CSE564: Visualization
Mini Project #1

Priyanka Nath
SBU ID: 112715634

Link to Youtube Video: https://youtu.be/J2BHoFvJCpk

Dataset Selection

I like video games so I decided to work with something related to them. After going through some of the popular sources like the UCI ML repository, Data.gov, etc I decided to work with the VideoGame Sales and Ranking Dataset.

Data Preprocessing

The dataset has 16,719 entries each having 16 features. The data was cleaned based as follows -

1. The entries were not all complete i.e. a large number of entries had at least one unknown NaN column. I used Python pandas to drop the incomplete entries, i.e. entries with at least one NaN column.
2. Some of the categorical features like 'Publishers' and 'Developers' had a huge domain visualizing which in the form of a bar chart would look extremely clumsy. I computed the most frequently occurring top 24 values and changed the other values to 'Others'. This enables us to get a clean bar chart with 25 unique values.
3. Some of the values of the categorical features were too long hence they were shortened manually.
4. The feature 'Name' is unique for each of the entries so I did not use it for visualization.
5. The feature' User_Score' had string entries that were actually floats so the column was converted to numerical data by casting the values to floating-point numbers.
6. The list of numerical and categorical data was obtained.
   - Numerical Features: 'Year_of_Release', 'NA_Sales', 'EU_Sales', 'JP_Sales', 'Other_Sales', 'Global_Sales', 'Critic_Score', 'Critic_Count', 'User_Score', 'User_Count'
   - Categorical Features: 'Rating', 'Platform', 'Publisher', 'Genre', 'Developer'

After these steps, the final clean dataset had 6825 entries and 15 features.

Data Visualization

index.html is the main file to be opened to view the webpage after creating a server

style.css contains the style information for the webpage, mostly the navigation bar
plot_bar.js contains the function to plot bar chars
plot_histogram.js contains the function to plot histograms
start_script.js contains scripts and functions to be executed when web page is viewed
reduced_data.csv contains the dataset after preprocessing
d3-tip-master is the folder for compatible tooltip for D3 version4

1. **Navigation Bar**
   The navigation bar style was adapted from a W3School Template.

   It has 2 dropdown options namely Numerical Features and Categorical Features from which we can select any one to view their corresponding graph plot.

Fig.1. Starting page

The navigation menu is populated using the populateNavigationBar() fucntion in start_script.js. It dynamically adds the menu items and specifies the function to be called when the item is clicked along with the necessary parameters.

Fig.2. Hovering over the menu reveal drop-down menu.

If a Categorical Feature, say, ‘Genre’ is selected then it will show the barplot for the selected feature and if a Numerical Feature is selected, for example, ‘User_Score’, then the histogram for the selected feature will be shown.

