Real-Time Bike Station Data Visualization Using Python and Plotly

Customers are at the heart of every modern business and innovative companies use data visualization in various ways to make their service more personal and efficient.

Data visualization is a useful strategy for turning abstract and vast quantities of data into a simple visual that end users can understand and use to make decisions at a glance.

Shape

Description automatically generated

It’s essential for companies to create meaningful visualization dashboards, especially real-time dashboards.

Table of Content

1. Introduction
2. Business Problem and Solution
3. Logic
4. Data Source
5. Visualization
6. Conclusion
7. References

## Business Problem and Solution

Since this article is just for demonstration purposes, I tried to make the logic very simple while focusing on solving a single problem.

## Problem:

A biker wants to rent a bike station and wants information about the available bikes at the station.

## You can expand on the problem as desired.

#### Solution

1. We will ask the biker to provide their city name.
2. Return a map of the requested city showing the different stations.
3. This map will also show each station’s name, the number of available bikes, availability of bikes, and payment options amongst others.
4. If the biker wants, they can check a bar chart and pie chart visualizing more information by providing the name of one of the stations shown on the map.
5. The biker can refresh the dashboard for updated information.

#### Visualizations

1. Live Map (Dashboard)
2. Bar Chart
3. Pie Chart

Note that the visualizations used are basic in other to give you an idea of how to create a real-time dashboard.

## Visualization Library

Visualization libraries exist to help us visualize data and understand complex ideas by creating interactive graphs, charts, and maps. There are several Python visualization libraries, each of which has its benefits.

We will use Plotly Express in this article. **[Plotly Express](https://plotly.com/python/plotly-express/" \t "_blank)** is a free and open-source library that provides more than 30 functions for creating interactive, publication-quality graphs.

## Plotly can be used for basic visualizations as shown below:

Graphical user interface, application

Description automatically generated

Source: <https://plotly.com/python/>

… or complex visualizations like Financial Charts, Maps, Artificial Intelligence, and Machine Learning as shown below.

Graphical user interface, application

Description automatically generated

Source: <https://plotly.com/python/>

### Data Source

To show how we create this dashboard with a real-world API, we use Citybikes data from the Citybikes API as an example.

Citybikes is a project that provides bike-sharing data for apps, research, and projects to use. It currently supports more than 400 cities and the Citybikes API is the most widely used dataset for building bike-sharing transportation projects.

Text

Description automatically generated

CityBikes API Documentation

This API provides information across different cities, networks, and companies that provide bike services.

You can quickly view information about one of the services by visiting the network API endpoint for a network.

For example, to inspect the bike data for the network, Velib, visit <http://api.citybik.es/v2/networks/velib> (preferable on Firefox) where velib is the network id. The output is shown below.

Graphical user interface

Description automatically generated with medium confidence

Here, you can see that the API returned information that the biker might be interested in. You can pick any station data you are interested in but for this article, we will pick the following:

**Latitude, Longitude, Station Name, Empty Slots, Free Bikes, Time Stamp**

**(the data was generated).**

You can get access to the Citybikes API documentation [here](http://api.citybik.es/v2/).

## Logic

A biker requests a list of bikes in a station by providing the city name and then a specific station’s name.

*For advanced applications, this can be directly gotten from location information provided by the biker’s device GPS.*

Set up  
Make sure you have the necessary packages installed (if you use Google Collab, you might not need to install the libraries).

Requests  
Pandas  
Plotly Express

Import packages  
Here we import the packages used for this article.

A picture containing graphical user interface

Description automatically generated

### Helper functions to get and prepare the live data

#### Get the Citybike Data

First, we use the Citybike API to get the data across all cities. We can inspect this in Firefox.

Graphical user interface, application

Description automatically generated

The JSON response gives us information about the city which includes the country, latitude, and longitude among others.

*More importantly, it provides information about the****network API endpoint****which is necessary to access the list of stations in the requested city.*

We can use the requests library to get the output in Python. This is a JSON data and is stored in the city\_bike\_networks variable.



We create a function that returns a list of dictionaries with information about the requested city. Note that this should work regardless of capital letter usage.

Graphical user interface, text, application

Description automatically generated

Here, we get a dictionary of all the networks and filter out the requested city.

