DC3: Visualizing Amazon Product Categories

Stewart Kerr
Ashwin Keshav Tayade

In this design challenge, we were given a set of product categories in Amazon which have a hierarchical relationship. There are many interesting encodings that can be used to depict this hierarchy, but the main problem we encounter is due to scale. Even though the hierarchy is not very deep (at most 6-7 levels deep), there are thousands of categories. Here we present some of the visualizations that we tried. Each design is accompanied by **the control flow** (which discusses what the design shows at each step using interaction), **the rationale of the design, tasks it is good for, and tasks it is not good for.** We also include a video which shows the designs being used interactively. At the end, we show a few designs that we tried, but which were ineffective due to the sheer number of things being displayed on the screen.

Design 1: Treemap

This is a popular design that emphasizes the part-whole nature of a hierarchical relationship. We implemented this treemap using a Javascript visualization library created by Google. This succeeds is helping the user explore the complete tree up and its scalable. But it lacks in showing direct hierarchy and comparison across subtrees is difficult.

Control Flow:

1. The starting view shows the children of the root node.

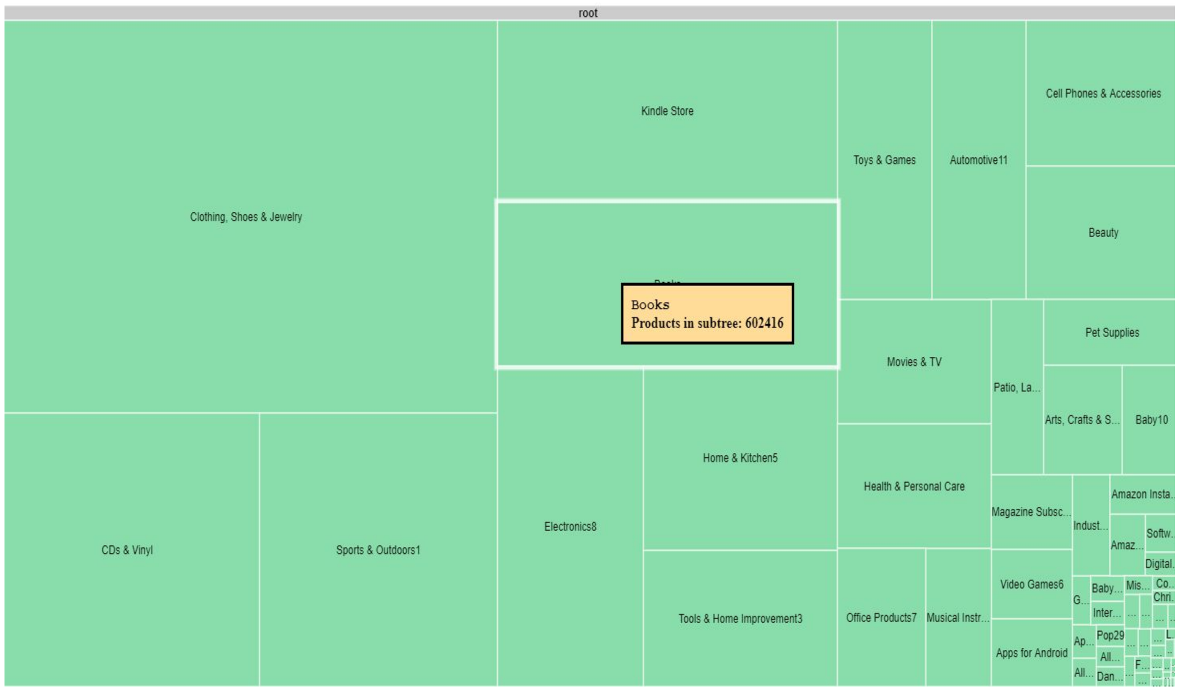


Image 1: All children of the root node

2. Two types of interaction are supported, clicks and hovers. Hovering displays the number of products in that child and below (subtree product count). The block is highlighted, name of the category and the number of cumulative products in subtree of this category is displayed (image 1).
3. Clicking on any block loads a new view with the children of the clicked category. In image 2, we see the children of the category Books.

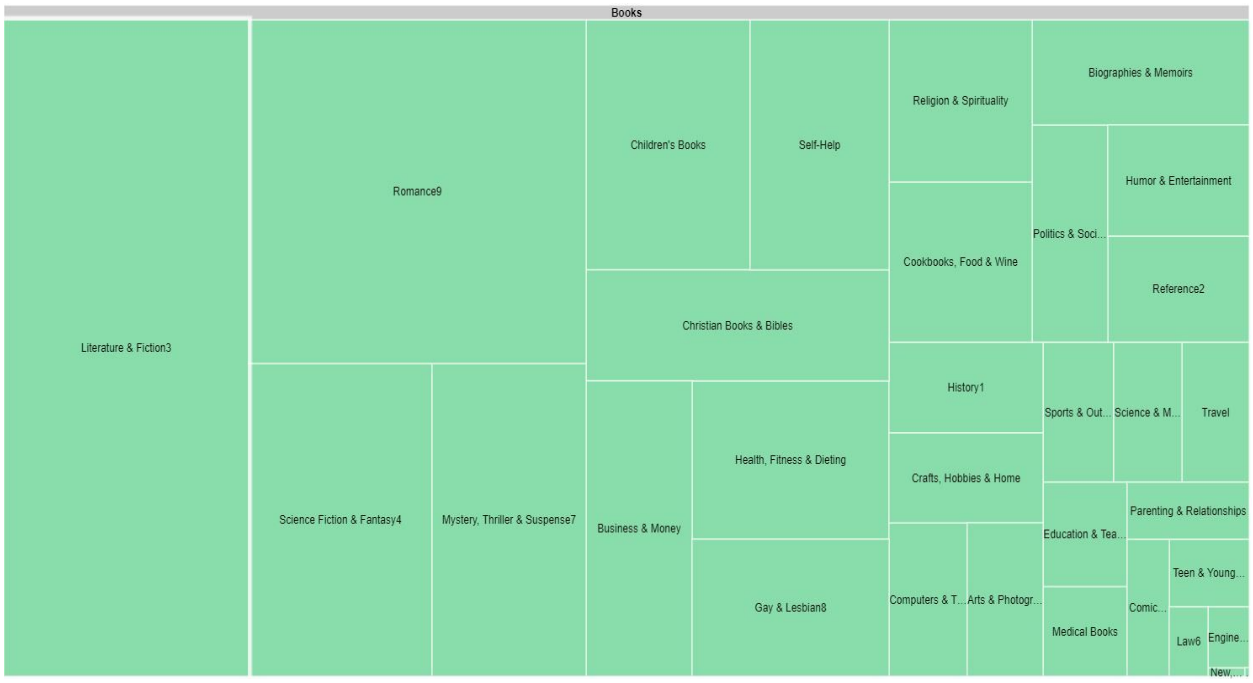


Image 2: Children of category 'Books' after clicking on the block 'Books' in image 1.

4. Now, we have the same options as above and additionally the option of going back to the parent node by right clicking on the grey tab 'Books' (image 2). Hence the design is recursive with each interaction supported in all levels
5. In image 3, we see the effect of going back to image 1 from image 2 and selecting Sports and Outdoors category.
6. Hence, we can traverse the whole tree by the use of mouse clicks and get the number of products by hovering.

Tasks Addressed:

- Data exploration using the complete traversal of the tree of categories and hovering to get the number of products in each category.
- Analyzing the distribution of the number of subtree products in a node by comparing the size of each block.

