**Demonstrated on 04/02/2022 to Amit Parekh.**

Introduction

The following report will provide screenshots of the answers and, when relevant, context to the code developed. Every exercise is self-contained and can be run on a browser. When requested, answers can be found in the console log, and I will highlight when this occurs. To test my code is working I used the Visual Studio extension Live Server.

I have committed all my answers to a public GitHub repository which can be accessed here: <https://github.com/JoshYang1/F21DV-Data-Visualisation-and-Analytics>

Exercise 1

Text

Description automatically generated

Exercise 2

Graphical user interface, application, Word

Description automatically generated

Exercise 3

Chart

Description automatically generated with low confidence

Exercise 4

Chart

Description automatically generated with low confidence

    // The div elements have been created and now we select the first element and edit it's text and style

    d3.select('div').text("start").style("color", "purple")

The loop creates the 10 div elements and once the loop has finished, we then select the first div element and change the text and styling.

Exercise 5

A picture containing text

Description automatically generated

Exercise 6

Table

Description automatically generated

Exercise 7

Text

Description automatically generated with medium confidence

Exercise 8



Exercise 9

Graphical user interface, text, application

Description automatically generated

As can be seen from the above screenshot, the count of Mr, Mrs and Other did not return a perfect answer. The reason for this is the split by whitespace function;

            // splitting the Name element, separated by a blank space

            var title = d.Name.split(" ")

            // adding the title of the passenger to an array which should be the element at index 1

            arrayN.push(title[1])

The code is splitting the Name element in the object and creating an array. We then push the first element of the newly created array which should be the title of the person but is not perfect as can be seen by the results, for example, people may have two surnames.

            const counts = {};

            for (var i = 0; i < arr.length; i++) {

                // the key of counts object is based off the array parameter provided and increment the count

                counts[arr[i]] = 1 + (counts[arr[i]] || 0);

            };

We created a function to count the number of values for each unique key. An object is created and then a key is created with the key of the array that has been passed in. If that key does not exist, the count is initiated but if it already exists then we add a 1 to the value of that key.

Exercise 10

Text

Description automatically generated

            //looping through the data to count per age bracket

            for (var i = 0; i < arr.length; i++) {

                // the key of counts object is based off the array parameter provided and increment the count

                if (arr[i].age <= 30) {

                    //count is either zero or already initialised

                    counts['1 - 30'] = 1 + (counts['1 - 30'] || 0)

                } else if (arr[i].age >= 31 && arr[i].age <= 40) {

                    counts['31 - 40'] = 1 + (counts['31 - 40'] || 0)

                } else if (arr[i].age >= 41 && arr[i].age <= 60) {

                    counts['41 - 60'] = 1 + (counts['41 - 60'] || 0)

                } else {

                    counts['61 - 100'] = 1 + (counts['61 - 100'] || 0)

                }

            };

            return counts;

        };

We created a loop to check the age and then store the count of each person in the relevant age group.

        // looping through each key in the object and creating a 'p' HTML element

        Object.keys(output).forEach(key => {

            // setting the text of the element to the key and value

            svg.insert("p").text("Age Bracket: " + key + " Count :" + output[key]);

        })

We then loop through each of the keys from the newly created object and insert a ‘p’ element with the required information.

Exercise 11

A picture containing box and whisker chart

Description automatically generated

Exercise 12

Chart

Description automatically generated

                switch (d.Shape) {

                    case 'circle':

                        // Create and append circle

                        svg.append("circle")

                            .attr("cx", d.Positionx)

                            .attr("cy", d.Positiony)

                            .attr("r", d.Radius)

                            .attr("fill", d.Color);

                        break;

We created a switch statement which is dependent on the Shape key from the csv file.

Exercise 13

                            .join(

                                    enter => enter.append(line.Shape)

                                                .attr("cx", line.Positionx)

                                                .attr("cy", line.Positiony)

                                                .attr("r", line.Radius)

                                                .attr("fill", line.Color),

                                    exit => exit.transition()

                                                .duration(5000)

                                                .attr('r', 0)

                                                .remove(),

The join lets us specify exactly what happens to the DOM. Using the enter function, new shapes are added to the svg. Exit will then remove the shape from the svg.

