**# Belly Button Biodiversity**

![Bacteria by filterforge.com](Images/bacteria\_by\_filterforgedotcom.jpg)

In this assignment, you will build an interactive dashboard to explore the [Belly Button Biodiversity DataSet](http://robdunnlab.com/projects/belly-button-biodiversity/).

**## Step 1 - Flask API**

**Use Flask to design an API for your dataset and to serve the HTML and JavaScript required for your dashboard page.** Note: We recommend **using the sqlite database file and SQLAlchemy inside of your Flask application code, but you are permitted to read the CSV data directly into Pandas DataFrames** for this assignment. You will still need to **output the data as JSON in the format specified in the routes below**.

\* First, create a template called `index.html` for your dashboard landing page. Use the Bootstrap grid system to create the structure of the dashboard page.

\* Next, create the following routes for your api.

```python

@app.route("/")

"""Return the dashboard homepage."""

```

```python

@app.route('/names')

"""List of sample names.

Returns a list of sample names in the format

[

"BB\_940",

"BB\_941",

"BB\_943",

"BB\_944",

"BB\_945",

"BB\_946",

"BB\_947",

...

]

"""

```

```python

@app.route('/otu')

"""List of OTU descriptions.

Returns a list of OTU descriptions in the following format

[

"Archaea;Euryarchaeota;Halobacteria;Halobacteriales;Halobacteriaceae;Halococcus",

"Archaea;Euryarchaeota;Halobacteria;Halobacteriales;Halobacteriaceae;Halococcus",

"Bacteria",

"Bacteria",

"Bacteria",

...

]

"""

```

```python

@app.route('/metadata/<sample>')

"""MetaData for a given sample.

Args: Sample in the format: `BB\_940`

Returns a json dictionary of sample metadata in the format

{

AGE: 24,

BBTYPE: "I",

ETHNICITY: "Caucasian",

GENDER: "F",

LOCATION: "Beaufort/NC",

SAMPLEID: 940

}

"""

```

```python

@app.route('/wfreq/<sample>')

"""Weekly Washing Frequency as a number.

Args: Sample in the format: `BB\_940`

Returns an integer value for the weekly washing frequency `WFREQ`

"""

```

```python

@app.route('/samples/<sample>')

"""OTU IDs and Sample Values for a given sample.

Sort your Pandas DataFrame (OTU ID and Sample Value)

in Descending Order by Sample Value

Return a list of dictionaries containing sorted lists for `otu\_ids`

and `sample\_values`

[

{

otu\_ids: [

1166,

2858,

481,

...

],

sample\_values: [

163,

126,

113,

...

]

}

]

"""

```

---

**## Step 2 - Plotly.js**

Use Plotly.js to build interactive charts for your dashboard.

**\* Use the route `/names` to populate a dropdown select element with the list of sample names.**

\* Use `document.getElementById`, `document.createElement` and `append` to populate the create option elements and append them to the dropdown selector.

\* Use the following HTML tag for the dropdown selector

```html

<select id="selDataset" onchange="optionChanged(this.value)"></select>

```

\* Create a function called `optionChanged` to handle the change event when a new sample is selected (i.e. fetch data for the newly selected sample).

![dropdown](Images/dropdown.png)

**\* Create a PIE chart that uses data from your routes `/samples/<sample>` and `/otu` to display the top 10 samples.**

\* Use the Sample Value as the values for the PIE chart

\* Use the OTU ID as the labels for the pie chart

\* Use the OTU Description as the hovertext for the chart

\* Use `Plotly.restyle` to update the chart whenever a new sample is selected

![PIE Chart](Images/pie\_chart.png)

**\* Create a Bubble Chart that uses data from your routes `/samples/<sample>` and `/otu` to plot the \_\_Sample Value\_\_ vs the \_\_OTU ID\_\_ for the selected sample.**

\* Use the OTU IDs for the x values

\* Use the Sample Values for the y values

\* Use the Sample Values for the marker size

\* Use the OTU IDs for the marker colors

\* Use the OTU Description Data for the text values

\* Use `Plotly.restyle` to update the chart whenever a new sample is selected

![Bubble Chart](Images/bubble\_chart.png)

**\* Display the sample metadata from the route `/metadata/<sample>`**

\* Display each key/value pair from the metadata JSON object somewhere on the page

\* Update the metadata for each sample that is selected

**\* You are welcome to create any layout that you would like for your dashboard. An example dashboard page might look something like the following.**

![Example Dashboard Page](Images/dashboard\_part1.png)

![Example Dashboard Page](Images/dashboard\_part2.png)

**\* Finally, deploy your Flask app to Heroku.**

---

## Optional Challenge Assignment

**The following task is completely optional**

\* Adapt the Gauge Chart from [https://plot.ly/javascript/gauge-charts/](https://plot.ly/javascript/gauge-charts/) to plot the Weekly Washing Frequency obtained from the route `/wfreq/<sample>`

\* You will need to modify the example gauge code to account for values ranging from 0 - 9.

