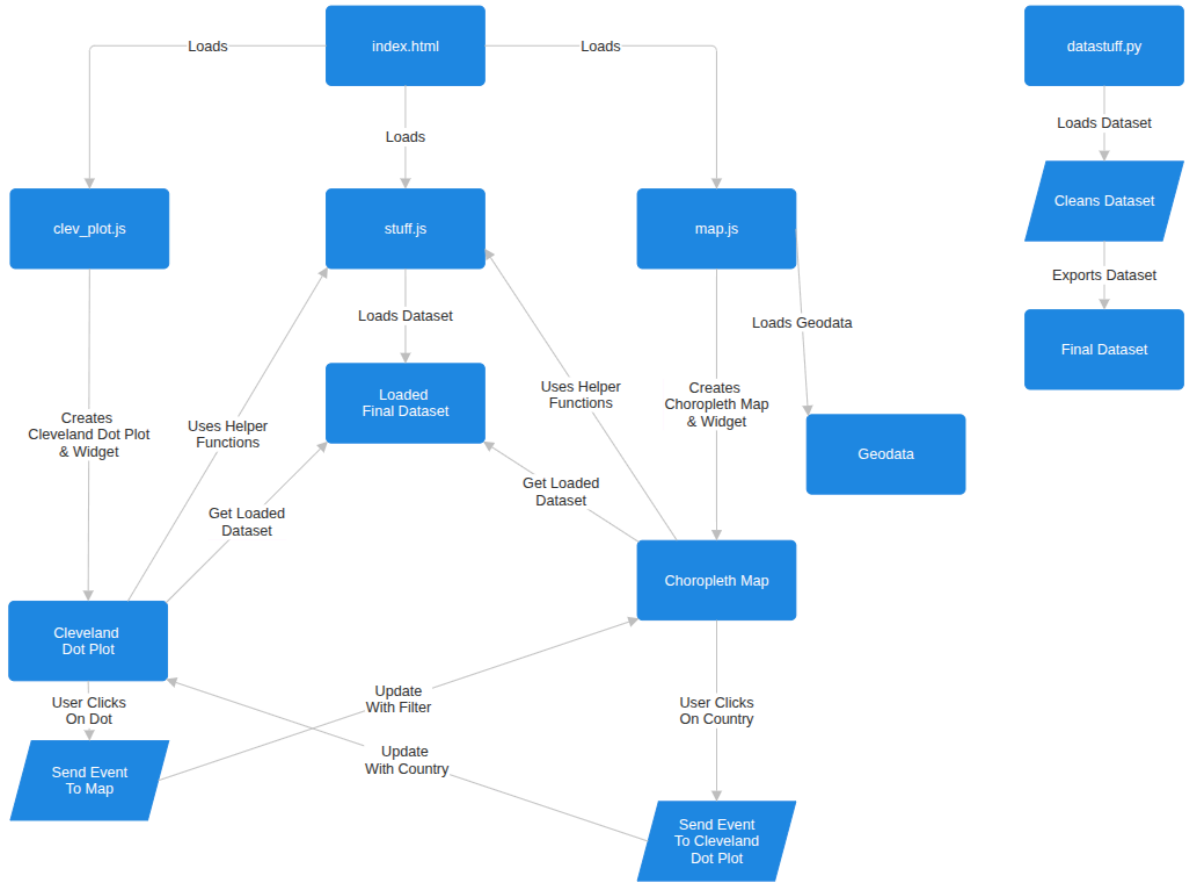
Checkpoint III: First Prototype

Group: 34
Date: 2025/09/29

Prototype Architecture



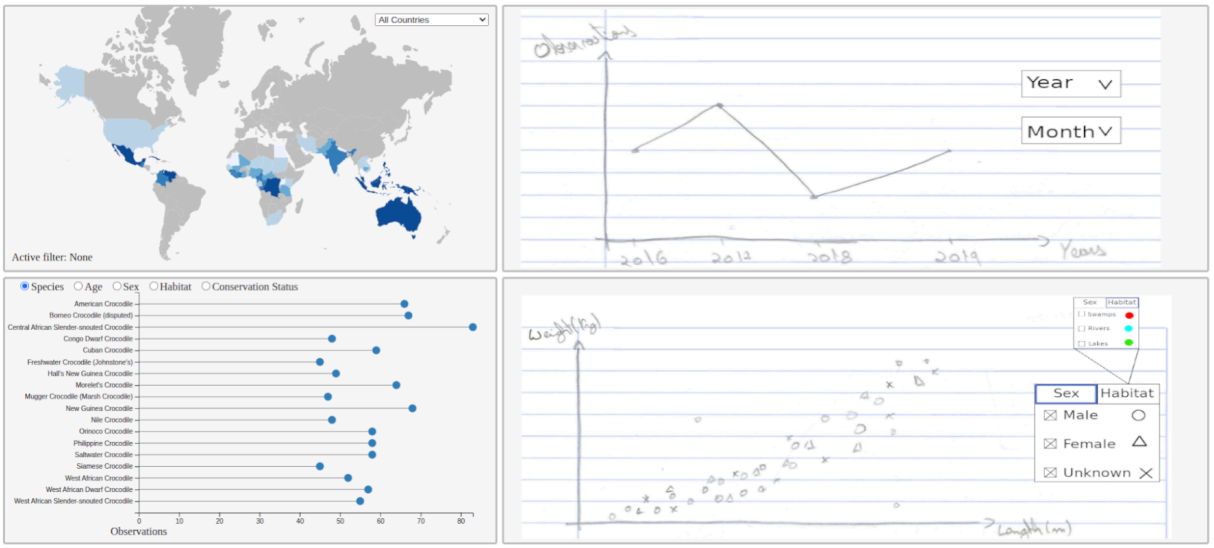
The code is structured into 5 modules:

- liveserver.py: allows for the creation of the live server that runs the rest of the code.
- index.html: loads the modules, creates the grid and applies the overall CSS.
- dataStuff.py: processes the dataset according to how we decided to process it on CP1, it also does some extra processing as described below in data processing.
- stuff.js: includes functions used by all idioms, further explained below in the chart integration and chart integration, also loads the dataset and exports it.
- clev_plot.js: includes all necessary functions to create and render the first idiom
- map.js: includes all necessary functions to create and render the choropleth map.

Initially, and not necessary for running the project, only for the setup, datastuff.py is executed on the original dataset. We do this to clean the dataset and adjust it for our own use. Afterwards, as the live server is being run, index.html is run and this file will load our D3/JS modules. Firstly, stuff.js is loaded and run, and this file will load the dataset onto memory so that the other modules can all access it without having to load it several times. Afterwards, clev_plot.js is loaded and executed to create the