Exercise 14 & 15

We struggled to complete this exercise as we could not access the data object easily. We know the issue is with the below:

        var g = svg.selectAll("g")

            .data(output)

            .enter()

            .append("g")

            .attr("transform", function (d, i) {

                console.log(d)

                return "translate(0," + i \* barHeight + ")";

            });

Exercise 16

Chart, icon

Description automatically generated

Exercise 17

A picture containing rectangle

Description automatically generated

Exercise 18

A picture containing bar chart

Description automatically generated

Exercise 19

Chart, bar chart

Description automatically generated

Exercise 20

Table

Description automatically generated with medium confidence

The translate transform attribute of g specifies the number of horizontal and vertical pixels by which to translate the element.

Exercise 21

Chart, bar chart

Description automatically generated

Created a container element so we could centre for cleaner display.

Exercise 22

Chart, line chart

Description automatically generated

Exercise 23

Chart

Description automatically generated

Exercise 24

Chart, histogram

Description automatically generated

Exercise 25

Chart, line chart

Description automatically generated

Exercise 26

A picture containing text, line, different, several

Description automatically generated

        // Hand drawn triangle rather than using symbolTriangle

        // https://stackoverflow.com/questions/60029911/change-symbol-for-graph

        var symbol = function() {

            return d3.create('svg:path').attr("d","M0,8L-5,-3L5,-3Z").node()

        }

        // Appending the triangle to every sine data point

        // https://stackoverflow.com/questions/33881962/triangle-scatter-plot-with-d3-js

        svg.append("g").selectAll(".symbol")

            .data(data.sine)

            .enter()

            .append(symbol)

            .attr("class", "symbol")

            // Fixing the symbol the coordinates of the data

            .attr("transform", function(d) { return "translate("+x(d.x) + "," + y(d.y) +")" })

    };

We tried to implement the symbolTriangle icon but opted to use a hand a drawn triangle instead.

Exercise 27

A picture containing text, boat, line, day

Description automatically generated

        // Adding text to the data points by creating a new class and appending text

        // https://stackoverflow.com/questions/12266967/d3-js-how-to-add-labels-to-scatter-points-on-graph

        svg.selectAll(".dodo")

            .data(data.cosine)

            .enter().append("text")

            .attr("class", "dodo")

            .attr("x", function(d) { return x(d.x); })

            .attr("y", function(d) { return y(d.y); })

            .attr("dx", ".71em")

            .attr("dy", ".35em")

            // Only append text to every 10th data point

            .text(function(d,i) { if (i % 10 == 0) {return i}});

Created a new class and appended text. The dataset is the same as the line so it will be associated correctly.

Exercise 28

Icon

Description automatically generated

    // https://stackoverflow.com/questions/41848677/how-to-make-a-color-scale-in-d3-js-to-use-in-fill-attribute

    g.append("rect")

        .attr("width", function (d) {

            return scale(d);

        })

        .attr("height", barHeight - margin)

        // Every datapoint is given a colour from the scheme based on its value

        .attr("fill",  d => myColor(d))

Exercise 29

A picture containing text, boat, line, different

Description automatically generated

Exercise 30

Chart, pie chart

Description automatically generated

Exercise 31

Chart, pie chart

Description automatically generated

    // Appending text to each arc

    // https://www.d3-graph-gallery.com/graph/pie\_annotation.html

    arcs.append("text")

        .text(function(d) {return d.value})

        .attr("transform", function(d) { return "translate(" + arc.centroid(d) + ")";  })

        .style("text-anchor", "middle")

        .style("font-size", 17);

Exercise 32

A picture containing text, transport, bicycle

Description automatically generated

        // Retrieve the bounds of the container element

        // https://stackoverflow.com/questions/24534988/d3-get-the-bounding-box-of-a-selected-element

        console.log(d3.select('#container').node().getBoundingClientRect())

        svg.append("svg:image")

            .attr("xlink:href", filepath)

            .attr("width", xMax)

            .attr("height", d3.select('#container').node().getBoundingClientRect().height)

We had an issue stretching the image (may be with the aspect ratio of the image) and tried several troubleshooting techniques to resolve it but to no avail. Although, one positive, discovered through our research, is a function that will give the bounds of any element so that the dimensions can be inferred.