**Belly Button Biodiversity Dashboard using Plotly ...The coolest study of biodiversity on the human body on the planet! Really!**

The belly button is one of the habitats closest to us, and yet it remains relatively unexplored. In January 2011, we launched *Belly Button Biodiversity* to investigate the microbes inhabiting our navels and the factors that might influence the microscopic life calling this protected, moist patch of skin home. In addition to inspiring scientific curiosity, *Belly Button Biodiversity* inspired conversations about the beneficial roles microbes play in our daily lives.

**#Overview**

Plotly provides graphing, analytics, and statistics tools, as well as scientific graphing libraries for Python, R, MATLAB, Perl, Julia, Arduino, and REST. Plotly is an open-source library that provides a list of chart types as well as tools with callbacks to make a dashboard. The charts we have constructed and embedded are all made in chart studio of plotly.

When Mrs. Roza came to us, she had partially completed a dashboard. A completed panel for demographic information allows for a user-friendly convenience. We have constructed a chart to visualize the bacterial data for each volunteer allowing them to identify the top 10 bacterial species in their belly buttons. Furnishing this information will allow Improbable Beef the ability to identify a species, as a candidate, to manufacture synthetic beef. Roza's volunteers will be able to identify whether that species is found in their navel.

The interactive dashboard explores the In this assignment, you will build an interactive dashboard to explore the Belly Button Biodiversity dataset, which catalogs the microbes that colonize human navels. The dataset reveals that a small handful of microbial species (also called operational taxonomic units, or OTUs, in the study) were pA close up of a toy

Description automatically generated with low confidenceresent in more than 70% of people, while the rest were relatively rare.

Elements created: or rather what lives in baby’s belly button

1. Create a Horizontal Bar Chart
2. Create a Bubble Chart
3. Create a Gauge Chart
4. Customize the Dashboard
5. A written report on the Belly Button Biodiversity Dashboard analysis

**## Resources**

* Data Source: BellyButton\_bar\_chart\_starter\_code.js, BellyButton\_bubble\_chart\_starter\_code.js, BellyButton\_bubble\_chart\_starter\_code.js and index.html
* Data Tools: ECMAScript, JavaScript, JSON and IO (Web Server)
* Software: ES6+, ECMAScript and Visual Studio Code 1.50.0\

**###Overview of ES6+**

To understand some of the reasoning behind our choices of resources and coding decisions, we have provided some background. ECMAScript, also referred to as "ES," is a scripting language designed to help standardize JavaScript. This means that ES provides guidelines and rules for JavaScript to follow, such as how a function should be created to run correctly, also known as the **proper syntax**.

Because ES has provided standardization for JavaScript, it also brings updates to the language. There are updates to every major coding language that fix bugs, update code, and provide overall quality of life improvements for the developers. ES6 is no exception as it had worked wonders for us!

Many updates have occurred to ES over the years, though the sixth update was a major one. You'll probably see "ES6+" mentioned out in the wild pretty often; this is a reference to the "big" update (ES6) as well as the later ones. It's also commonly known as "ES2015" or "ECMAScript 2015." (It was such an important update that it's even known by its year!) There are quite a few different ways to reference this language, but we'll be referring to it as ES6, JavaScript, or JS in this project.

**###Benefits of the ES6 Update**

We've briefly mentioned that the ES6 update was useful, but let's talk a bit more about why it was such a big deal. Imagine two laptops, one old and one new, side by side. They're similar enough: they're close in size and shape and can complete many of the same tasks, but the newer laptop has an edge. It is faster and can perform tasks with greater efficiency than the older model.

JavaScript after the ES6 update is like the newer computer. This update included many updates to the syntax, which streamlined the code and made it easier to both read and write. Additional, quality of life improvements were implemented as well, such as adding Python-like generators and for...of loops. Even functions were updated and streamlined!

**### JavaScript - functionality**

The map() method in JavaScript applies a transformation to each element in an array. Like a for loop, it can perform an operation to every element of an array. Here is an example in which all the numbers of an array are doubled:

var numbers = [1,2,3,4,5];

var doubled = numbers.map(function(num){

return num \* 2;

});

console.log(doubled);

In this code, an array named numbers contains five integers: var numbers = [1,2,3,4,5];. Let's break down the rest of the code in more detail:

* The numbers array calls the map() method.
* Inside the map() method, there is another function. This function is anonymous, meaning that the function does not have a name. When map() is called, it in turn calls this anonymous function.
* The anonymous function takes a parameter, named num, and returns the number multiplied by 2. Its sole task is to perform this single action.
* For every element in the array, the map() method calls the anonymous function, which doubles the value of the element.
* The map() method returns an array of doubled values, which is assigned the variable doubled.

Here, the map() function becomes a method of the numbers array. It then takes in an anonymous function whose sole task is to double the value of num, its argument.

Behind the scenes, an iterative process similar to a for loop takes place. The anonymous function takes in each integer of the numbers array and doubles it. Finally, the variable doubled is an array of integers whose values are twice their original values.

Try running the code in your browser console and view the results for doubled. You should see the following:

A picture containing calendar

Description automatically generated

**### The filter() Method**

Another functional programming technique is the filter() method. Like the map() method, it accepts another function as its parameter. Like map(), filter() performs an operation on every element in the original array. Unlike map(), however, filter() does not necessarily return an array whose length is the same as the original array.

Let's see what this means in an example. Run the following code in your console. What does larger return?

var numbers = [1,2,3,4,5];

var larger = numbers.filter(function(num){

return num > 1;

});

console.log(larger);

It returns an array of integers that are larger than 1: [2,3,4,5].This example is remarkably similar to the last one, with one major difference.