Cleveland Dot Plot. Auxiliary functions are then invoked by the Cleveland Dot Plot, from stuff.js, to help create the chart - everything from counting the number of observations to deciding what items are going to be visible or not. Then, the map.js file is loaded and executed, loading the geoData itself and fetching the dataset from stuff.js to create the Choropleth Map, similarly to the Cleveland Dot Plot with the auxiliary functions from stuff.js. Each of the graphs is created and rendered inside the grid in index.html and in their own containers.

Dashboard Layout



Compared to CP2, we decided to make some changes to the dashboard layout. The way we had set up the dashboard, put a major emphasis on the Choropleth Map, and we believe that to not be the best course of action. The other idioms are better suited to represent the data and help answer the tasks created in CP1, the map works as a union between all idioms and helps the user filter the data by country visually. We still believe it to be an important part of our visualization, so we didn't remove it, only reduced its size. This also improves the readability of the other idioms that will take better advantage of the gained space to better present their more granular and detailed data. To do this we decided to rearrange the dashboard like the image shows. Similarly, we reduced the first idiom size but not as much, reducing the size of this idiom is acceptable since the marks use the position and length as a channel. The Choropeth Map is on the top left; the first idiom (the Cleveland Dot Plot) is on the bottom left; and the second (Line Chart) and third idioms (Scatter Plot) are in the top right and bottom right, respectively.