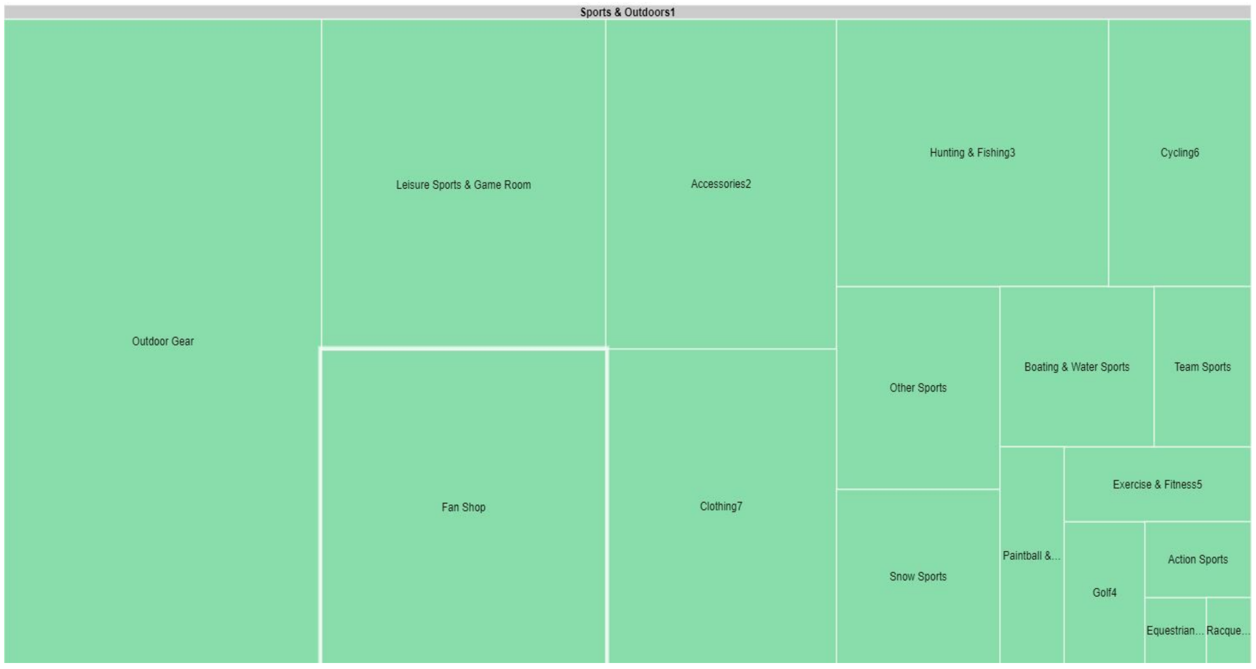


Image 3: Going back to Image 1 from Image 2 and clicking ‘Sports and Outdoors’

Design Rationale:

- A treemap is a good choice for visualizing the part-whole relationship of the number of products in a category.
- Size of the box is an encoding for the subtreeProductCount + productCount of the category.
- We wanted to introduce color encoding as an indication of the productCount but we could not do this due to limitations of the tool.
- Given the sheer number of nodes in the tree, we decided to show only the nodes at the current level, thus deviating from the standard treemap design which also shows subsequent levels too.
- We tackled the problem of scale by using interactive actions of clicking and hover. Clicking tackles scale by going deeper/shallower in the tree, while hover tackles scale by showing information for categories that might be too small for their complete name to be seen on the screen.

Design Shortcomings:

- If the task was to show the entire hierarchy in one view, then showing only one level of nodes is not enough.
- If showing all categories in one view, then name truncation for small categories might be confusing. This problem is somewhat alleviated by the use of hover interaction
- If the task was to compare the number of products in a category, then using area does not lead to the most accurate comparisons.
- Two categories in two different subtrees cannot be compared directly.

Design 2: Collapsible Tree

This design is the most obvious vis of tree data. It succeeds in helping the user explore the tree and solves one of the problems of design 1 by showing all the hierarchical relationships in one view. If we show product count on hover, we can facilitate some comparison across subtrees, but it still requires that the user remember values. This design was implemented using D3.

Control Flow:

1. This design starts of with the view as shown in image 4. We see only the top level nodes of the tree. These are the direct children of the root. The nodes which can be expanded are black and the nodes which are leaves are greyed out.

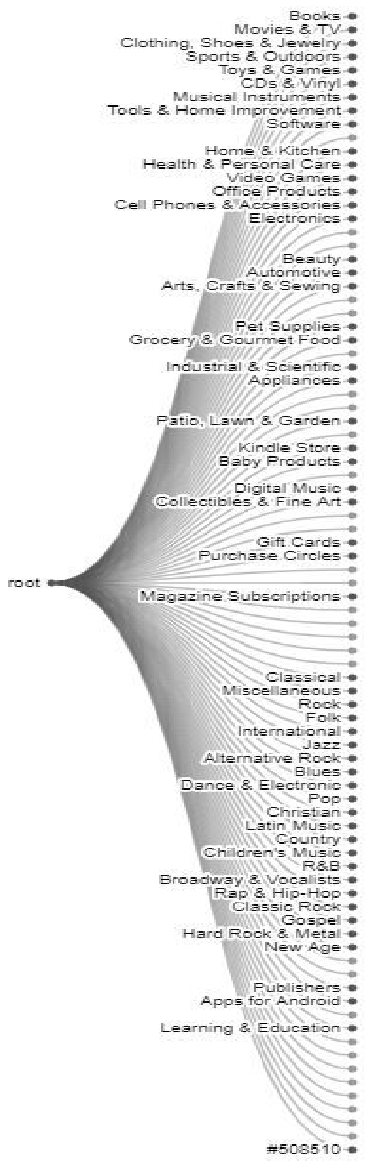


Image 4: Tree with the first level of children

2. This is a more standard view of the hierarchy than design 1. Clicking on any node of the tree expands it as shown in image 5.

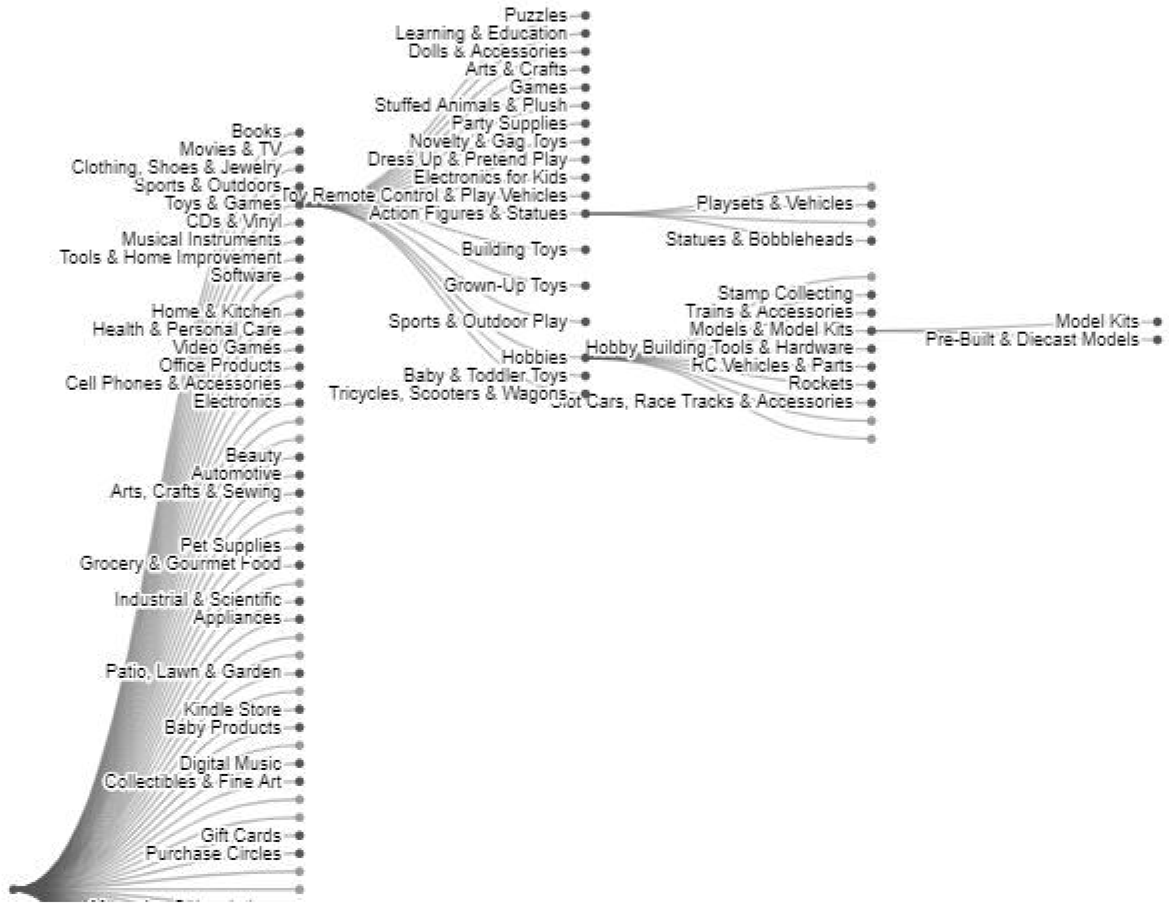


Image 5: Partial view of the tree with the game subtree expanded.

