**Data & Visualizations Team: Apurva Deore, Vishal Padma**  
  
This is the Sprint by Sprint breakdown of Tasks done by Data team.  
For a detailed document on all aspects of Data, please see the Document in the following path: **GITHUB\TeamRocket\Project\_Management\Documentation\Data Management\COVID-19 Data Management Document.docx**

**Sprint 1:**

*-****Consider data that is available within the scope of the project  
-Researching potential API's to use***

***-Collecting raw data from the potential shortlist of all available sources***

Considering that we need the data that’d answer **statistical** and **theoretical** questions asked to the **Chatbot**; we further divided the data into two broad sections.

I) Statistics of COVID-19 cases (total, currently active, recovered, deceased, tests done) [real time data required, updates happen regularly] -> **LIVE DATA**

II) Data already loaded in (symptoms of COVID-19, healthy habits to be followed, whom to contact for testing related situations...) [stored-and-forgotten scenario, doesn’t need updates] -> **OLD DATA**

* **DYNAMIC / LIVE DATA**

**Statistics of COVID-19 cases Per Country [Real Time Data, Updating regularly]**

-Total cases

-Active cases

-Recovered cases

-Deceased cases

-Tests done

* **STATIC DATA**

**Data already loaded in and doesn’t need frequent updates.**

-Symptoms of COVID-19?

-Healthy habits to be followed?

-Whom to contact for testing related situations (some helpline numbers)

Initial research was done on various APIs such as below,

* Johns Hopkins COVID-19 : Majorly using ( <https://api.covid19api.com/summary> )
* European Centre for Disease Prevention and Control: COVID-19

***API 1: Johns Hopkinns***

* It has all the data that is required for us to work on, ie. A lot of information about the statistics of MANY countries to start with.
* It gives the information specific to countries starting from the month of January until the latest updated date, ie. If we’re getting the data for the date 16th June, we’d get the data updated until 15th June. So basically, it provides us with real time data! This could be useful when we’re plotting visualizations for various countries (to show trends that the cases follow)
* We’ve already collected raw data pertaining to the total cases, recovered cases and deaths for about 186 countries as of now.
* We’ve also collected the time series data pertaining to the same 186 countries (for each day until today, starting from 22nd Jan, until the most recent update date of data collection).
* There were some data discrepancies, where the data was not recorded or recorded as wrong. To handle this we have replaced the statistics of the previous date wherever required and in some cases the active cases for each country were recorded falsely. So we calculated the active cases on a given date(where it was recorded falsely) by subtracting the death cases and recovered cases from confirmed cases.

**Sprint 2+3**:  
  
***-Finalize potential API's to use  
-Data cleaning and preprocessing from the chosen APIs  
-Aim to fine-tune the data samples , Preparation of global summary data of all countries until the date of visit to the website  
-Provide snapshot of the final raw data and processed data to the cloud team for collaboration  
-Start deciding on the types of visualizations that could be used for the website  
(Once the data has been finely prepared and curated)***

***-Identify potential tools to be used for Visualizations***

**Data Cleaning/Pre-processing**

1. **Johns Hopkinns**

Pre\_processing:

1. We need the summary data to be clean enough in the matter of Timestamp and other elements.
2. We have dropped only one column from the summary data which is “slug”. This column was used to refer to country names.
3. Next is the country data which we collected. This data has many columns which were null so they were dropped.
4. The next thing was to convert the data into time series data to provide detailed statistics starting from January 2020 until the latest day of update.

=> Now we have two main datasets pertaining to the Johns Hopkinns API

-> One with global summarized statistical data for 186 countries  
-> One with the time series data with information starting from the last week of Jan until the latest date of update.

For the cleaning/preprocessing we planned and did the following,  
-Summarized (1 dataframe having the statistical data) (To clean this dataset, we tried to remove unwanted columns like slug, it wasn’t really required, re-arranged the columns to make a better looking dataframe, changed/modified the date column such that it represent only the DATE and not the timestamp )  
-Time Series (in this, there are 4 different dfs corresponding to Active cases, deaths, confirmed cases and recovered cases)

-> Created a separate column to store the Country and another column that'd store the (Active, Confirmed, Recovered and deaths for every country)

So for every country A, there would be 4 rows dedicated to the Country A, e.g. Afghanistan  
eg, Country Afghanistan : Confirmed  
 Country Afghanistan : Active  
 Country Afghanistan : Deaths  
 Country Afghanistan : Recovered

So when we'd be referring to the cases in particular, we'd use two keys (Country name with the cases column) to identify the same.  
We also added additional columns to separate the date into the respective DAY of the week and MONTH of the year. (Didn't consider the month, it's constant: 2020). Used the Calendar package for the same. Additional columns added separately for each country's records.