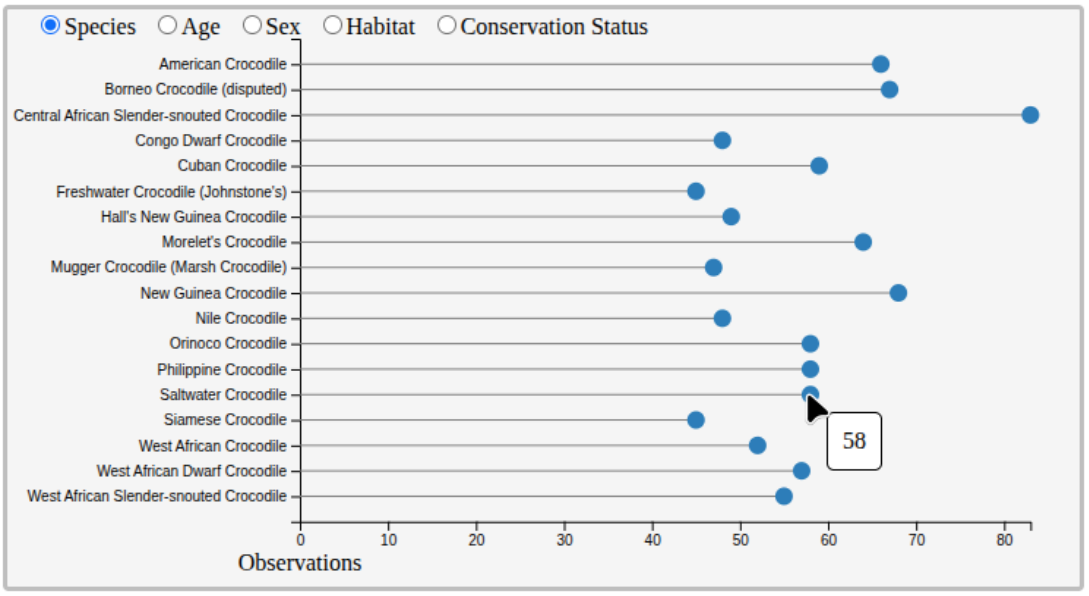
Data Processing

In addition to the data processing explained in CP1, we also had to make another two changes. We first standardized column names, by removing unnecessary spaces, capital letters and information like the units of measurement. We did this to more easily access the information on the columns using D3, since this way we can just access them by referencing the value in the data. Second, we merged certain regions of countries into one single country, because we didn't find that information pertinent for the visualizations and also because the geoData[1] does not support these very specific regions. The "Congo Basin Countries" value is now also shared between "Congo" and "Eq. Guinea" as these are the most accurate labels supported by the geoData. This is all done in the datastuff.py script.

Very minor data processing is done in the actual JS files comparatively. The map.js file creates a dropdown widget with all the countries listed. For this, the list of countries is extracted from the

Dataset and sorted into alphabetical order. An extra value (“All Countries”) is appended to the country list in the dropdown widget, and when selected this shows data for all countries (it is the default setting). Both the map.js and clev_plot.js depend on an observation counter (found in stuff.js) which will take the Dataset and count the number of occurrences, be it by variable (get_count) or by country (get_count_by_country).

Chart Interaction & Integration



The first idiom’s code resides in the clev_plot.js file, and there are several ways that the user can interact with this chart. Above the chart there are a series of radio buttons, the default of which is “Species”. Selecting any of these radio buttons will make the chart change its contents to the observation count of said option. The user can also hover on the dots, and doing so will display a tooltip showing the exact value. In the example above, we can see that there were 58 observations of the Saltwater Crocodile.