3. The first click on the node expands it and the second click shrinks it back. This can be seen better in the included video.
4. We wanted to show the number of products in the category when you hover over the node, but this has not been implemented.
5. By clicking on nodes, the whole tree can be expanded.

Tasks Addressed:

- Exploration of the tree is very intuitive and encapsulated in a single view. Any user can explore any category subtree of their choosing completely.
- Unlike the previous design, the hierarchy is very obvious using collapsible trees.

Design Rationale:

- It’s the most natural way to visualize a tree structure.

- The issue of scale is handled by using clicks as interaction. The user can expand only the tree they are interested in and collapse those they do not want to see. This addresses the issue of displaying too much stuff on the screen.
- The initial view shows only the top level categorize so that the user does not have to collapse any unwanted trees.
- We wanted to show the productCount and subtreeProductCount on hover, but this has not been implemented.
- Color distinction between leaf nodes and non-leaf nodes is done so that the user does not end up clicking leaf nodes and deciding that the vis is broken.

Design Shortcomings:

- If the task is to compare the number of products between categories, then this design is unable to facilitate that comparison. With the intended 'information on hover', this problem is somewhat alleviated but the user still has to remember the number in each category as there is no direct visual encoding that shows the number of products.
- Scale of the data can become an issue. For example, the subtree of 'Clothing, Shows and Jewellery' is large. The user needs to scroll to view all of the nodes in this category. If we add an additional subtree, then comparison of nodes in this subtree and another is even more difficult because even more scrolling is required.

Design 3: Bubble chart

This design tries to alleviate some of the problems in designs 1 and 2. The bubble chart shows hierarchical relationships as well as facilitates comparison across any number of subtrees. But it is at the cost of scalability. The 'Books' subtree is the largest set we could visualize coherently. This design was implemented using D3.

Control Flow:

1. To overcome some of the problems of design 1, specifically, not being able to compare sizes of categories across subtrees and not providing a clear view of the hierarchy, and problems of design 2, not being able to compare sizes directly, we made a bubble chart that does both (image 6). We see all the categories in the subtree of books as a part whole relationship as well as the hierarchical relationship. The names of the direct children of 'Books' are seen.
2. Clicking on any of the circle zooms in on that child as seen in image 7.
3. We can keep on expanding the categories until only the leaf nodes remain.
4. Clicking outside the category zooms back out to the previous view. This helps us traverse the tree in both up and down directions.

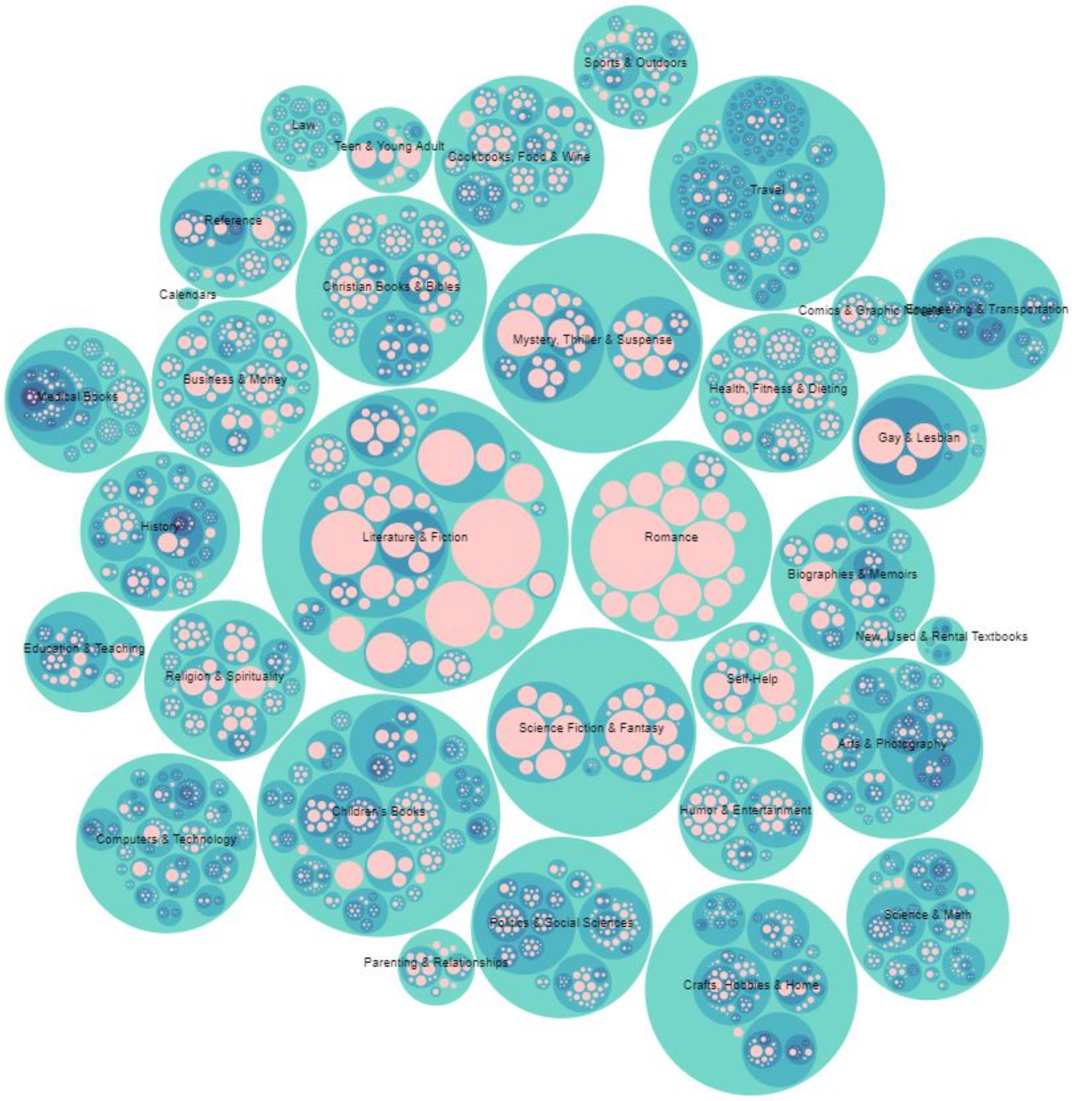


Image 6: Bubble chart showing everything in 'Books'

