**MODULE 12**

**Background**

Roza has a partially completed dashboard that she needs to finish. She has a completed panel for demographic information and now needs to visualize the bacterial data for each volunteer. Specifically, her volunteers should be able to identify the top 10 bacterial species in their belly buttons. That way, if Improbable Beef identifies 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.

**What You're Creating**

This new assignment consists of four technical analysis deliverables. You will submit the following:

* Deliverable 1: Create a Horizontal Bar Chart
* Deliverable 2: Create a Bubble Chart
* Deliverable 3: Create a Gauge Chart
* Deliverable 4: Customize the Dashboard

**Files**

Use the following links to download the Challenge starter codes.

**Deliverable 1: Create a Horizontal Bar Chart (35 points)**

**Deliverable 1 Instructions**

Using your knowledge of JavaScript, Plotly, and D3.js, create 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 will display 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.

Your bar chart should look like the following image:

IMAGE HERE

For this deliverable, you’ve already done the following in this module:

* [**Lesson 12.1.3:**](https://courses.bootcampspot.com/courses/1687/pages/12-dot-1-3-create-a-bar-chart) Create a bar chart
* [**Lesson 12.2.1:**](https://courses.bootcampspot.com/courses/1687/pages/12-dot-2-1-functional-javascript) Use JavaScript functions and methods: map(), filter(), reverse(), and slice().
* [**Lesson 12.2.2:**](https://courses.bootcampspot.com/courses/1687/pages/12-dot-2-2-practicing-javascript-methods) Create a bar chart with filtered arrays
* [**Lesson 12.3.2:**](https://courses.bootcampspot.com/courses/1687/pages/12-dot-3-2-load-a-json-file-with-d3-dot-json) Load a JSON file with d3.json()
* [**Lesson 12.4.3:**](https://courses.bootcampspot.com/courses/1687/pages/12-dot-4-3-belly-button-demographics-panel) Create a function that reads in json data
* [**Lesson 12.4.3:**](https://courses.bootcampspot.com/courses/1687/pages/12-dot-4-3-belly-button-demographics-panel) Write code to use the ID number to create the sample’s information on a panel or chart
* [**Lesson 12.5.1:**](https://courses.bootcampspot.com/courses/1687/pages/12-dot-5-1-deploy-the-project-to-github-pages) Deploy your project to GitHub Pages

Download the BellyButton\_bar\_chart\_starter\_code.js, add it to the js folder of your GitHub pages (GitHub.io) folder, and rename the file charts.js. Use the instructions below to add code where indicated by the numbered-step comments in the starter code file.

In Steps 3-6, you’ll initialize variables that hold arrays for the sample that is selected from the dropdown menu on the webpage.

**IMPORTANT**

Make sure that you use console.log() to help debug any issues.

1. In Step 1, we’ve provided the code for the buildCharts(); function that contains the argument sample, which is the sample that is selected from the dropdown menu.
2. In Step 2, we’ve provided the code to retrieve the samples.json file using the d3.json().then() method.
3. In Step 3, create a variable that has the array for all the samples.
4. In Step 4, create a variable that will hold an array that contains all the data from the new sample that is chosen from the dropdown menu. To retrieve the data from the new sample, filter the variable created in Step 3 for the sample id that matches the new sample id chosen from the dropdown menu and passed into the buildCharts() function as the argument.
5. In Step 5, create a variable that holds the first sample in the array.

**NOTE**

You can combine Steps 4 and 5 as one line of code, but make sure you use the correct variable name for Step 6 when retrieving the array data.

1. In Step 6, create variables that have arrays for otu\_ids, otu\_labels, and sample\_values.
2. In Step 7, create the yticks for the bar chart.

HINT: Chain the slice() method with the map() and reverse() functions to retrieve the top 10 otu\_ids sorted in descending order.

In Steps 8-10, create the trace object, the layout, and Plotly.newPlot() function for the horizontal bar chart.

1. In Step 8, create the trace object for the bar chart, where the x values are the sample\_values and the hover text for the bars are the otu\_labels in descending order.
2. In Step 9, create the layout for the bar chart that includes a title.
3. In Step 10, use the Plotly.newPlot() function to plot the trace object with the layout.