The chart also interacts with the map chart, as when we click on the dot, the Choropleth Map will filter to show only the countries that have Saltwater Crocodiles. As such:

The map also has interaction. As can be seen in the image above, the “Active filter” label changes accordingly with what was clicked on the previous chart; hovering over (or choosing a country) will apply a thicker black outline to it; and a tooltip will appear next to the mouse indicating the name of the country and the number of observations (in this case, of Saltwater Crocodiles). The map is also zoomable and pannable and clicking on an empty space (the ocean) will reset its filter. As previously mentioned, it’s possible to click on a country, and the result of that is that the Cleveland Dot Plot is filtered to show information about that country, as in the following image showing crocodile information in Australia:

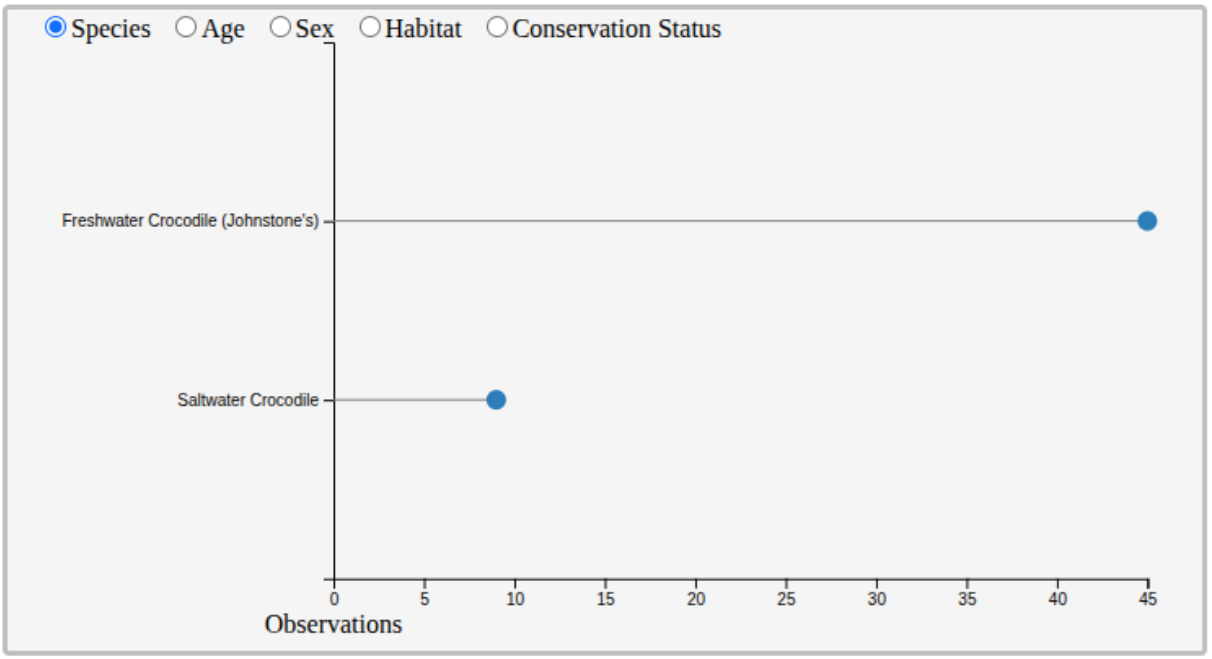


Chart integration, or interaction between the two charts, is done via event dispatching. This makes it very easy to expand and add new modules because all we need to do is listen to new events in the modules. Clicking on a dot in Idiom 1 dispatches an event which is listened to by the Map, and then the Map updates according to the information in the event. Clicking on a country on the Map dispatches an event that is being listened to by Idiom 1, which updates itself to show information about that country.

Resources

1 - geoData (dataset/geo.json) - https://geojson-maps.kyd.au