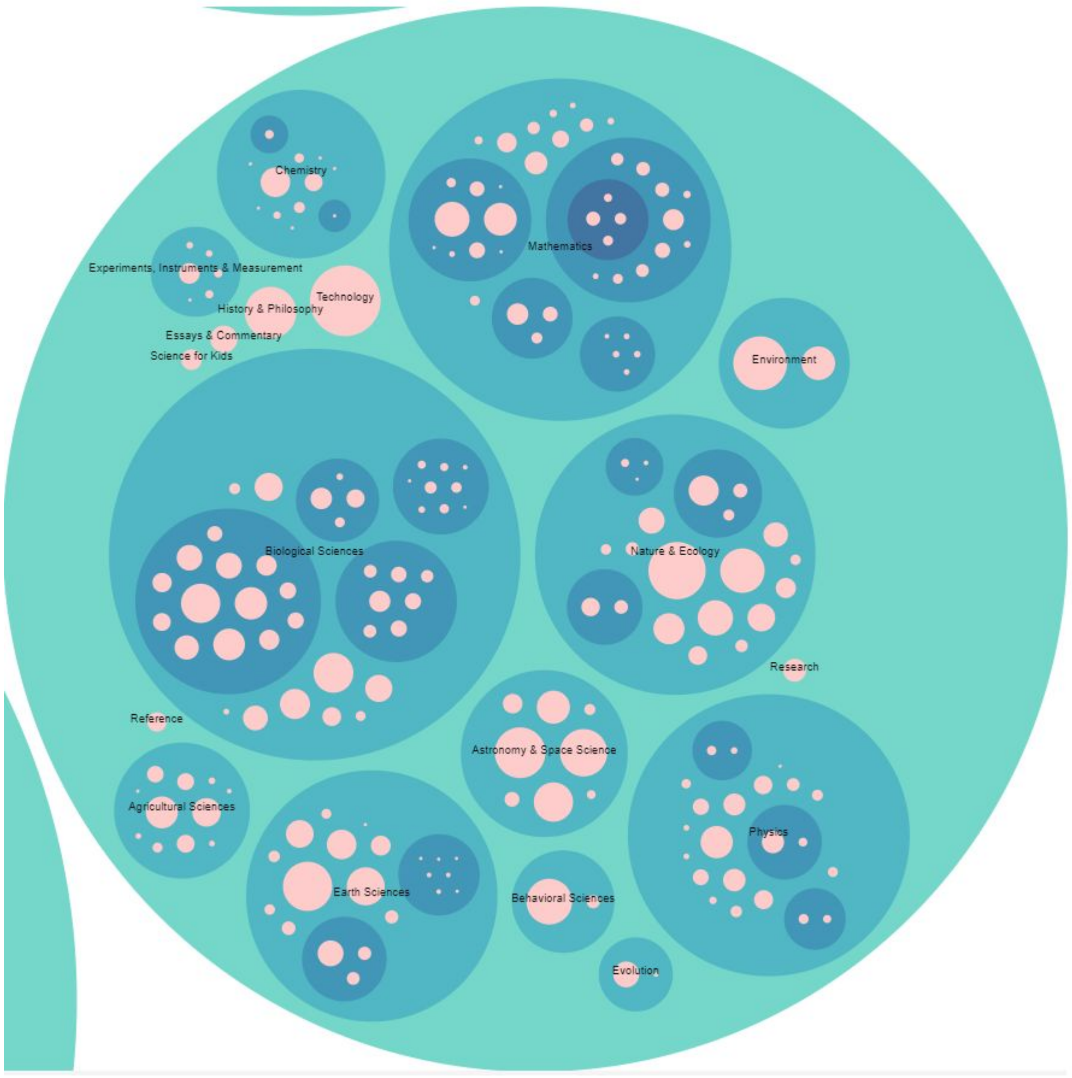


Image 7: Showing the 'Books' subtree of 'Science & Math'

Tasks Addressed:

- Visualizing hierarchical relationships.
- Comparing the number of products in a subtree and across subtrees.
- We can get an idea about the number of levels in the subtree by focusing on the colors of the bubbles.

Design Rationale:

- The design shows the hierarchical relationships as well as the relative sizes of the subtrees. This was an attempt to cover the shortcomings of the 1st and 2nd designs.
- The size encoding of the circles is the number of products in that subtree. This facilitates comparison across categories and subtrees giving the user a clearer idea about which categories have more number of products than others.

- The color encoding is more of an aesthetic thing. The darker the color of a bubble, the deeper in the subtree it is. The pink bubbles indicate leaf nodes.
- The design is good for seeing hierarchy and comparing number of products, but it is not scalable after a point. Using the complete data breaks the code and using the biggest subtree (Clothing & Jewellery) renders the vis incoherent with lots of data colliding with each other. It works from the second largest subtree and smaller.

Design Shortcomings:

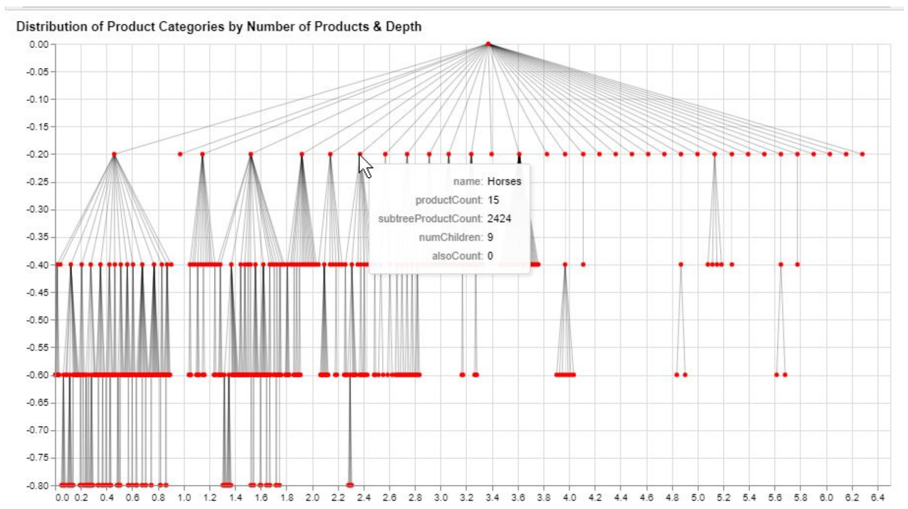
- As talked about earlier, the design is not scalable after a limit. The second most populous subtree is the limit after which the coherency of the design is lost.
- If any subcategory has very few products, it would practically be invisible in the design before zooming in.
- If there are too many levels in the tree(in this data there are only 6-7 levels), coherency will be lost.

Design 4: Interactive Dashboard

The purpose of this design is to explore other aspects of the tree data. We want to better understand the distribution of depth and number of products in the tree. First, we show a graph of the tree data. This helps us get the overall shape of the tree and allows for interaction which drives the rest of the visualization. Below the graph, there's a histogram of product categories by number of products and a bar chart of product categories by depth. The screenshots below are for the 'Pet Supplies' dataset, but we were also able to apply the visualization to the 'Books' dataset. This design was implemented using Altair in a Jupyter notebook.

Control Flow:

1. The user is first presented with a vertical tree graph of the data (image 8). The user can hover over a particular node to see information about that particular node (name, product count, alsos, etc.). A set of nodes can be selected by clicking and dragging on the visualization. This highlights the selected nodes and affects the charts below.
2. After selecting a set of nodes, the user looks to the two graphs below - a histogram on the left and a bar chart on the right. The data corresponding to the selected set above is highlighted (image 9). By default, all nodes are selected, thus summarizing the entire tree. The user can also select particular regions/bars of this visualization to highlight those product categories in the other visualizations (image 10). Exact counts can be seen by hovering over the bars.


Distribution of Product Categories by Number of Products & Depth

name: Horses
productCount: 15
subtreeProductCount: 2424
numChildren: 9
alsoCount: 0