**VISUALIZATIONS**

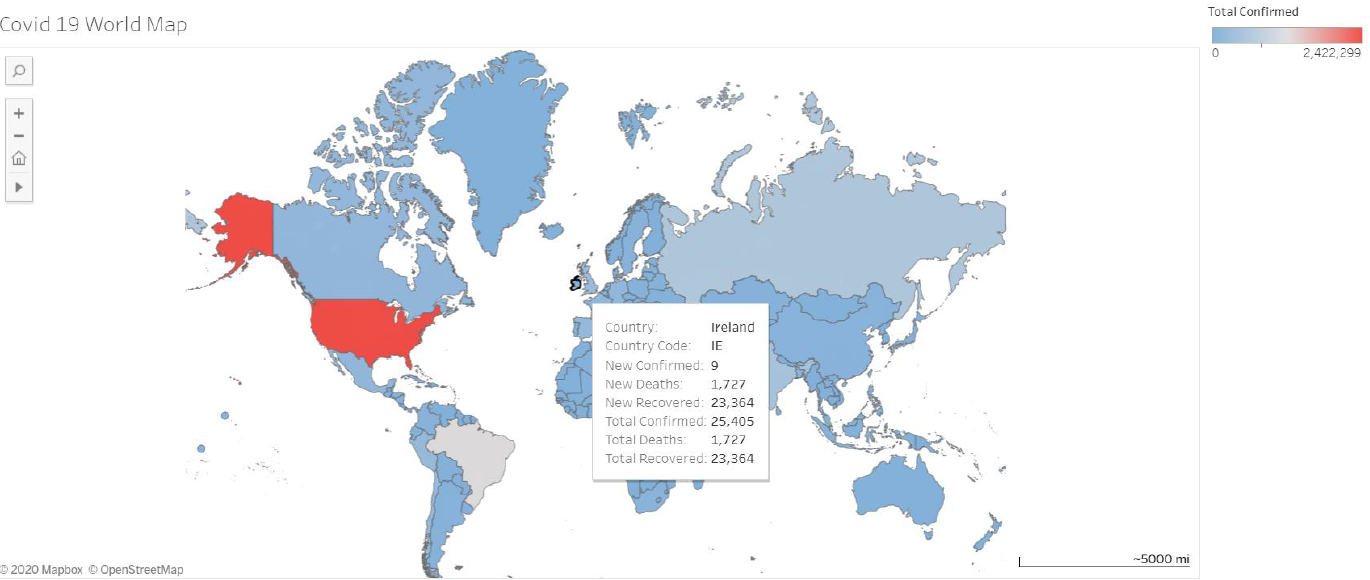
For now we’re using Tableau. We want to focus on providing useful insights revolving around the following (not limited to):

* Which countries have the most number of cases?
* Which countries have the best recovery rate?
* Which countries recorded a lot of deaths?
* What is the number of tests performed by each country?
* How many critical cases are present in X country?