First, the similarities:

* The numbers array uses the filter() method.
* The filter() method, in turn, takes an anonymous function as its argument. The anonymous function's sole task is to take in a parameter, called num.

The filter() method operates on each element of the numbers array. So how does it differ from map()?

The map() method transforms every element of the original array, and so the size of the transformed array is the same as that of the original array.

The filter() method, on the other hand, returns an array of values that meet certain criteria. Values in the original array that do not fulfill the condition are filtered out. In this case, specifically, the anonymous function called by filter() returns true if an argument is larger than 1, and false if it does not. The filter() method runs the anonymous function on every element of the original numbers array. Only numbers that are larger than 1 are returned: [2,3,4,5]. So whereas applying map() to the numbers array would have returned an array with five elements, applying this specific filter returned an array of only four elements.

**### The Arrow Functions**

Let's do a quick review of arrow functions. An arrow function in JavaScript is syntactic sugar. That is, an arrow function does the same thing as a standard JavaScript function, but it streamlines the syntax used to accomplish the same task.

The anonymous function inside map() and filter() can be simplified as an arrow function. Here's an example:

var numbers = [1,2,3,4,5];

var doubled = numbers.map(num => num \* 2);

console.log(doubled);

The map() method performs the identical operation as before: it doubles each element in the numbers array. However, the anonymous function inside map() has been replaced by an arrow function. Contrast the two:

var familyAge = [3,2,39,37,9];

var sortedAge = familyAge.sort((a,b) => a - b);

console.log(sortedAge);

sortedAge returns the array [2,3,9,37,39]. Like map() and filter(), sort() takes in an anonymous function. During each iteration, the anonymous function, an arrow function in this case, compares one element of the array (a) with another element in the array (b). From a, it subtracts b. If the result is negative (i.e., b is larger than a) then it stays put. If the result of the subtraction is positive, the order of the two elements is reversed. Look at a modified version of this example.

**### The slice() Method**

Roza also needs to be able to select a subset of the data. In her project, for example, she might perform a transformation on an array, filter it, sort it, and then display only the top five results.

var integers = [0,1,2,3,4,5];

var slice1 = integers.slice(0,2);

In this example, the slice() method returns the first two elements of the integer array: [0,1]. The first argument is the position of where to begin the selection. Here, it is at index position 0. The next argument, 2, denotes the position of the array where the slicing ceases. In other words, the slice() method begins selecting the array at index position 0, and stops right before reaching index position 2. So here, it returns elements at index positions 0 and 1, but not 2.

Let's move on!

**## Element created:**

**### 1: Create a Horizontal Bar Chart**

Using JavaScript, Plotly, and D3.js, we created a horizontal bar chart to display the top 10 bacterial species (OTUs) when an individual’s ID is selected from the dropdown menu on the webpage. The horizontal bar chart displays the sample\_values as the values, the otu\_ids as the labels, and the otu\_labels as the hover text for the bars on the chart.

Chart, bar chart, funnel chart

Description automatically generated**Code and Image**

1. Code is written to create the arrays when a sample is selected from the dropdown menu.
2. Code is written to create the trace object in the buildCharts() function, and it contains the following:
   * The y values are the otu\_ids in descending order.
   * The x values are the sample\_values in descending order
   * The hover text is the otu\_labels in descending order.
3. ​Code is written to create the layout array in the buildCharts() function that creates a title for the chart.
4. When the dashboard is first opened in a browser, **ID 940**’s data should be displayed in the dashboard, and the bar chart has the following:
   * The top 10 sample\_values are sorted in descending order
   * The top 10 sample\_values as values
   * The otu\_ids as the labels

**Code and Image**

Chart

Description automatically generated

**### 2: Create a Bubble Chart**

Using your knowledge of JavaScript, Plotly, and D3.js, create a bubble chart that will display the following when an individual’s ID is selected from the dropdown menu webpage:

* The otu\_ids as the x-axis values.
* The sample\_values as the y-axis values.
* The sample\_values as the marker size.
* The otu\_ids as the marker colors.
* The otu\_labels as the hover-text values.

1. The code for the trace object in the buildCharts(); function does the following:

* + Sets the otu\_ids as the x-axis values
  + Sets the sample\_values as the y-axis values
  + Sets the otu\_labels as the hover-text values
  + Sets the sample\_values as the marker size
  + Sets the otu\_ids as the marker colors

2. The code for the layout in the buildCharts(); function does the following:

* + Creates a title
  + Creates a label for the x-axis
  + The text for a bubble is shown when hovered over

3. When the dashboard is first opened in a browser, ID 940’s data should be displayed in the dashboard. All three charts should also be working according to their requirements when a sample is selected from the dropdown menu.

**Code and Image**

Chart, bubble chart

Description automatically generated

**### 3: Create a Gauge Chart**

Finally we created a gauge chart that displays the weekly washing frequency's value, and display the value as a measure from 0-10 on the progress bar in the gauge chart when an individual ID is selected from the dropdown menu.

**Code and Image**

A picture containing diagram

Description automatically generated

1. The code to build the gauge chart does the following:
   * Creates a title for the chart.
   * Creates the ranges for the gauge in increments of two, with a different color for each increment.
   * Adds the washing frequency value on the gauge chart.
   * The indicator shows the level for the washing frequency on the gauge.
   * The gauge is added to the dashboard.
   * The gauge fits in the margin of the <div> element.
2. When the webpage loads, the bar and bubble chart are working according to the requirements in Deliverable 1 and 2, respectively, and the gauge chart is working according to the requirements listed for this Deliverable