Image 8: Showing "hover to discover" interaction of tree graph

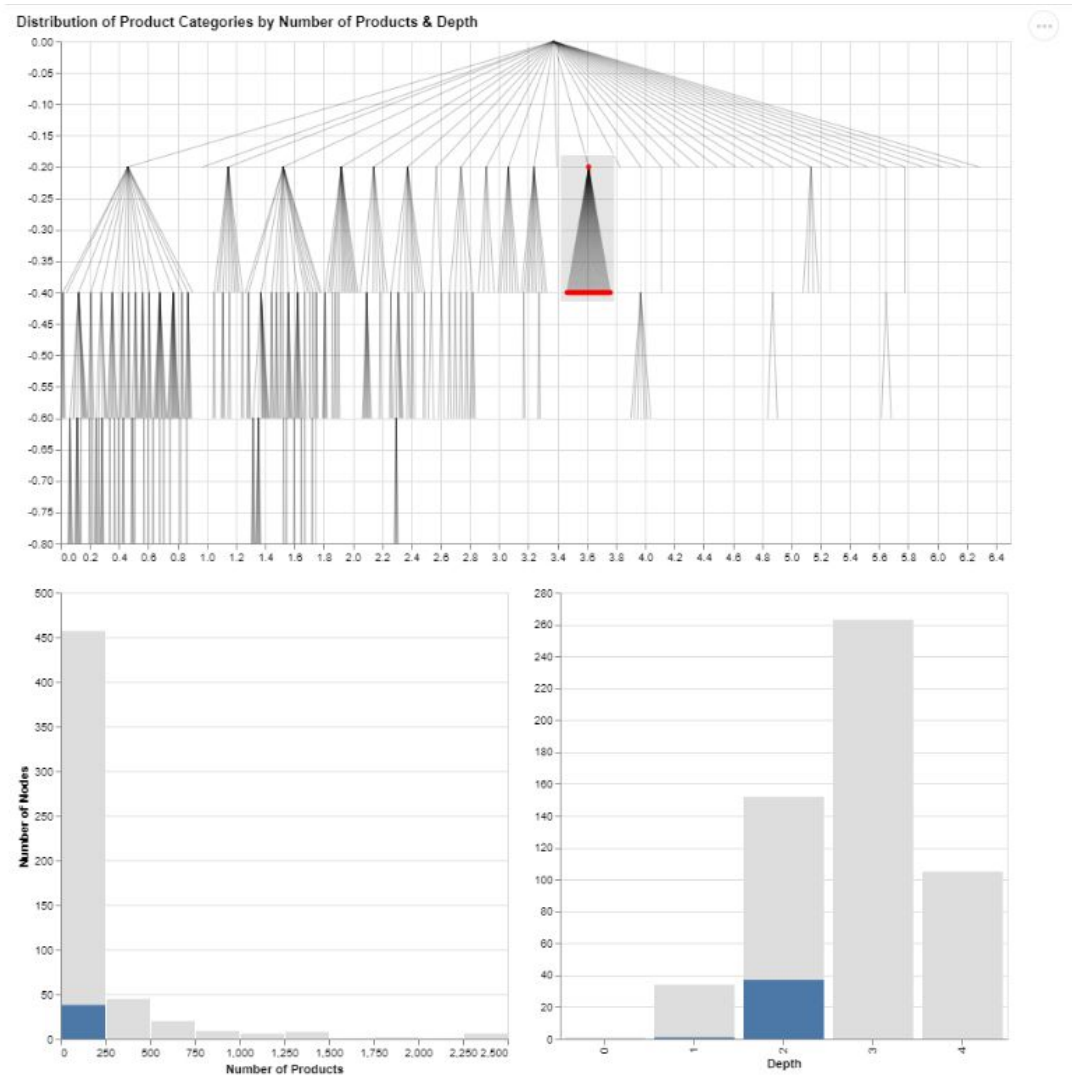


Image 9: A subset of nodes is selected and the resulting data points are highlighted.

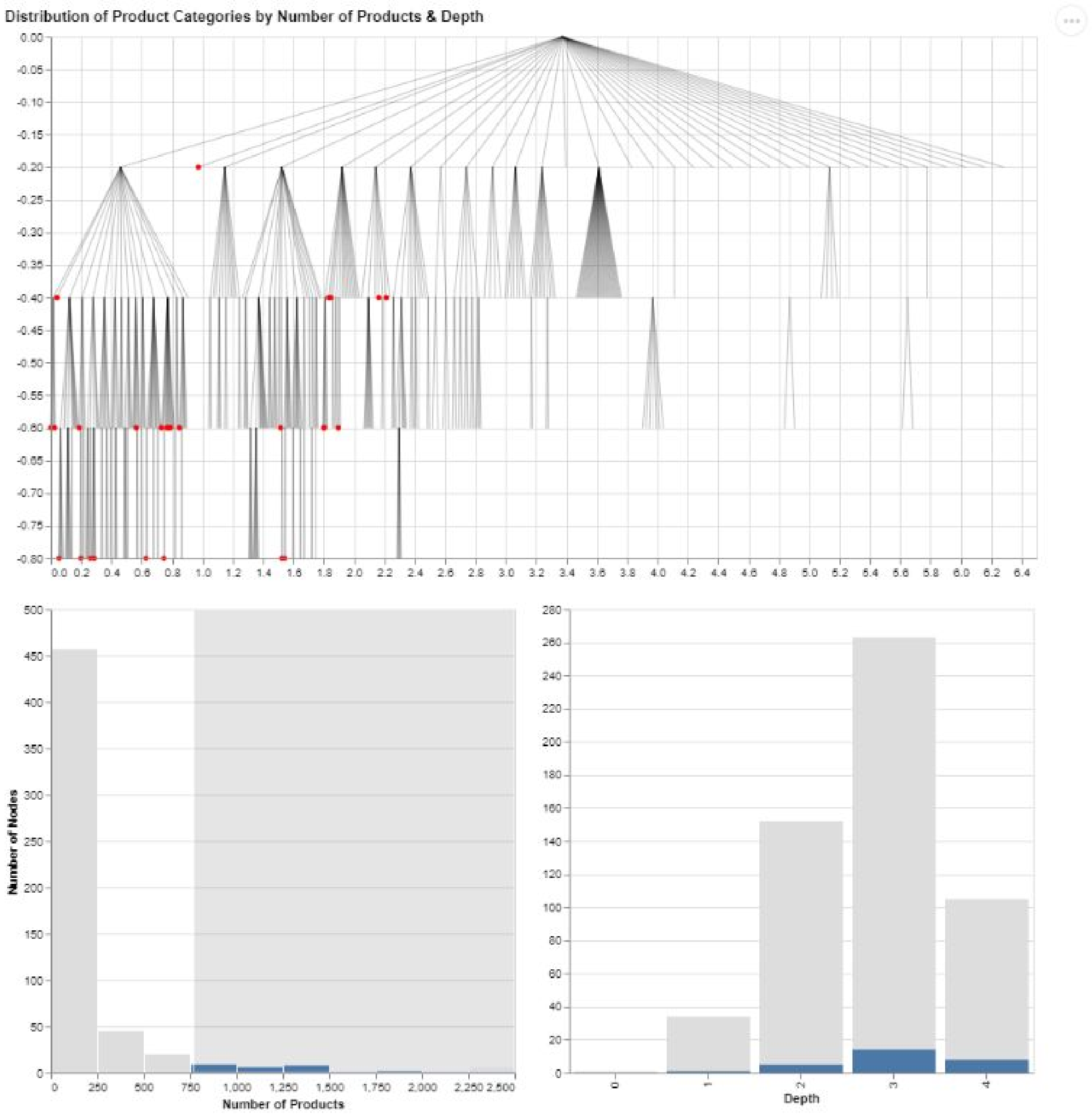
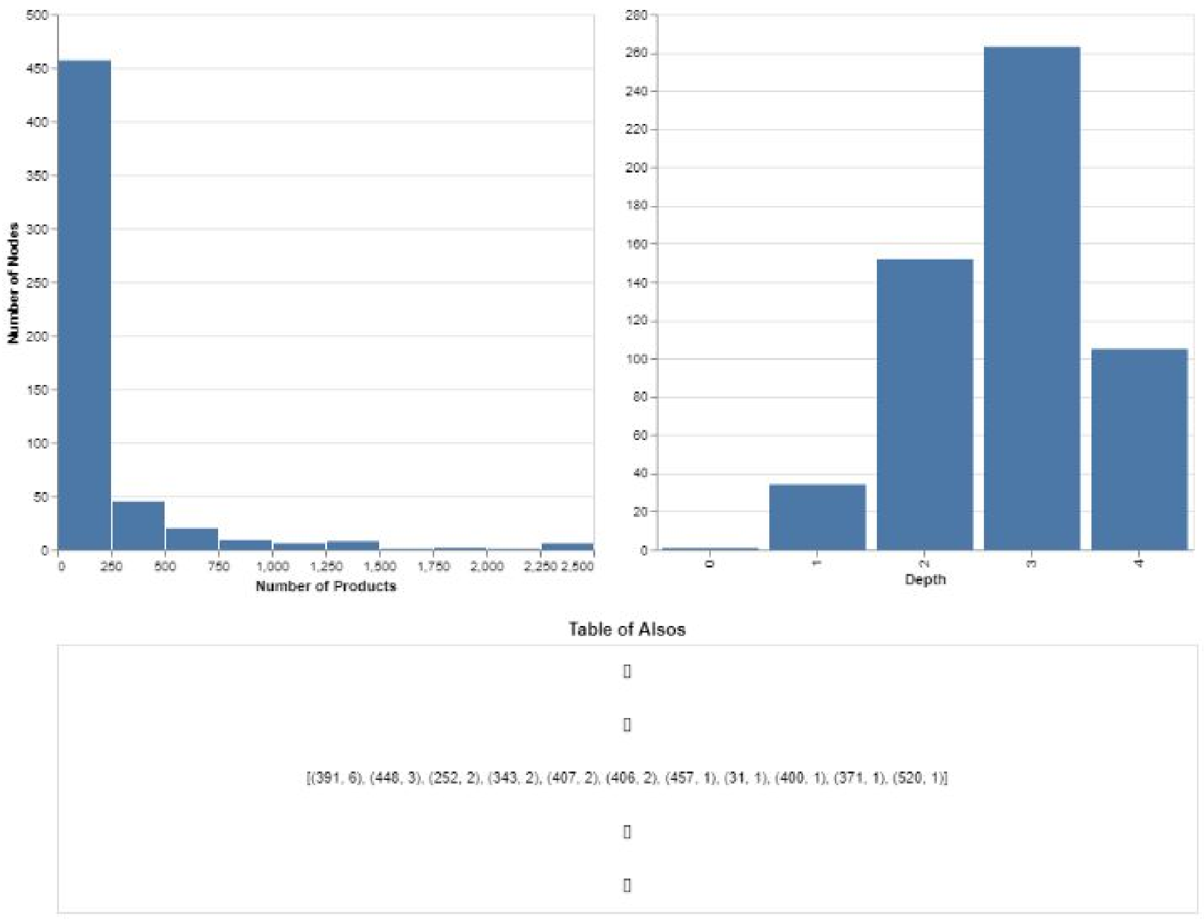


