project audiophile: first visualization

report + design brief

Link to visualization: http://www.sfu.ca/~ngmandyn/ProjectAudiophile/
GitHub repository: https://github.com/ngmandyn/ProjectAudiophile

Many people listen to music leisurely, and most who do also want their personal listening experience to be a pleasant one. Those with the desire to upgrade theirs have the privilege of choosing from a myriad of high-fidelity ("hi-fi") headphones to purchase.

From the budding audiophile's perspective, the choices can be overwhelming. Technical information and advice is often fragmented across forums and review sites. Through Project Audiophile, we will develop an interactive visualization that assists a broad range of audio listeners with exploring purchase alternatives for hi-fi headphones.

The questions we hope to answer will depend on the user and their specific needs. For example, they can vary from "What are the most power-efficient pairs of headphones $200 can buy?", to "Which brands offer options for closed-back headphones with 80-ohm or higher impedance?", to "Does a headphone's weight have anything to do with its price?". A common visualization task between these questions is examining the entirety of options based on certain criteria. However, different users will have different specs of interest. To support these differences, we began Project Audiophile with the creation of a reconfigurable scatter plot.

Our scatter plot uses the x- and y- axes to encode continuous numerical values, with the addition of colour to encode categorical data. We opted to represent the headphone data crowdsourced from r/Headphones using a scatter plot, which is useful for visualizing the distribution or "big picture" of the options available (each point representing a single headphones model) as well as for visualizing relations between variables (such as for answering the question of whether a headphone's weight is related to its price).

The choice of visual encoding is flexible. Users are able to adjust the mapping of data attributes to the y-axis and to colour. Dimensions available for mapping to the y-axis include `impedance`, `efficiency`, and `weight`. Dimensions for attributes available for mapping to colour include `brands`, `form factor`, and `amplifier requirements`. As the user reconfigures the encoding, the y-axis and colour legend are updated.

In future iterations, we will add features supporting other important visualization tasks, to better help users answer the above questions. Such features will include filtering with dynamic query widgets to narrow down a set of choices, brushing via a search function to select specific items, details-on-demand and annotations revealed on-hover, and an additional view to facilitate comparisons of selected models. We would also like to integrate additional data to cover more of the criteria users may be interested in, such as reviews, sound signature, and materials.

code design

Below are the main constructs and functions used in our code and what they do:

dataLoad.js

`dataLoad.js` holds the functions and initial properties used to set up the visualization, including importing the dataset, `prepData(data)` for trimming the data and converting quantitative values to integers, initializing position vectors, variables, the scatter plot along with its scale and legend using `initData(data)` and `initVis(data)`.

changeableAxis.js

Whenever a user reconfigures the mapping of the y-axis using the drop-down menu, `updateYAxis(data, value, yscale, yaxis)` is executed. The function takes the value (`Convert to Efficiency`, `Impedance`, or `Weight`) selected by the user and rescales the y-axis using the appropriate range and units.

changeableColour.js

Whenever a user reconfigures the mapping of the colours using the drop-down menu, `updateColours(data, value)` is executed. The function takes the value (`Manufacturer`, `Form factor`, or `Amp required`) selected by the user and updates the colour scale and legend.

dataFunctions.js

`dataFunctions.js` contains helper functions used to format data for use such as `trimWhitespace(data, arrOfColumnNames)` and `stripUnits(data)`, and functions for basic operations such as `getMax(data, columnName)` and `getMin(data, columnName)`.

external resources used

Data

- Headphones Database (crowdsourced from r/Headphones): https://docs.google.com/spreadsheets/d/10Rrh9lOZBKNOQjEbb3b9BB_l0vurApr2cWF6o3zo-vw/edit – adapted by removing redundant and duplicate data, populating missing data, and formatting values for consistency.

Libraries and code

- D3 (Data-Driven Documents): https://d3js.org/ – used to create the scatter plot visualization by manipulating the HTML DOM (Document Object Model) based on data.
- D3 SVG Legend: http://d3-legend.susielu.com/ – used to create the scatter plot legend using a D3 ordinal colour scale as the basic input.
- jQuery: https://code.jquery.com/ – used to simplify the manipulation of our HTML code.