After you have completed the coding requirements, your dashboard will look like this image when it loads for the first time:

IMAGE HERE

**Deliverable 1 Requirements**

You will earn a perfect score for Deliverable 1 by completing all requirements below:

* Code is written to create the arrays when a sample is selected from the dropdown menu **(10 pt)**
* Code is written to create the trace object in the buildCharts() function, and it contains the following: **(10 pt)**
  + 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.
* Code is written to create the layout array in the buildCharts() function that creates a title for the chart **(5 pt)**
* 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: **(10 pt)**
  + The top 10 sample\_values are sorted in descending order
  + The top 10 sample\_values as values
  + The otu\_ids as the labels

**Deliverable 2: Create a Bubble Chart (30 points)**

**Deliverable 2 Instructions**

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.

Your bubble chart should look like the following image:

For this deliverable, you’ve already done the following in this module:

* [**Lesson 12.2.2**](https://courses.bootcampspot.com/courses/1687/pages/12-dot-2-2-practicing-javascript-methods) Create a trace object and layout for a chart
* [**Lesson 12.5.1:**](https://courses.bootcampspot.com/courses/1687/pages/12-dot-5-1-deploy-the-project-to-github-pages) Deploy your project to GitHub Pages

Download the BellyButton\_bubble\_chart\_starter\_code.js file, copy the starter code from Steps 1-3, and add it to your charts.js file after Step 10 for Deliverable 1.

Use the variables that were created in Deliverable 1 to populate the bubble chart. Then, use the instructions below to write the code for the trace object, the layout, and Plotly.newPlot() function to create the bubble chart.

1. To create the trace object for the bubble chart do the following:
   * Assign the otu\_ids, sample\_values, and otu\_labels to the x, y, and text properties, respectively.
   * For the mode and marker properties, the mode is "markers" and the marker property is a dictionary that defines the size, color, and colorscale of the markers.

If you’d like a hint on how to create a trace object for a bubble chart, that’s totally okay. If not, that’s great too. You can always revisit this later if you change your mind.

HINT: Using d3.select(), you can select the element that has changed and retrieve the property and HTML id that have changed.

Check out the Plotly [bubble chart documentation](https://plotly.com/javascript/bubble-charts/#hover-text-on-bubble-charts)

1. To create the layout for the bubble chart, add a title, a label for the x-axis, margins, and the hovermode property. The hovermode should show the text of the bubble on the chart when you hover near that bubble.

If you’d like a hint on how to create a layout for a bubble chart, that’s totally okay. If not, that’s great too. You can always revisit this later if you change your mind.

Using d3.select(), you can select the element that has changed and retrieve the property and HTML id that have changed.

Check out the Plotly [layout object documentation](https://plotly.com/python-api-reference/generated/plotly.graph_objects.Layout.html)

1. Lastly, use the given Plotly.newPlot() function to plot the trace object and layout.

After you have completed the coding requirements, your dashboard will look like the image below when it loads for the first time, with the bar chart you created in Deliverable 1 and the bubble chart.

IMAGE HERE

**Deliverable 2 Requirements**

You will earn a perfect score for Deliverable 2 by completing all requirements below:

* The code for the trace object in the buildCharts(); function does the following: **(10 pt)**
  + 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
* The code for the layout in the buildCharts(); function does the following: **(10 pt)**
  + Creates a title
  + Creates a label for the x-axis
  + The text for a bubble is shown when hovered over
* 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 **(10 pt)**

**Deliverable 3: Create a Gauge Chart (20 points)**

**Deliverable 3 Instructions**

Using your knowledge of JavaScript, Plotly, and D3.js, create 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.

Your gauge chart should look similar to the following image:

IMAGE HERE

For this deliverable, you’ve already done the following in this module:

* [**Lesson 12.2.2:**](https://courses.bootcampspot.com/courses/1687/pages/12-dot-2-2-practicing-javascript-methods) Create a trace object and layout for a chart
* [**Lesson 12.5.1:**](https://courses.bootcampspot.com/courses/1687/pages/12-dot-5-1-deploy-the-project-to-github-pages) Deploy your project to GitHub Pages

Download the BellyButton\_gauge\_starter\_code.js, using Steps 1-3 in the buildCharts() function initialize variables that hold arrays for the sample that is selected from the dropdown menu on the webpage.