\* Use `Plotly.restyle` to update the chart whenever a new sample is selected

![Weekly Washing Frequency Gauge](Images/gauge.png)

---

**## Hints**

\* Use `Plotly.d3.json` to fetch data for all of your api routes

\* Refer to the [Plotly.js Documentation](https://plot.ly/javascript/) when building the plots

\* Use Bootstrap to structure your HTML template.

\* Use Pandas inside of your Flask routes to help format, filter, or sort the data before converting to JSON

**SUPPORTING KNOWLEDGE BASE - NOTES**

**document.getElementById - EXAMPLE BELOW which can be used in the code.**

<!DOCTYPE html>

<html>

<body>

<p id="demo">Click the button to change the text in this paragraph.</p>

<button onclick="myFunction()">Try it</button>

<script>

function myFunction() {

document.getElementById("demo").innerHTML = "Hello World";

}

</script>

</body>

</html>

Output - Hello World

ACTION – If text is changed from “Hello World" to “Hi” in <script> above, Hi will be printed.

**document.createElement` and `append`**

<!DOCTYPE html> // **Create a <button> element:**

<html>

<body>

<p>Click the button to make a BUTTON element.</p>

<button onclick="**myFunction()**">Try it</button>

<script>

function **myFunction()** {

var **btn** = document.createElement("BUTTON");

document.body.appendChild(**btn**);

}

</script>

</body>

</html>

Output – button with Try it label on top.

<!DOCTYPE html> // **Create a button with text**

<html>

<body>

<p>Click the button to make a BUTTON element with text.</p>

<button onclick="**myFunction()**">Try it</button>

<script>

function **myFunction()** {

var **btn** = document.createElement("BUTTON"); // Create a <button> element

var t = document.createTextNode("CLICK ME"); // Create a text node

**btn**.appendChild(t); // Append the text to <button>

document.body.appendChild(**btn**); // Append <button> to <body>

}

// **Tip:** After the element is created, use the [element.appendChild()](https://www.w3schools.com/jsref/met_node_appendchild.asp) or [element.insertBefore()](https://www.w3schools.com/jsref/met_node_insertbefore.asp) method to insert it to the document.

</script>

</body>

</html>

Output – When Try it button is clicked, CLICK ME button is created.

<!DOCTYPE html> **// Create a <p> element with some text, and append it to document:**

<html>

<body>

<p>Click the button to create a P element with some text.</p>

<button onclick="**myFunction()**">Try it</button>

<script>

function **myFunction()** {

var **para** = document.createElement("P"); // Create a <p>

var **t** = document.createTextNode("This is a paragraph."); // Create a text node

para.appendChild( **t** ); // Append the text to <p>

document.body.appendChild(**para**); // Append <p> to <body>

}

</script>

</body>

</html>

Output - Button Try it created, which if clicked prints -

“This is a paragraph.”

<!DOCTYPE html> // **Create a <p> element and append it to a <div> element:**

<html>

<head>

</head>

<body>

<p>Click the button to create a P element with some text, and append it to DIV.</p>

<div id="myDIV">A DIV element</div>

<button onclick="**myFunction()**">Try it</button>

<script>

function **myFunction()** {

var **para** = document.createElement("P"); // Create a <p>

var **t** = document.createTextNode("This is a paragraph."); // Create a text node

para.appendChild( **t** ); // Append the text to <p>

document.getElementById("myDIV").appendChild(**para**); // Append <p> to <div> with id="myDIV"

}

</script>

</body>

</html>

Output - **Click the button to create a P element with some text, and append it to DIV.**

**A DIV element**

**This is a paragraph.**

**This is a paragraph.**

Try it //This button pressed twice.

**ISSUES**

1. [http://localhost:5000/wfreq/BB\_942 throws error as BB\_942](http://localhost:5000/wfreq/BB_942%20throws%20error%20as%20BB_942) doesn’t exist. How to display ‘sample doesn’t exist.
2. print() doesn’t print to console.