the d3.raise() function.

**Data Cleaning**

The data that we received initially was not formatted in a way that worked well with our visualization. The data had groups, sub-groups, and the individual language rows with the totals and sub-totals as their own row. This would lead to problems with aggregating the data (since it would duplicate number values) and so we took the totals and sub-total rows and created new columns for them. We did the main portion of the data manipulation manually in excel since we decided it would take too long to complete all of it using a programming language and we were eager to start using d3 on the final dataset. We validated the new totals against the old sub-totals and totals to make sure we were not duplicating or deleting values and to help reduce the human-error in manipulating the data. There were also some letters being used for null values, and so we replaced all of the letters with NA’s so we had a common way to filter out null values in the data.