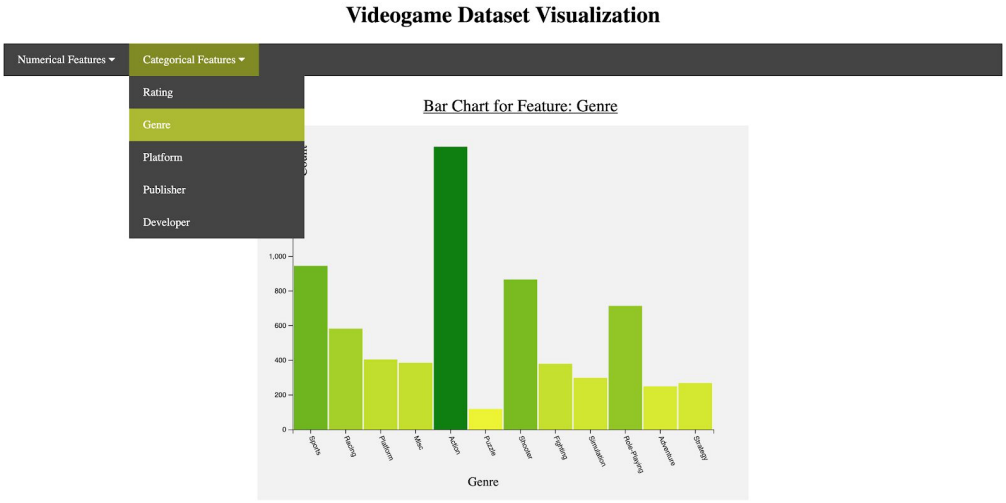

Fig.3. Selecting Genre gives the bar graph for Genre

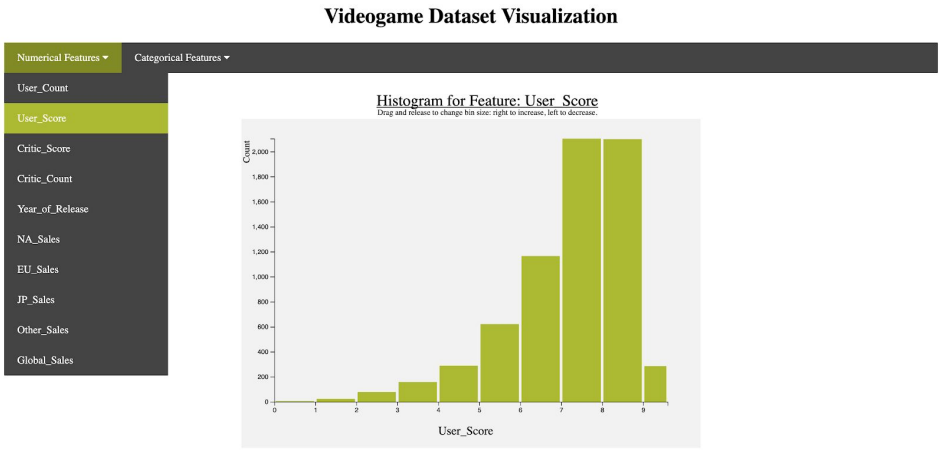

Fig.4. Selecting User_Score gives the histogram for User_score

2. Starting Scripts

Scripts written in start_script.js are executed when the webpage is loaded. The populateNavigationBar( ) as shown in Script.1. is called to populate the dropdown menu in the navigation bar which is called from index.html. The populateNavigationBar( ) function also creates a gray svg object with a prompt written in the heading area as shown in Fig.2. (Script.2.).

```
function populateNavigationBar() {
  const margin = {top: 100, right: 50, bottom: 100, left: 50};
  var width = 600;
  var height = 400;

  var categorical = ['Rating', 'Genre', 'Platform',  'Publisher', 'Developer'];
  var numerical = ['User_Count', 'User_Score', 'Critic_Score', 'Critic_Count', 'Year_of_Release', 'NA_Sales', 'EU_Sales', 'JP_Sale

  d3.select("#nav")
    .select("#cat")
    .select("#cat_content")
    .selectAll("a")
    .data(categorical)
    .enter()
    .append("a")
    .attr("xlink:href", "#")
    .attr("id", function(d) { return d })
    // the function for onclick should look like plotBar(colum_wise_data.<feature_name>, <feature_name>)
    .attr("onclick", function(d) { return "plotBar(column_wise_data." + d + ",\"" + d + "\")"; })
    .text(function(d) { return d });

  var num_bins = 10; // number of bins, default = 10
  d3.select("#nav")
    .select("#num")
    .select("#num_content")
    .selectAll("a")
    .data(numerical)
    .enter()
    .append("a")
    .attr("xlink:href", "#")
    .attr("onclick", function(d) { return "plotHist(column_wise_data." + d + ",\"" + d + "\"," + num_bins+ ")"; })
    .text(function(d) { return d });