1. ***TABLEAU***

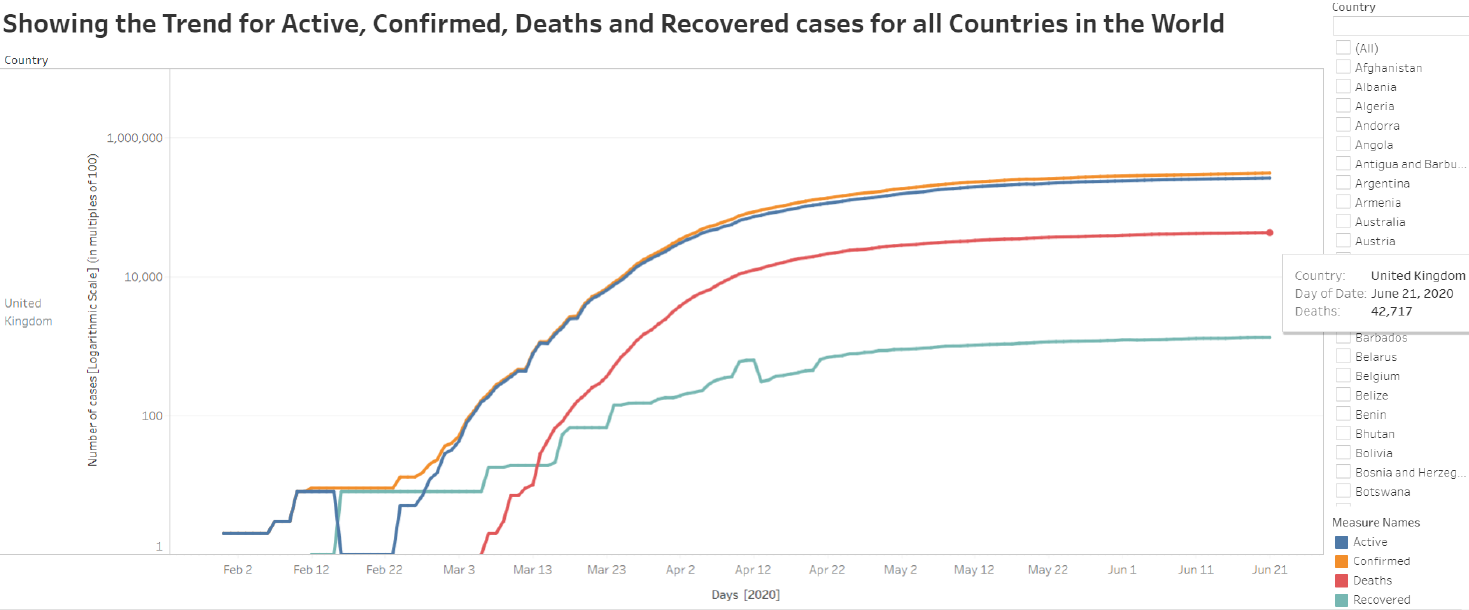
We’ve decided to make the following main types of visualizations

1) **Map related visualizations** (when a user hovers over a particular country, the map would show the user statistics for that country (ie. Details about active, deceased, recovered and confirmed cases). This would be displayed on the main webpage and uses the Summarized data which has info about cases for all countries affected by COVID-19.



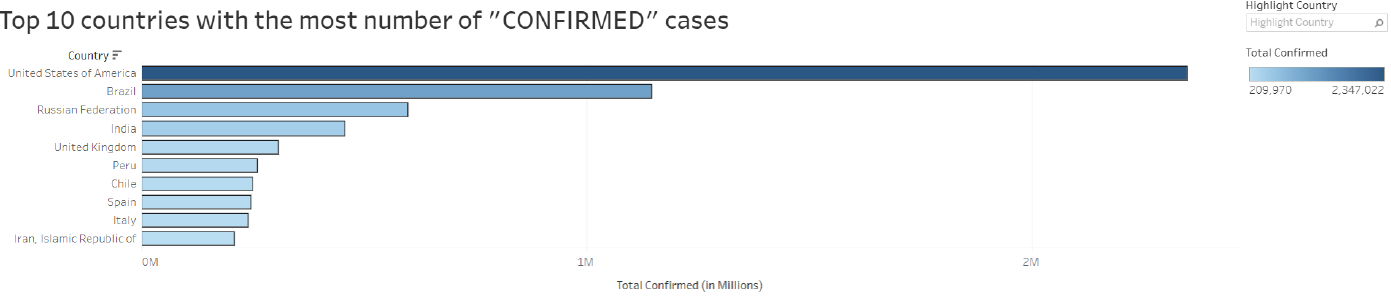
2) **Trend graph** : A visualization that’d show users the trend followed by a particular country. Basically 4 trend lines in one graph. (representing active, deceased, recovered and confirmed cases trend-lines).

