Assignment 3: Data Foundations, Histograms& Tooltip **(20 points)**

Due: 15.05.2023 8AM

Task 1: Data Foundations **(5 Points)**

Task 1a) Data Types: Imagine you work for a fruit trade company, and your job is to create a categorization scheme for apples (e.g., to sell them to customers). Name <u>**three**</u> **descriptive attributes** of **nominal, ordinal, and numerical data type**, and possible value ranges.

An example for cars:
Numerical: Horsepower, 50-500 HP

Attributes for apples:

1. Nominal:
   a. Name: [Gala, Red Delicious, Granny Smith]
   b. Color: [Green, Red, Yellow]
   c. Origin: [Poland, Italy, France]

2. Ordinal:
   a. Taste: [Sweet, Sour, Combination]
   b. Quality: [Poor, Average, Excellent]
   c. Ripeness: [Unripe, Semi-ripe, Fully-ripe]

3. Numeric:
   a. Weight: [50 Grams to 300 Grams]
   b. Diameter: [5cm to 10cm]
   c. Sugar Content: [5% to 10%]

Task 1b) Missing values – When using a **global constant to fill missing values** of an attribute, why should we <u>not</u> use this value for further calculations? Can you think of a situation where it might be useful to consider missing values?

Answer: We should not use global constants for further calculations because it could distort the distribution of data. The constant might not accurately represent the measure of the attribute leading to incorrect analysis. Suppose we request our customers to rate our app and provide feedback. In some cases, customers may give bad ratings without providing any feedback, resulting in missing values. However, considering these missing values can be beneficial to us as they suggest that the customer had no specific reason for giving a negative rating and may potentially be bluffing.

Task 1c) Binning – In which cases should we prefer equal-depth binning compared to equal-width binning?

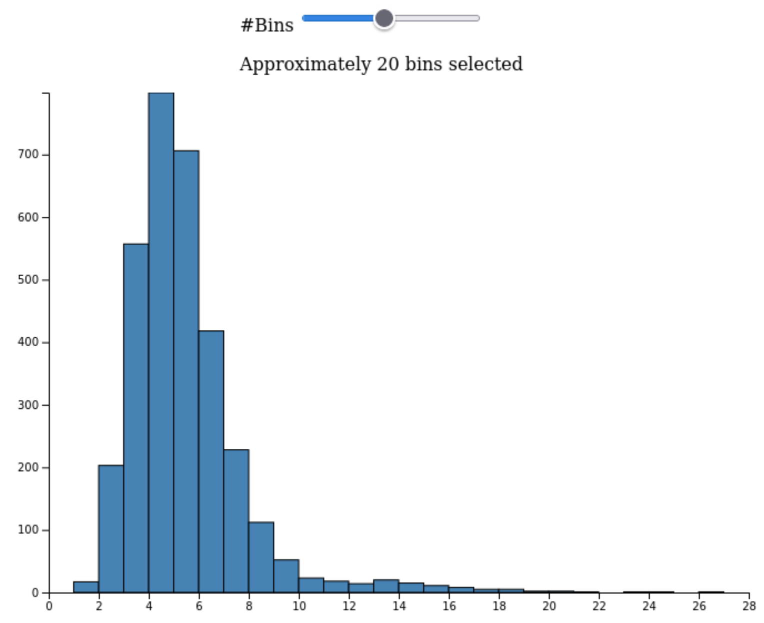
Answer: We should prefer equal-depth binning compared to equal-width binning in the following cases:

1. When data distribution is skewed.
2. When data density is not uniform.
3. When we want to focus on data distribution.
4. When we want to handle outliers.

- ➔ **After putting in your answers, export the docx-File to PDF and upload it alongside the source code files.**

- ➔ **Continue with the programming tasks on the next page, you do not need to change this document any further**

Task 2: Programming – Creating a Histogram (10 Points)



Goal of this exercise is to implement a histogram with D3. Attached to this exercise you will find a folder called *histogram*which contains an unfinished implementation of a histogram. Your task is to finish the implementation such that opening the *index.html* shows a histogram as depicted in figure above. This example shows the unemployment rate of U.S. counties as of August 2016. To finish the implementation, follow the steps described as comments within the dedicated file. Each comment starting with *TASK* indicates a position you must add code. Within the folder *histogram* there are 4 files:

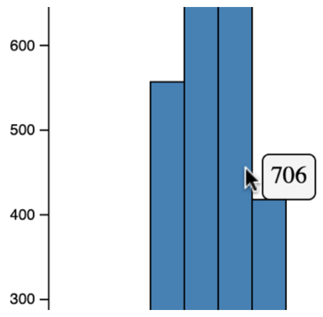
- **index.html**
  The main entry point of the visualization. When your implementation is finished, opening this file with a browser should show a *linechart*. Currently, opening the *index.html* will show a heading but no visualization.
- **index.js**
  The main JavaScript entry point which implements a line-chart.
- **index.css**
  Implements CSS Rules for specific elements.
- **data.js**
  Initializes a variable called *data* and reflects the dataset we want to visualize. It represents unemployment rate of U.S. counties as of August 2016.

Required steps:

- Implement the **update(numbins)**function.
- Define x- and y-scales
- Created binned data from the raw values, be sure to utilize the *numbins* attribute we pass to the function
- For each entry of bin, draw a rectangle by specifying x, y, width and height

- Styling from the CSS should be automatically applied to your added rectangle elements
- We want to be able to dynamically update the data. **Be sure to remove old rectangles before adding new ones!**
- Finally, add axes

Task 3: Programming – Adding mouse effects & tooltip (5 Points)



Finally, we want to add a tooltip element to our visualization. When hovering over a rectangle, we should see a little tooltip showing the current value for each bin. Follow Task 3 in the code (also part of the update function)

Required steps:

- Add an empty div element to the body, give it the id “tooltip” (which should automatically do the styling via CSS)
- For the rectangles, add event listeners (via .on()) for **mouseover, mousemove and mouseout**. Tipp 1: Remember how you can show or hide elements via CSS. Tipp 2: The data of a d3 element is passed as the second parameter of the on callback function *(d3Element.on(“eventname”), (event, data) =>console.log(“This is the data”, data))*

Submission: Zipped folder including all files of the programming exercise (index.html, index.js, index.css, data.js) and a PDF of the completed written exercise.

Please find yourself in Groups of **2 Students**. Only 1 member of the group must submit the exercise in ILIAS.