```

Script.1. start_script.js - code to dynamically populate the menu

```
  // create a empty svg item with start instructions
  console.log("lets begin!");
  d3.select("#graph_plot").select("svg").remove();
  var svg = d3.select("#graph_plot")
              .append("svg")
              .attr("width", width + margin.right + margin.left)
              .attr("height", height + margin.top + margin.bottom);

  svg.append("rect")
      .attr("id", "display")
      .attr("width", width + margin.right + margin.left)
      .attr("height", height + margin.top)
      .attr("y", margin.right + 20)
      .attr("fill", "#f2f2f2");

  svg.append("text")
       .attr("transform", "translate(50,0)")
       .attr("x", width/2)
       .attr("y", margin.top/2) // x, y coordinate
       .attr("text-anchor", "middle")
       .attr("font-size", "24px")
       .text("Select a feature from the dropdown navigation menu to get started!");
  d3.select("#graph_plot").attr("align","center");

```

Script.2. start_script.js - code to add gray rectangle and prompt

In the start_script.js the data is loaded from the "*reduced_data.csv*" file and stores as a dictionary where the keys are the feature names and the values are arrays of all values of the corresponding feature. This is done so that the graph plot functions can be reused with proper parameters. The code is shown below.

```
// load data and store it in a dictionary
var column_wise_data = {}; // {'<feature_name1>': Array(6825), '<feature_name2>': Array(6825)...}
column_wise_data.Rating = [];
column_wise_data.Publisher = [];
column_wise_data.Developer = [];
column_wise_data.Platform = [];
column_wise_data.Genre = [];

column_wise_data.User_Score = [];
column_wise_data.Year_of_Release = [];
column_wise_data.NA_Sales = [];
column_wise_data.EU_Sales = [];
column_wise_data.JP_Sales = [];
column_wise_data.Other_Sales = [];
column_wise_data.Global_Sales = [];
column_wise_data.Critic_Score = [];
column_wise_data.Critic_Count = [];
column_wise_data.User_Count = [];

d3.csv("reduced_data.csv", function(error, data){
  // throw error
  if (error) {
    throw error;
  }
  data.forEach(function(d, i) { ...
  });
  // console.log("1", column_wise_data);
});
```

Script.3. Start_script.js - code to load csv and store data

3. Visualizing Numerical Data - Drawing Histograms

The javascript code is written in *plot_histogram.js*. The plotHist(*data, column_name, num_bins*) function is used to draw the histogram for the selected feature.

Parameters

- *data* is the array of values of the selected feature
- *column_name* is the name of the selected feature
- *num_bins* is the number of bins in the histogram; default value is 10

Logic

- Remove svg object if it exists.
- Add a new svg object and create a grey rectangle and add a heading along with instructions.
- Define x-scale and draw and name x-axis.
- Set up histogram( ) with proper parameters and use it on the data parament to get binned data (stored in bins).
- Define y-scale and draw and name y-axis.
- Using the d3 version4 compatible tooltip script, define the tool_tip function and call it on svg.
- Get the list of mid-values of each bin and store it.
- Draw rectangles giving them a class name of bar using the binned data to create the histogram.
- In the case of a mouseover event on a rectangle define a function to color it red, increase its height and width and show the number of entries in that bin along with the bin's middle value.