* One worldwide (trends for the world’s total cases)
* One country-wise (filter on the country)



3) To show top 5/top 10 countries with respect to the cases as for recovered, confirmed and deceased cases. (considering the world-wide summarized cases dataset)

* Countries with rank in the Top 5 for Deaths, Recovered, Confirmed cases
* Countries with rank in the Top 10 for Deaths, Recovered, Confirmed cases



End of the sprint some more research for potential API’s was done and below are the details:

* Yatko API (<https://api.quarantine.country/api/v1/summary/latest>)
* Bing covid 19 data from github (<https://raw.githubusercontent.com/microsoft/Bing-COVID-19-Data/master/data/Bing-COVID19-Data.csv>)
* Ireland specific data (<https://data.gov.ie/>)

**Sprint 4 (Just before interim)**

***-Research on the second potential API: Yatko API, required and write scripts for Data Collection.***

***-Merging of the two APIs to create a single dataset which will contain all the data from both the API’s.***

***-Identify any data discrepancies in the john hopkins data, and fill the missing or bad data into john hopkins from the BING API.***

***-Finally to be a bit more ambitious for the upcoming interim demo, the team is exploring related research data carried out on the COVID-19 pandemic to integrate.***

**DATA COLLECTION**

***API 2: YATKO***

After analyzing the data from Yatko API, we have discovered the below mentioned details about the data:

* Firstly the data is a summary data which we found in Yatko API, where there are a total of 224 countries.
* There are several columns in this data and most of them are similar to the john hopkins data.
* The additional columns like, Death ratio, recovery ratio and tested cases in a particular country are also present in this data.

As of now we have two API which give summary for different countries but both have some different columns, so we decided to integrate both the datasets of summary and create a single dataset which will contain all the data from both the API’s.

So we have integrated the summary data for 186 countries from both the API and the final merged Dataset set has below mentioned columns for each country which will be later used in the visualization.

* CountryCode
* Country
* Date
* NewConfirmed
* NewDeaths
* NewRecovered
* TotalConfirmed
* TotalDeaths
* TotalRecovered
* Death\_Ratio
* Recovery\_Ratio
* Total\_Cases\_Tested

***API 3: BING***

The 3rd data which we will be looking into is the data from Bing-19 covid data and here the data is collected from many different sources and the same is converted to time series data. This data records data like Confirmed cases, deaths cases and recovered cases. Data is recorded when the first case is seen in that particular country. It also records the data for each date and for some countries the data is recorded for each date for each state/province. But some countries don't have data recorded according to the state/province. One interesting thing is that this API also records data for the whole world for each date.

As of now we have extracted the data from the API and saved into datasets and the next we are trying to convert the data in the format of what we have for the John Hopkins data. The idea of this is to, make sure when we have any data discrepancies in the john hopkins data, we can refer to this data and fill the missing or bad data into john hopkins from this data to maintain and fill the gap in the datasets which will be used by the chatbot or the world map.

**Data Cleaning/Pre-processing**

1. **Johns Hopkinns**

Pre\_processing: Later at this stage we have found out that the JOHN Hopkins data has some gaps in it where the active cases are recorded as zero and the recovered cases are recorded with wrong value in them. So to handle this we have pinpointed at the exact location of where the value is being recorded falsely. After getting the gaps, we have replaced the data with the same data from another API (Bing). Since bing and John Hopkins data was of the same format, we could integrate the data from one API to another to fill all the gaps in the main API.

**B) YATKO API**

i) The Yatko API provides with the columns: **name, iso3166a2, iso3166a3, iso3166numeric, total\_cases, active\_cases, deaths, recovered, critical, Tested, Death\_Ratio, Recovery\_Ratio, change**

After considering the Yatko API has additional data as compared to the Johns Hopkinns data, i.e. **critical, Tested, Death\_Ratio, Recovery\_Ratio, change** in particular, so we decided to combine those two datasets and keep the necessary columns integrated in one dataframe.

**ii) Merged Dataframe: Yatko + Johns Hopkinns API**

This is the merged dataframe in which we performed pre-processing and cleaning in order to make the Yatko dataframe compliant with the main dataframe (John Hopkinns).

In the pre-processing phase, we took care of the countries like Namibia (countryCode “NA” was getting counted as a null value when reading the dataset), differences between namings between the two APIs (USA/United States of America ; UK = United Kingdom), other countries like Congo had two different names assigned for same values, CAR = Central African Republic. To make the dataframe look more appealing and easier to interpret, we renamed columns and cleaned it further.

Now the merged dataframe consists of the following columns giving information for 186 countries:  
**CountryCode, Country, Date, NewConfirmed, NewDeaths, NewRecovered, TotalConfirmed, TotalDeaths, TotalRecovered, DeathRatio, RecoveryRatio, Total Tests Done.**For a clear and concise view of the data sources, we’ve put up a simple flow diagram to explain the main sources and what all data records/columns from them we would be using going further.

