# 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 - Plotly.js

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

\* Create a PIE chart that uses data from your samples route (`/samples/<sample>`) to display the top 10 samples.

\* Use `sample\_values` as the values for the PIE chart

\* Use `otu\_ids` as the labels for the pie chart

\* Use `otu\_labels` as the hovertext for the chart

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

\* Create a Bubble Chart that uses data from your samples route (`/samples/<sample>`) to display each sample.

\* Use `otu\_ids` for the x values

\* Use `sample\_values` for the y values

\* Use `sample\_values` for the marker size

\* Use `otu\_ids` for the marker colors

\* Use `otu\_labels` for the text values

![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 all of the plots any time that a new sample 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)

## Step 2 - Heroku

Deploy your Flask app to Heroku.

\* You can use the provided sqlite file for the database.

\* Ask your Instructor and TAs for help!

- - -

## Advanced Challenge Assignment (Optional)

The following task is completely optional and is very advanced.

\* Adapt the Gauge Chart from <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.

\* Update the chart whenever a new sample is selected

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

- - -

## Flask API

Use Flask API starter code to serve the data needed for your plots.

\* Test your routes by visiting each one in the browser.

- - -

## Hints

\* Don't forget to `pip install -r requirements.txt` before you start your server.

\* Use `console.log` inside of your JavaScript code to see what your data looks like at each step.

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

- - -

### Copyright

Data Boot Camp © 2018. All Rights Reserved. // @TODO: Complete the following function that builds the metadata panel

------------------------------------------------------------------

// Use `d3.json` to fetch the metadata for a sample

// Use d3 to select the panel with id of `#sample-metadata`

// Plot the default route once the page loads

function buildMetadata(sample) {

// @TODO: Complete the following function that builds the metadata panel

var defaultURL = "/metadata/" + sample;

console.log(defaultURL);

d3.json(defaultURL).then(function(data) {

var sel = d3.select("#sample-metadata");

sel.html("");

//var obj = Object.entries(data);

// obj.foreach();

Object.entries(data).forEach(function([key, value]) {

console.log(key, value);

var cell = sel.append("h5");

cell.text(key + ":" + value);

// console.log(obj);

});

});

}

// @TODO: Use `d3.json` to fetch the sample data for the plots

// HINT: You will need to use slice() to grab the top 10 sample\_values,

// otu\_ids, and labels (10 each).

function buildCharts(sample) {

var defaultURL2 = "/samples/" + sample;

console.log(defaultURL2);

d3.json(defaultURL2).then(function(data) {

console.log(data);

//data2 = data.slice(0, 9);

var data = [{

values: data.sample\_values.slice(0, 20)

labels: data.otu\_labels.slice(0, 20),

type: 'pie'

}];

var layout = {

height: 400,

width: 500

};

// Array of names

//const names = ["Jane", "John", "Jimbo", "Jedediah"];

// Slices first two names

//const left = names.slice(0, 2);

//console.log(left);

//Plotly.newPlot('pie', data, layout);

// @TODO: Use `d3.json` to fetch the sample data for the plots

// @TODO: Build a Bubble Chart using the sample data

// @TODO: Build a Pie Chart

// HINT: You will need to use slice() to grab the top 10 sample\_values,

// otu\_ids, and labels (10 each).

});

}

// Use d3 to select the panel with id of `#sample-metadata`

// Update the plot with new data

//function updatePlotly(newdata) {

// Plotly.restyle("bar", "x", [newdata.x]);

// Plotly.restyle("bar", "y", [newdata.y]);

//}

// Get new data whenever the dropdown selection changes

//function getData(route) {

// console.log(route);

// / d3.json(`/${route}`).then(function(data) {

// console.log("newdata", data);

// updatePlotly(data);

// });

//}

// Use `d3.json` to fetch the metadata for a sample

// Use d3 to select the panel with id of `#sample-metadata`

// Plot the default route once the page loads

//}

// Use `.html("") to clear any existing metadata

// Use `Object.entries` to add each key and value pair to the panel

// Hint: Inside the loop, you will need to use d3 to append new

// tags for each key-value in the metadata.

// BONUS: Build the Gauge Chart

// buildGauge(data.WFREQ);

//function buildCharts(sample) {

// @TODO: Use `d3.json` to fetch the sample data for the plots

// @TODO: Build a Bubble Chart using the sample data

// @TODO: Build a Pie Chart

// HINT: You will need to use slice() to grab the top 10 sample\_values,

// otu\_ids, and labels (10 each).

//}

function init() {

// Grab a reference to the dropdown select element

var selector = d3.select("#selDataset");

// Use the list of sample names to populate the select options

d3.json("/names").then((sampleNames) => {

sampleNames.forEach((sample) => {

selector

.append("option")

.text(sample)

.property("value", sample);

});

// Use the first sample from the list to build the initial plots

const firstSample = sampleNames[0];

buildCharts(firstSample);

buildMetadata(firstSample);

});

}

function optionChanged(newSample) {

// Fetch new data each time a new sample is selected

buildCharts(newSample);

buildMetadata(newSample);

}

// Initialize the dashboard

init();

# Heroku Deployment

If you are having trouble with Heroku deployment through the terminal, please follow these instructions.

## Procfile

Your procfile should have the following structure:

`web: gunicorn app:app`

## Requirements

Please follow the given directions to create your `requirements.txt` file

## Deployment

1. Push all changes to your repository

2. Open [Heroku](heroku.com)

3. Create a new application or open an existing application

4. Select the \*\*Deploy\*\* menu

5. Scroll down to \*\*Deployment Method\*\* and select \*\*GitHub\*\*

6. Allow access to your GitHub, search for your specific repository and connect to it

7. Scroll all the way down and select \*\*Deploy Branch\*\*

Once complete you will be able to launch your application

### Optional Automatic Deploys

If you want to make changes the future, but don't want to keep coming back to deploy your application, you can enable automatic deploys so that Heroku will deploy your app with the changes as soon as you push to your repository.