**Technical Report**: Satellite Exploration

Prepared by Kholiswa Tsotetsi, Azzy Caceres, Ravi Chinnamaidu and Michael Striffler

Rutgers University Data Science Bootcamp

December 10th, 2020

**Abstract**: In this report, data from real-time satellite tracking website, N2YO.com was extracted utilizing web scraping and API calls. From here, data was either stored in a PostgreSQL database and/or directly converted to a json file. Visualizations were created using Javascript D3, Plotly, Leaflet, and ThreeJS libraries and plugins. Ultimately, a dashboard was created to display this real-time data utilizing Flask and live server on the local server.

**Introduction**

Since the 1950s, the curiosity and fascination with space has led to the boom in its exploration. Though space exploration was initially limited to governmental programs, its expansion has allowed for non-governmental agencies to utilize space for their own benefit. In today’s age, the use of rovers, spacecraft, satellites and other instruments has allowed us, as a society, to gain a deeper understanding of this universe.

The recent launches by SpaceX, a private space exploration company founded by Elon Musk, has sparked our interest in what man-made spacecrafts and satellites are truly orbiting above us. As these instruments are used for governments, climate tracking, cell phones, television, radio, and more, our group wanted to focus on these instruments, specifically satellites, by exploring the following questions:

1. What satellites might be over our heads at this moment?
2. Where were these satellites launched from? Which country?
3. How many satellites have been launched throughout the years?
4. What is their altitude?

To answer these questions, we searched for data pertaining to the different satellites that are currently in orbit. In doing so, we came across NY20.com as a satellite data source that tracks all satellites in real time. We chose this data source because it consolidates global satellite information all in one source. As you will find later in this report, there are thousands of satellites in orbit that are divided in many different categories. With that said, each one serves a purpose.

**Data Acquisition**

In order to access the information for our analysis, an API was required. Initially, we attempted to call the API directly in javascript however, the url for this API key required the input of each satellite ID. Thus, Python Jupyter notebook was utilized to loop through a select set of satellite IDs to receive the requested information. We compiled a list of satellite IDs from the Earth Resources category into a csv file which was uploaded into our notebook file. From here, we defined an open list and looped through this list which inputted each satellite ID into the API url for each variable of the list until the end. The returned data was then appended to our open list until completion. From here, we converted this list to a json object which was later used in our javascript for the Earth Resources visualizations. A similar procedure was performed to retrieve data for our satellite chart, which will be described below. In this case, a different API url was used to acquire all satellites, regardless of category, that are within a 90 degree radius from New Jersey. This API call resulted in close to 3,000 satellites which were converted into a json object for later use.

In addition to calling our API from a Python jupyter notebook and converting the response into a json object, we also utilized web scraping to obtain the real-time satellite data by satellite country and category. Here, we also utilized Python jupyter notebook and called SQLAlchemy among other toolkits to merge our data to a PostgreSQL database. Using the splinter toolkit, we called the N2YO.com satellite homepage which includes the satellite category, and number of satellites, debris, and rocket bodies currently in orbit. This information was scraped into an empty list which was then converted to a pandas dataframe. From here, the database was pushed to our SQL database. This process was repeated to extract data for the satellite country and additional satellite information. For the satellite country, we scraped the N2YO webpage for satellites by country and limited our search to satellites from the United States and the former states of the USSR which included over 3,000 satellites. The extracted data included the name, ID, international code, launch date, and period (in minutes) for each satellite. Again, this data was appended to a list, converted to a pandas dataframe, then sent to the corresponding table in our PostgreSQL database. Lastly, for additional satellite information, we scraped the N2YO.com for a list of countries along with the corresponding number of satellites launched. Once all of our extracted data was transferred to the PostgreSQL database, we then created a python file to create a Flask app to run the data.

**Data Analysis & Visualizations**

Maps

Using our extracted data, we created four different styled maps to represent this data.

For data representation we chose to create four different style maps. On the main page you will find that the map changes based on a selection where the user can choose country owner, satellite category, or satellite ID/Name. This main map on our homepage shows satellite location in realtime after making calls to the API. Secondly, we decided to create a 90 degree radius map, which shows satellites that are within this 90° degree radius from our current standing location. Think, the 90 degrees of sky overhead. This map has its own .json file for data representation. The markers on this map have different radius sizes, which represents the altitude of that specific satellite, the larger the radius the higher the satellite is in orbit.

Thirdly, we decided to create a marker cluster map to help us understand and show where the satellite was launched from and which country owns that satellite. The tooltip allows a hover over which shows that information. There are surprisingly very few launch locations on the globe, and countries without a space center will have to partner with other countries if they want to launch their own satellite.

Bar chart

What we wanted to understand about our satellite data was how many satellites are truly still in orbit and when they were launched. To get a better idea of this we thought it would be best if we looped through our json file to retrieve just the launch dates. We enacted a string slicing method to identify just the launch year. Once we were able to identify all the dates we totaled the launches throughout the decades. It was interesting to find that there are still satellites in orbit that were launched years ago, and interestingly enough more satellites were launched in the 90s then any other decade. It is important to note that the bar chart uses the same json data as the 90 degree radius map, which represents about 2900 satellites,

**Limitations**

One of the biggest issues was the sheer amount of objects in orbit, roughly over

21, 000 space objects are being tracked by N2YO. So, when it came to API calling we encountered many errors, which made us change our strategy on how to build on our data set, starting small and working our way to a bigger scale. As an example, it takes a few seconds to call real-time locations of just 10 satellites.

We set out with an ambitious goal of mapping live locations on a 3-D global sphere, however we quickly learned that a 3D globe was not an attribute of mapbox, therefore our real-time map and others are on 2D maps.

**Errors**

One hurdle we had to jump through was reading the API. As mentioned, it was not an option to call through the API directly with a loop due to the CORS error. After countless tries, we decided to create Python code to read through the CSV and loop through the webpage and store the data into the empty list.

Like we mentioned in our limitations we really wanted a 3D sphere for mapping. As our project progressed we also needed to find another js library There were multiple libraries that we tried but were unsuccessful in creating the image. Coming across the Three.js library was the solution to this situation and it allowed us to create a 3D image that spins and shows the plotted data.

**Conclusion**

Ultimately we were able to create an HTML webpage that held all our findings. We worked through the errors and limitations and successfully produced visual images of the results.