Image 10: A subset of product categories are selected and the resulting data points are highlighted.

3. Below the charts, a table of the top 5 most connected product categories among those selected is displayed. Again, by default, all nodes are selected - thus the table will show the 5 most connected product categories (image 11). Note that we were not able to actually implement the table showing the name and count of connectedness, but, given more time, we would implement this feature.



▯

▯

[(391, 6), (448, 3), (252, 2), (343, 2), (407, 2), (406, 2), (457, 1), (31, 1), (400, 1), (371, 1), (520, 1)]

▯

▯

Image 11: Showing the table of connected product categories for the entire tree dataset.

Tasks Addressed:

- The main task addressed by this design, which our other designs fail to address, is providing a sense of the structure of the tree in terms of counts of products, depth, and their interaction.
- By selecting a specific interval in the number of products distribution, we can determine which subtrees have a higher number of products than other subtrees.
- The table of alsos allows the viewer to see which type of other categories may be highly connected to a selected subtree of interest. Given more time, I would also hope to implement highlighting the shared product categories of a selected set of nodes in the tree graph.
- Interaction allows the viewer to see summary data of the entire tree as well as any subtree they are interested in (although it requires the subtree can be enclosed in a rectangle).

Design Rationale:

- We wanted to create a series of visualizations connected via interaction that addressed some of the shortcomings with our other designs. Placing the graph of the tree first allows the viewer to explore the entire tree and informs what they are most interested in. We decided to make the tree graph vertical rather than radial because it allows the viewer to more easily select a subset of the nodes. It also better reflects the depth of the tree because nodes in the same depth are at the same place on the y-axis. We could have decided to encode one of the attributes of the nodes in the node color hue/saturation.

- As the number of nodes and number of products are both numeric ratio data, a histogram is an appropriate design to depict their distribution. We determined that bins of size 250 adequately subset the distribution of number of products. We decided to depict number of products rather than number of subtree products because number of products is unique for each node while number of subtree products counts products multiple times.
- For the last chart, depth is a categorical variable while number of nodes is ratio, thus, a bar chart is a good selection. It allows us to accurately compare the number of nodes between depths and get a sense of the distribution among the different tree depths.

Design Shortcomings:

- In its current form, the visualization does not handle large scale data very well. In particular, the amount of overdraw in the top chart increases with increasing number of product categories. A possible solution to this problem is adding pan and zoom, however, due to a limitation with Altair, we can't have both interval selection and the ability to zoom in the same image. This is a feature that is available in VegaLite and will be coming to Altair shortly. If we have the ability to pan and zoom, then this visualization scales with larger data fairly well.
- There are many product categories which have a very low number of products (0 - 250) and few product categories which have a very large number of products (2000+). This causes the histogram to be left skewed and certain categories are occluded.
- While the table of alsos provides some information about the interconnectedness of the tree, it does a poor job of summarizing the alsos in the tree. Ultimately, the design would need some other encoding to better depict the connectedness of the graph. One idea is to include dotted lines between nodes that have shared products with either a color saturation or thickness ramp according to number of shared products. I think it's unlikely that we would be able to implement that feature using Altair.

Discussion of Data & Findings

We were provided with trees of various sizes, ranging from the pet supplies tree (smallest) to the books tree and a tree that included all subtrees (largest). Ultimately, only designs 1 & 2 above, the treemap and the collapsible tree, were able to handle the largest tree that contained all subtrees. Designs 3 & 4 were able to handle up to the largest subtree provided (Books). While the treemap and collapsible tree were able to handle the largest tree, these designs do not provide as much description of the structure of the tree, thus we will focus on examples of what designs 3 and 4 helped us to see. We will examine the examining the books and musical instruments trees.

Books:

By visually scanning the bubbles in the chart below (image 12), we can see that, while the books tree has a few subtrees that are larger or smaller, most of the depth 1 subtrees are approximately equal in terms of the number of products in the subtree. The subtrees that are abnormally larger or smaller (for example, literature & fiction and textbooks respectively) stick out. As leaf nodes are encoded with the color white, we can see that the larger subtrees (literature & fiction, romance) have very large leaf nodes, Furthermore, for the romance genre, these leaves are shallow in the subtree.

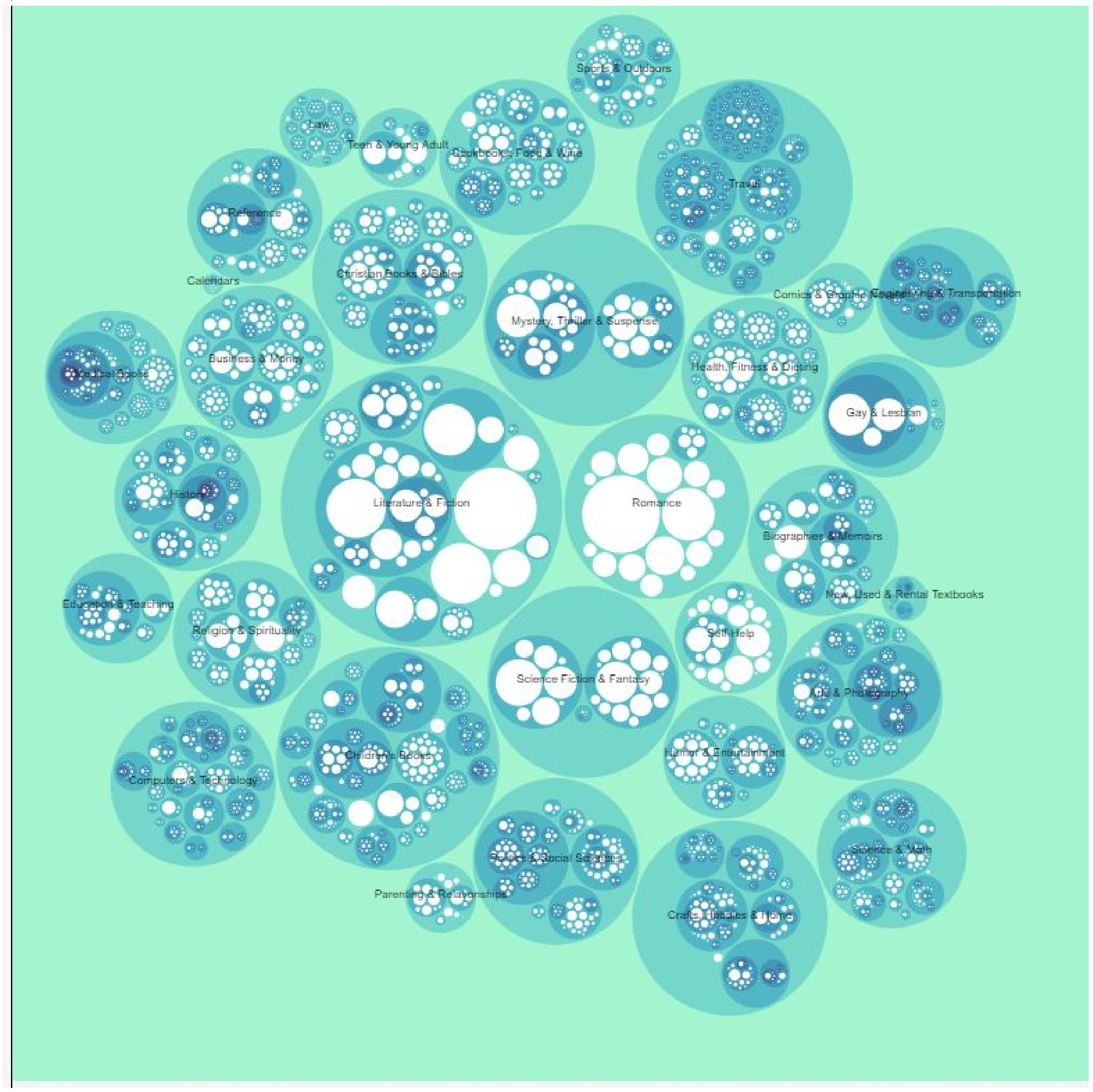


Image 12: Bubble chart of the books tree.