Showing that it works for different cities:

Graphical user interface, text, application, email

Description automatically generated

Let’s get the network data for one city: Paris

Graphical user interface, text, application, Word, email

Description automatically generated

We can pick the href (network API endpoint): *v2/networks/velib*, which is useful for our next helper function.

#### Get station data

This function takes a city name as a parameter and returns its list of stations’ info, only if the city name matches exactly one city bike company.

If there isn’t exactly one city bike company for a given string, we print an error message and return None (or more useful information to the biker).

Graphical user interface, text, application

Description automatically generated

The full network URL here is <http://api.citybik.es/v2/networks/velib>

If we use Paris as the input, we get this response:

A picture containing text

Description automatically generated

This gives us information about each station in Paris. This is the information the biker is interested in. We can pick some useful information for a station.

empty\_slots': 14,  
 'extra': {'banking': True,  
 'ebikes': 9,  
 'last\_updated': 1651231991,  
 'payment': ['creditcard'],  
 'renting': 1,  
 'returning': 1,  
 'station\_id': 653144717,  
 'uid': '44014'},  
 'free\_bikes': 11,  
 'id': 'cef647062b0306bc184c430223a2155e',  
 'latitude': 48.774703,  
 'longitude': 2.373685,  
 'name': 'Camille Risch - Paul Armangot',  
 'timestamp': '2022-04-29T12:16:42.381000Z'}

## We will be visualizing this.

#### Get Available Stations

This function returns a Pandas dataframe containing the information we want to visualize across all stations. It takes the city name as an input.

Graphical user interface, text

Description automatically generated

Showing this for Paris.

Table

Description automatically generated with medium confidence

## Helper functions for visualization with Plotly

#### Show Map

We use Plotly to display a map of stations in the city selected.

Graphical user interface, text, application, email

Description automatically generated

We imported plotly.express as px earlier, and use the [scatter\_mapbox](https://plotly.com/python/scattermapbox/" \t "_blank) function which accepts several parameters including the data frame, longitude, latitude, and hover\_name.

This map shows information about each station including: empty\_slots, free\_bikes, ebikes.

*Note: To plot on Mapbox maps with Plotly you may need a Mapbox account and a public [Mapbox Access Token](https://www.mapbox.com/studio" \t "_blank).*

## Putting these together



### We get:

Chart, scatter chart

Description automatically generated

##### Zooming in and hovering on any station:

Diagram

Description automatically generated

##### Visualizations for a particular station.

Table

Description automatically generated

We can pick any station name, and we will pick Gare RER les Ardoines for demonstration.

##### Pie Chart Visualization

Graphical user interface, text

Description automatically generated with medium confidence

Running this gives:

Chart

Description automatically generated

##### Bar Chart Visualization

Graphical user interface, text, application

Description automatically generated

##### Running this gives:

Chart, bar chart

Description automatically generated

For live data, we only need to call any of the following functions depending on the visualization we need to see.

##### Map



##### Pie Chart



##### Bar Chart



# Conclusion

In this article, we have shown how to get live data from a public API (Citybikes) and use Plotly to visualize it.

Now you know how to make a real-time streaming dashboard and a refreshable dashboard in python using plotly.

This can be expanded outside Google Collab.

The code for this article can be found [here](https://github.com/ajosegun/Data-Visualization-Projects/blob/main/Real_Time_Bike_Station_Data_Visualization_Using_Python_and_Plotly.ipynb).

Hope you find this article helpful! Follow me on [Medium](https://medium.com/@ajosegun_) or [here](https://bio.link/ajosegun). :)

# References

**[CityBikes: bike sharing networks around the world](https://citybik.es/" \t "_blank)**

[Some years ago we wanted to create an android app for our local bike sharing system and found out there was no open…](https://citybik.es/" \t "_blank)

[citybik.es](https://citybik.es/" \t "_blank)

**[Plotly](https://plotly.com/python/" \t "_blank)**

[Plotly's Python graphing library makes interactive, publication-quality graphs. Examples of how to make line plots…](https://plotly.com/python/" \t "_blank)

[plotly.com](https://plotly.com/python/" \t "_blank)