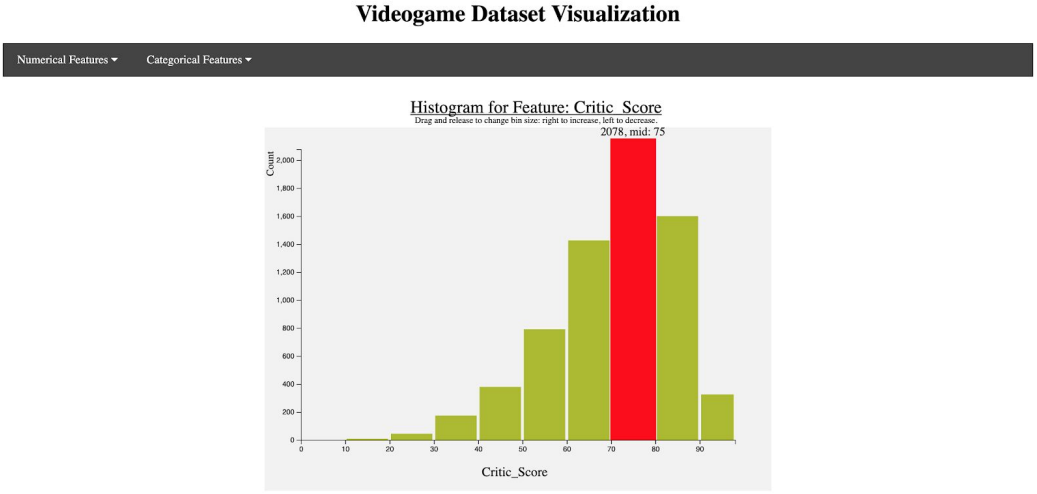

Fig.5. Hovering over a bar shows the count and mid-value of the bin.

- In the case of a mouseout event on a rectangle define a function to revert back to original parameters.
- Center align the svg.
- Define a function for the mousedown event for the grey rectangle which changes bin size based on mouse drag i.e. if the mouse is clicked and dragged to the left, bin size should increase and if dragged to the right then bin size should decrease.

```
// increase and decrease bin size based in mousedrag left and right
// logic - on mousedown define mouseover action, on mouseup make mousedown and mouseover do nothing
// define what to do during mousedown event
d3.select("#display").on("mousedown", function() {
  // get the current x coordinate of mouse
  var x_coordinate = d3.mouse(this)[0];
  // define what to do in case of mousemove and mouseup events when mousedown has happened
  d3.select(this)
    .on("mousemove", updateNumBins)
    .on("mouseup", function() {
      console.log("Bin size changed, plotting with new num_bins", num_bins);
      plotHist(data, column_name, num_bins);
      // set action for mousemove and mouseup as null
      //so that till mousedown happens bin size will not change
      d3.select(this).on("mousemove", null).on("mouseup", null);
    });


  function updateNumBins() {
      // +5 and -5 is done so that the increase and decrease appears gradual
      // mousemove left -> bin size decreases -> number of bins increases
      if(d3.mouse(this)[0] + 5 < x_coordinate && num_bins < 100){
        num_bins += 1;
        console.log("Decrease in binsize, increase in number ->" + num_bins);
        x_coordinate = d3.mouse(this)[0];
      }
      // mousemove right -> bin size increases -> number of bins decreases
      else if(d3.mouse(this)[0] - 5 > x_coordinate && num_bins > 2){
        num_bins -= 1;
        console.log("Increase in binsize, decrease in number ->" + num_bins);
        x_coordinate = d3.mouse(this)[0];
      }
  }
});
```

Script.3. Plot_histogram.js - code to update bin size and bin number

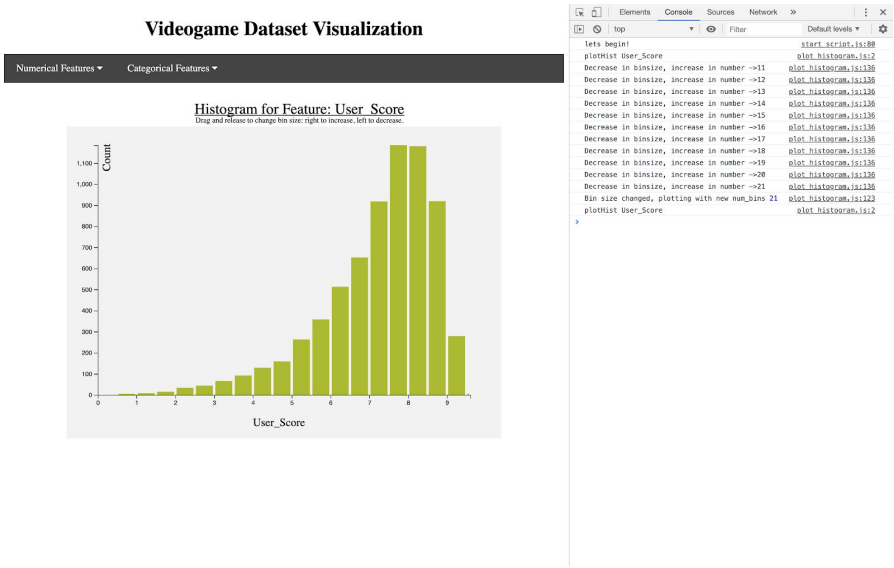

Fig.6. Bin size is decreased, the number of bins increases.