1. In Step 1, create a variable that filters the metadata array for an object in the array whose id property matches the ID number passed into buildCharts() function as the argument.
2. In Step 2, create a variable that holds the first sample in the array created in Step 2.

**NOTE**

You can combine Steps 1 and 2 as one line of code, but make sure you use the correct variable name for Step 3 when retrieving the washing frequency value.

1. In Step 3, create a variable that converts the washing frequency to a floating point number.
2. In Step 4, create the trace object for the gauge chart.

If you’d like a hint on how to create a gauge chart, that’s totally okay. If not, that’s great too. You can always revisit this later if you change your mind.

HINT: Using d3.select(), you can select the element that has changed and retrieve the property and HTML id that has changed.

Check out the Plotly [gauge charts in JavaScript (Links to an external site.)](https://plotly.com/javascript/gauge-charts/) documentation and use these hints.

* Assign the variable created in Step 3 to the value property.
* The type property should be "indicator".
* The mode property should be "gauge+number".
* For the title object, assign the title as a string using HTML syntax to the text property.
* For maximum range for the gauge should be 10.
* Set the bar color of the gauge to black or a dark color to contrast against the range colors.
* Assign different colors as string values in increments of 2 for the steps object. The colors can be named colors as in the [Matplotlib colors (Links to an external site.)](https://matplotlib.org/3.1.0/gallery/color/named_colors.html) or rgba values.
* In Step 5, create the layout for the gauge chart making sure that it fits in the <div></div> tag for the gauge id.
* In Step 6, use the Plotly.newPlot() function to plot the trace object and the layout.

After you have completed the coding requirements, your dashboard will look like this image when it loads for the first time, with the bar chart you created in Deliverable 1, the bubble chart created in Deliverable 2, and the gauge chart:

**Deliverable 3 Requirements**

You will earn a perfect score for Deliverable 3 by completing all requirements below:

* The code to build the gauge chart does the following: **(10 pt)**
  + 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.
* 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 **(10 pt)**

**Deliverable 4: Customize the Dashboard (20 points)**

**Deliverable 4 Instructions**

Use your knowledge of HTML and Bootstrap to customize the webpage for your dashboard.

1. Customize your dashboard with three of the following:
   * Add an image to the jumbotron.
   * Add background color or a variety of compatible colors to the webpage.
   * Use a custom font with contrast for the colors.
   * Add more information about the project as a paragraph on the page.
   * Add information about what each graph visualizes, either under or next to each graph.
   * Make the webpage mobile-responsive.
   * Change the layout of the page.
   * Add a navigation bar that allows you to select the bar or bubble chart on the page.
2. When the dashboard is first opened in a browser, ID 940’s data should be displayed in the dashboard, and the three charts should be working according to their requirements.
3. When a sample is selected, the dashboard should display the data in the panel and all three charts according to their requirements.

**Deliverable 4 Requirements**

You will earn a perfect score for Deliverable 4 by completing all requirements below:

* The webpage has three customizations. **(10 pt)**
* When the dashboard is first opened in a browser, ID 940’s data should be displayed in the dashboard, and all three charts should be working according to the requirements when a sample is selected from the dropdown menu **(5 pt)**

**Submission**

Once you’re ready to submit, make sure to check your work against the rubric to ensure you are meeting the requirements for this Challenge one final time. It’s easy to overlook items when you’re in the zone!

As a reminder, the deliverables for this Challenge are as follows:

* Deliverable 1: Create a Horizontal Bar Chart
* Deliverable 2: Create a Bubble Chart
* Deliverable 3: Create a Gauge Chart
* Deliverable 4: Customize the Dashboard

Upload the following to your GitHub pages repository:

* The updated index.html file.
* The charts.js file, which should be in the js folder of the static folder.
* The samples.json file.
* A README.md that describes the purpose of the repository. Although there is no graded written analysis for this challenge, it is encouraged and good practice to add a brief description of your project.

To submit your challenge assignment for grading in Bootcamp Spot, click Start Assignment, click the Website URL tab, then provide the URL to your deployment and your GitHub repository, and then click Submit. Comments are disabled for graded submissions in BootCampSpot. If you have questions about your feedback, please notify your instructional staff or the Student Success Manager. If you would like to resubmit your work for an improved grade, you can use the **Re-Submit Assignment** button to upload new links. You may resubmit up to 3 times for a total of 4 submissions.