Musical Instruments:

For the musical instruments tree, we wanted to explore what we could learn from the final design. First, if we ignore the highlighting and look just at the height of the bars, we see that most of the product categories have under 500 products but there are a few that have a much larger number of products. Also, we see that most of the nodes in this tree are depths 3 & 4. There are relatively few categories in the deepest part of the tree (depth 6). If we select the depth 6 bar, we can see that these deep nodes exist entirely in the smallest number of products bin (image 13).

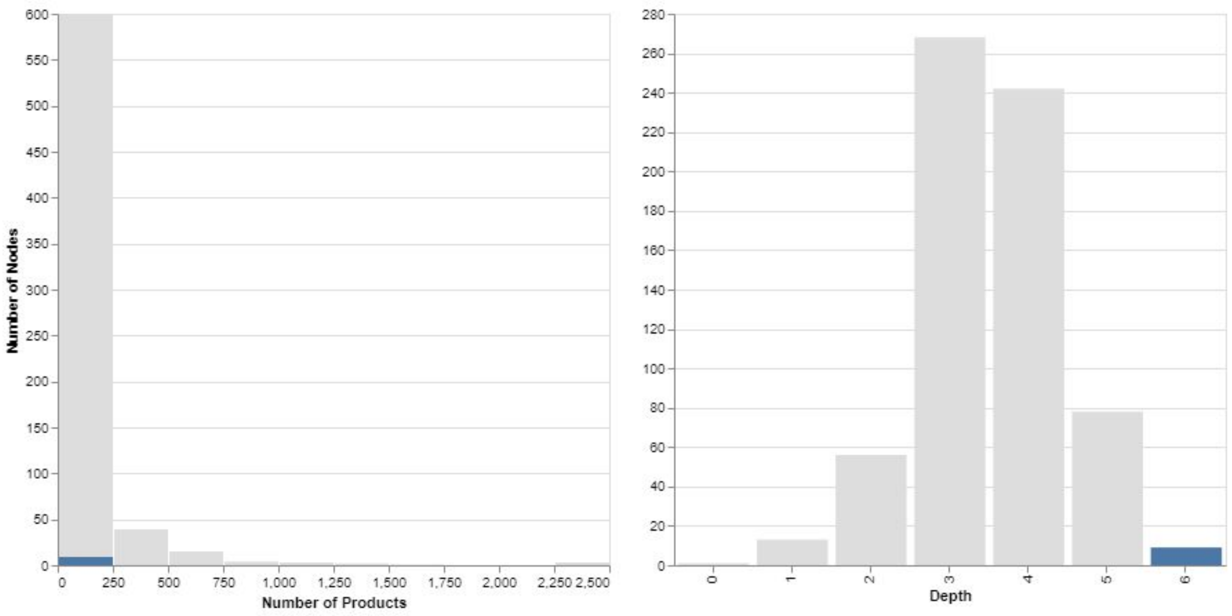


Image 13. Highlighted number of products for categories that are depth 6 in the musical instruments tree.

Self-Assessment & Conclusion

The designs were implemented with different libraries. Designs 1,2, and 3 were implemented using d3 with code that was readily available online. Neither of us had previous knowledge of d3 or Javascript, but because we were able to find existing packages we did not have to learn too much. Design 4 was a custom non-standard design implemented using Altair package in python. Stewart used Altair for design challenge 2, so he was more familiar with using it.

We tried a number of alternate designs using other packages. For example, we tried implementing design 1 using matplotlib, but it did not turn out as good as the one shown here (image 14).

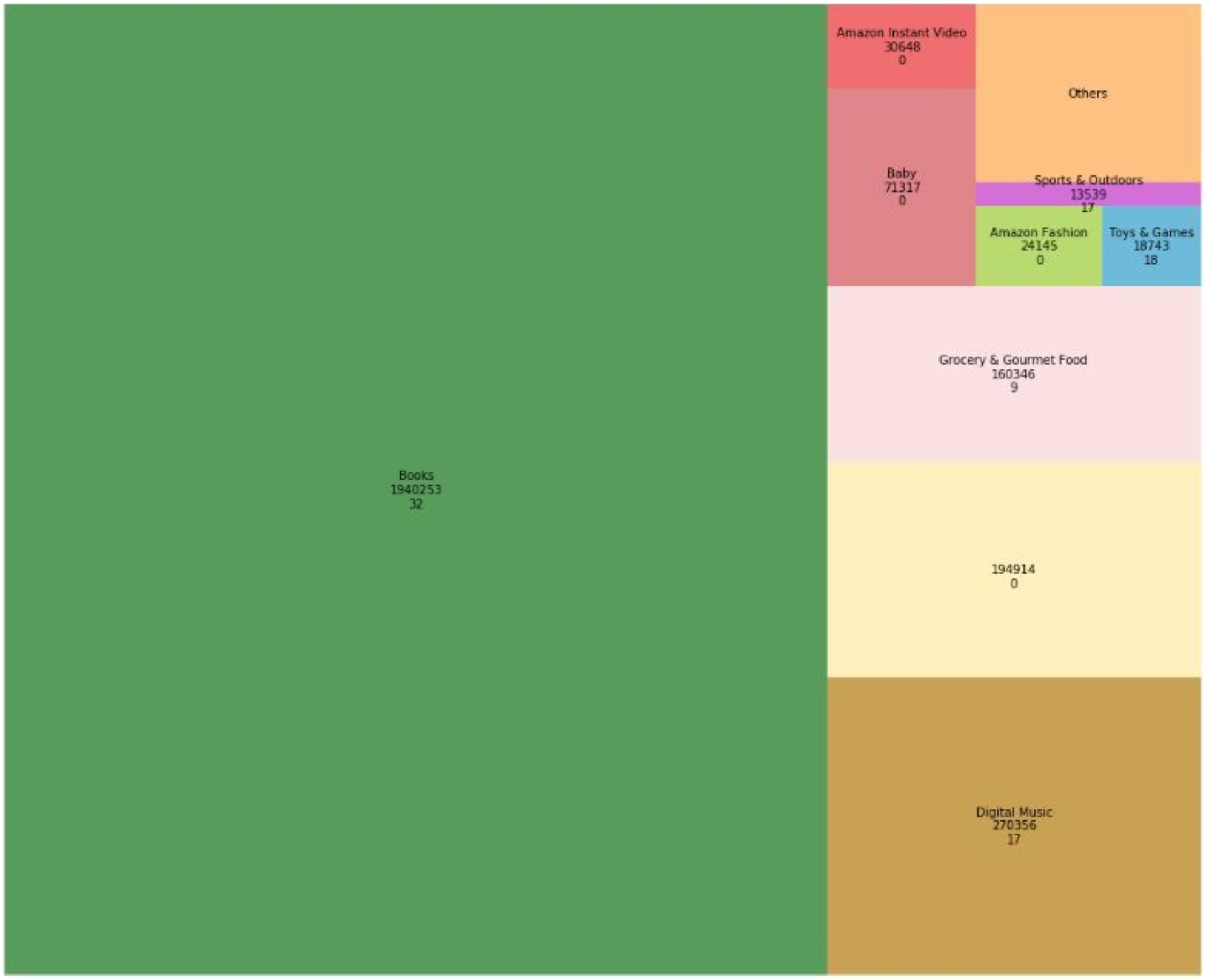


Image 14: Treemap using matplotlib with 'Others'

- Shows the 9 largest children of the current node and groups the remaining categories under the 'Others' node. Note that this design is wrong in the sense that productCount was used instead of the subtreeProductCount. The correct version was not implemented as we found the superior design in d3.
- Clicking the 'Others' block shows the next 9 largest children of this same node and other children are clubbed under 'Others' as seen in image 15. Note that 'Others' now has all other children of current category other than the 18 children already shown.