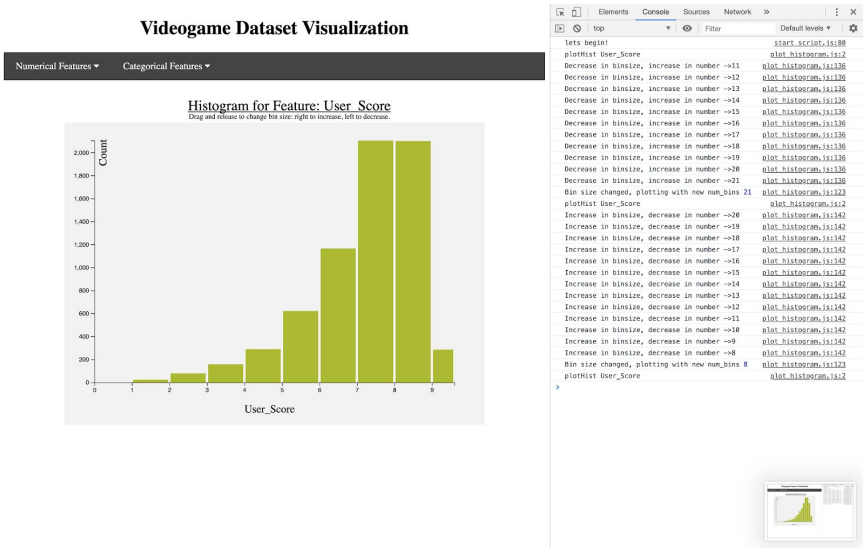

Fig.7. Bin size is increased, the number of bins decreases.

4. Visualizing Categorical Data - Drawing Bar Graphs

The javascript code is written in *plot_bar.js*. The plotBar(*column, column_name*) function is used to draw the histogram for the selected feature.

Parameters

- *column* is the array of values of the selected feature
- *column_name* is the name of the selected feature

Logic

- Remove svg object if it exists.
- Add a new svg object and create a grey rectangle and add a heading.
- Now since it is categorical data it needs to be processed to be plotted. Count the number of occurrences of each of the unique values of the feature and store it in the form of an

array of dictionaries. Each element in the array will have a “key” which stores the unique feature value and a “value” which stores the number of occurrence of the feature value. The code is shown below -

```
// get the count of the unique feature values in form of a dictionary
// data = [ {key:<unique_feature_value1}, value: <number_of_occurrences_of)unique_feature_value1}, ...]
var data = d3.nest() // make it nested datastructure; array of dictionaries
            .key(function(d){ return d;}) // set grouping value
            .rollup(function(d){ return d3.sum(d, function(g) {return 1;});})
            // basically a group by function where we sum the occurrences of a unique entry
            .entries(column);
// console.log(data);
```

Script.3. plot_bar.js - code to group data by unique feature values and count their occurrences.

- From the above array, we obtain an array of just the “key” values (x_labels) and another array of just the ”value” values (counts).
- Define x-scale and y-scale and draw and name the x-axes.
- Using the d3 version4 compatible tooltip script, define the tool_tip function and call it on svg.
- For better aesthetics define a color scale that ranges from yellow to green to with a domain of (0, max_value). Using this color the rectangles in the bar chart, the higher ones will be greener while the smaller ones will be more yellow.