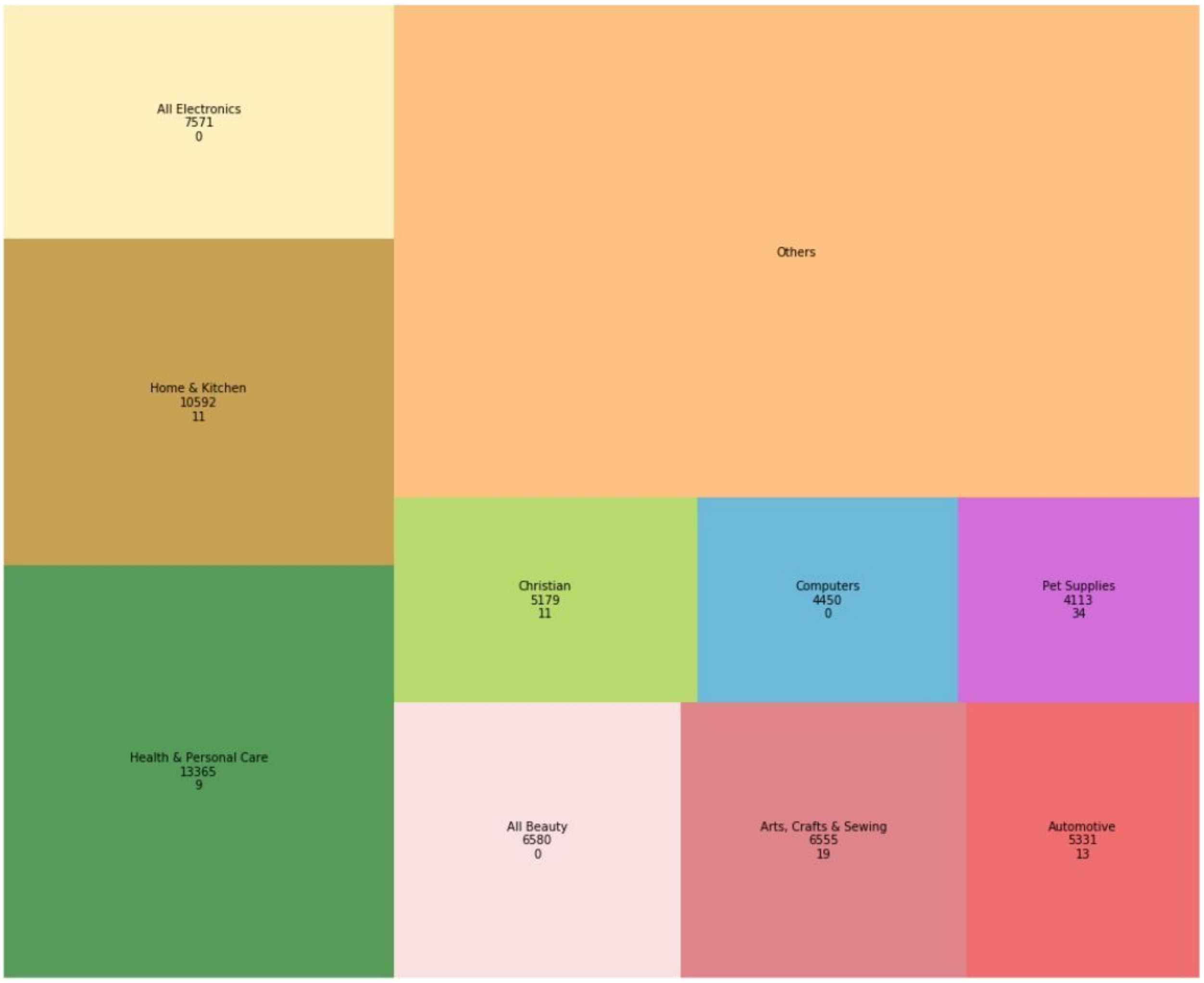


Image 15: Next 9 largest children of root with the remaining children under ‘Others’

- Whereas clicking on any other block than ‘Others’ in image 14, will display the children of that node in similar fashion (top 9 children with rest clubbed under ‘Others’)

To explore the alsos concept we tried a matrix plot with leaf nodes and color coded by the alsos count. This was not coherent at all due to the large number of leaf nodes. Image 16 shows a zoomed version of such a plot but it is not very helpful.

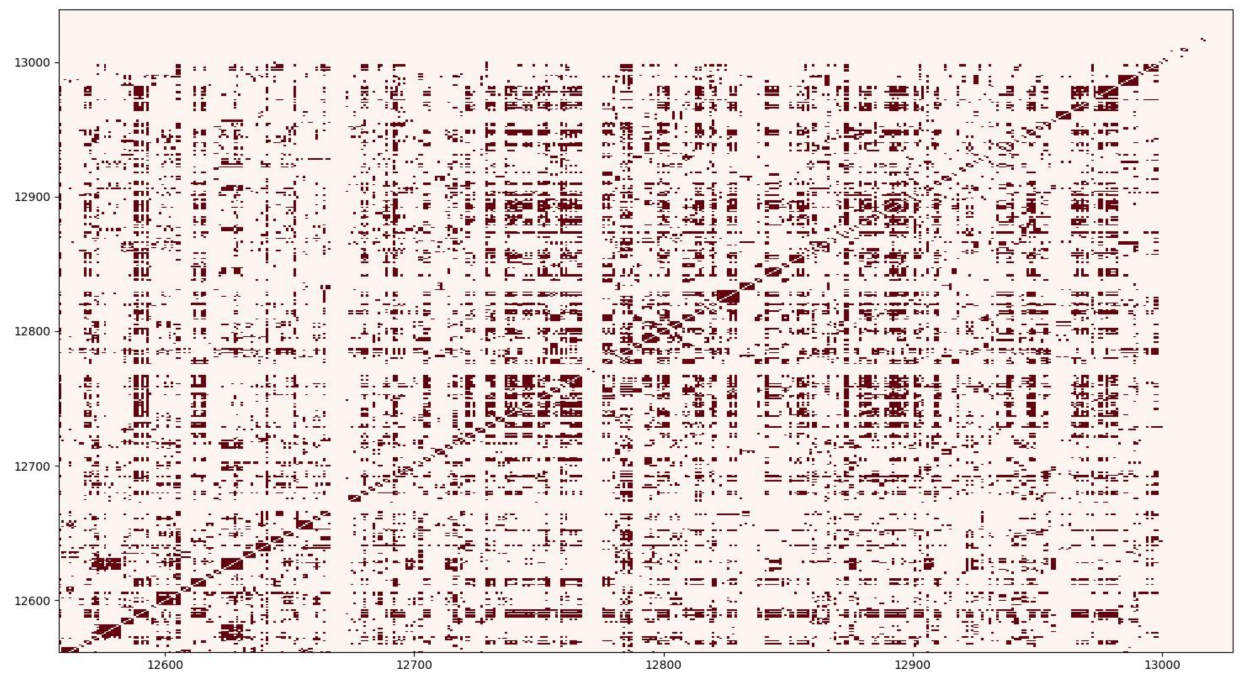


Image 16: Matrix plot with leaf nodes in columns and rows.

There are some features we wanted to implement, but were unable to do so because of either time or technology constraints. With the collapsible tree design, we wanted to implement hovering to show productCount and subtreeProductCount. This would seriously improve the design's ability to address an exploration task. For design 4, we really wanted to implement some better way of visualizing the alsos. For example, when a set of nodes is selected, the nodes that share products with the selected set are highlighted. We could also encode number of shared products with size or color saturation. Due to limitations/difficulties with the Altair library and its interface with networks, we were not able to implement that feature. Regardless, despite the challenges of working with large tree data, we are generally pleased with our designs and implementations.