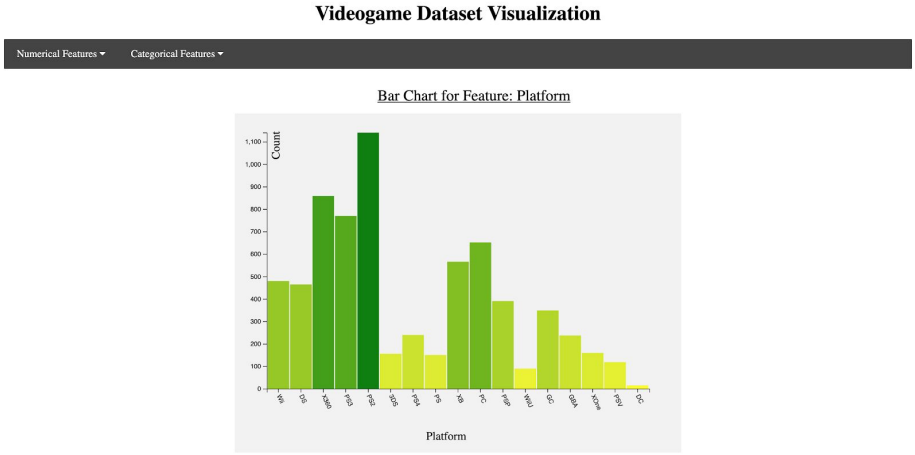

Fig.8. Bar plot using the color scale

- Draw rectangles giving them a class name of bar using the grouped data and color it based on the color-scale to create the bar chart.
- In the case of a mouseover event on a rectangle define a function to color it red, increase its height and width and show the count of the feature value.

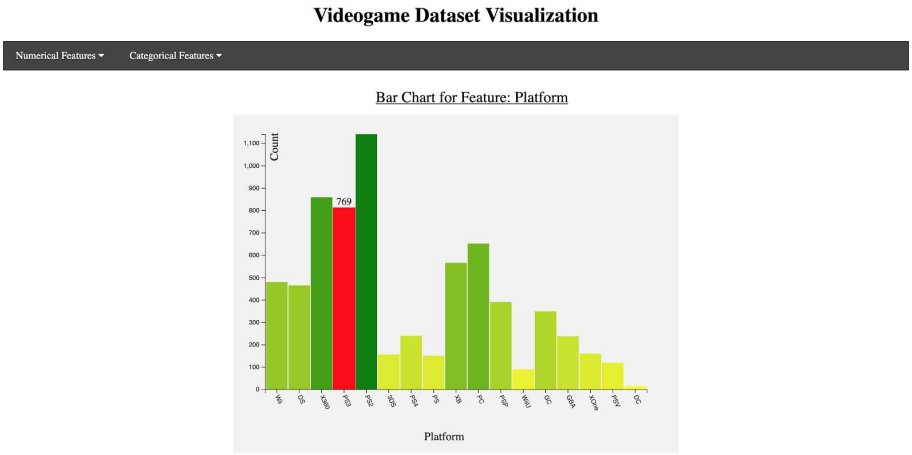

Fig.9. Hovering over a rectangle shows the count of the feature value in the dataset.

- In the case of a mouseout event on a rectangle define a function to revert back to original parameters.
- Center align the svg.

Note: I am using D3 version 4 and since tooltip for D3 did not have compatibility with version 4, I downloaded this compatible tooltip and used that.

References

Dataset: https://www.kaggle.com/gregorut/videogamesales
Navigation Bar: https://www.w3schools.com/howto/howto_css_dropdown.asp
Tooltip: https://github.com/VACLab/d3-tip
Other:
https://www.w3schools.com/
https://d3js.org/
https://medium.com/swlh/data-visualization-with-d3-js-bar-chart-4948d9e85000
https://www.d3indepth.com/scales/
https://bl.ocks.org/d3noob/8952219
https://www.d3-graph-gallery.com/histogram
http://learnjsdata.com/group_data.html