**C) BING API**

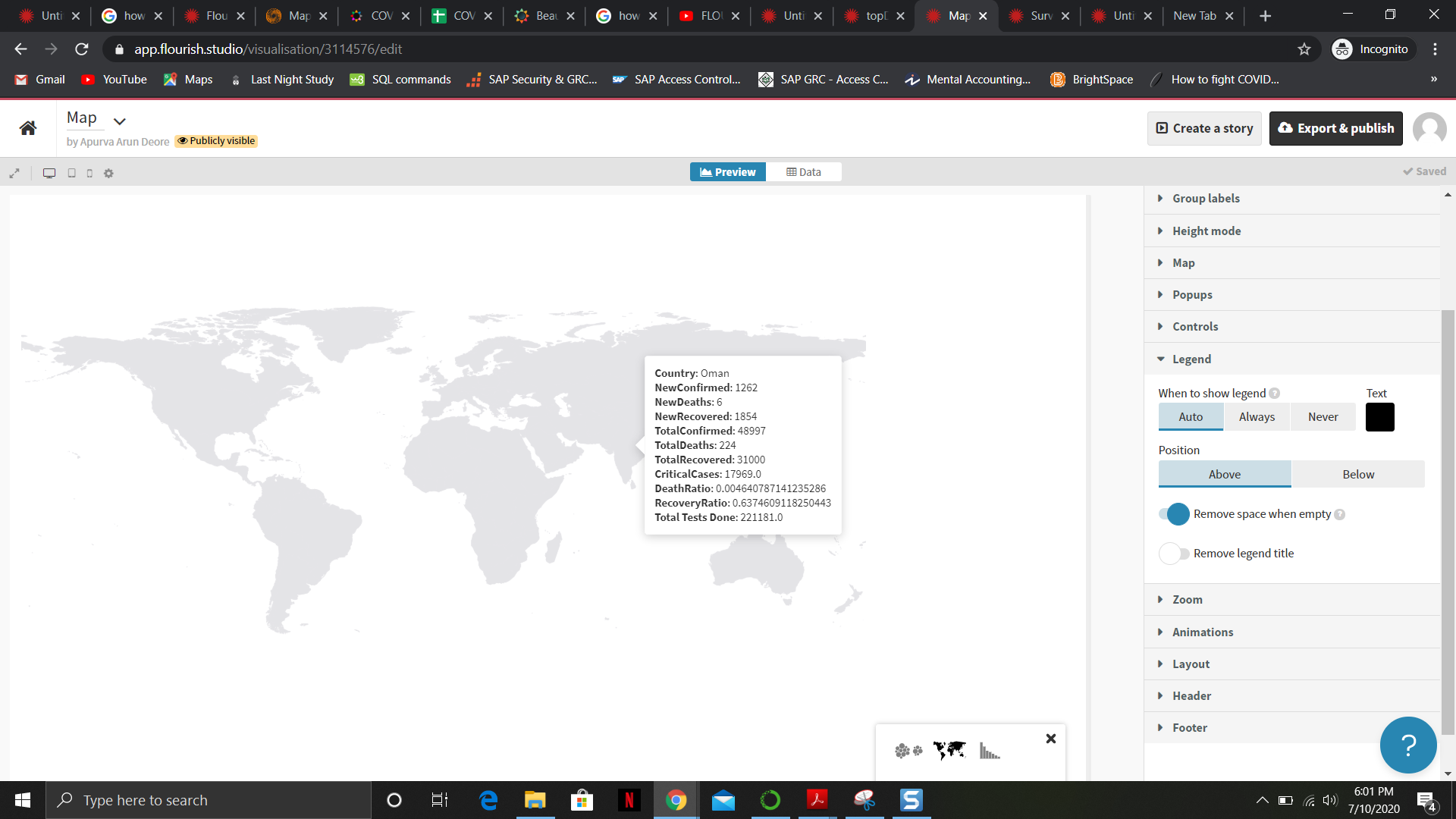
The Bing csv provides us with the columns similar to those provided by Johns Hopkinns API. We’re considering the BING dataset so as to help us to handle emerging/probable data discrepancies (if any).

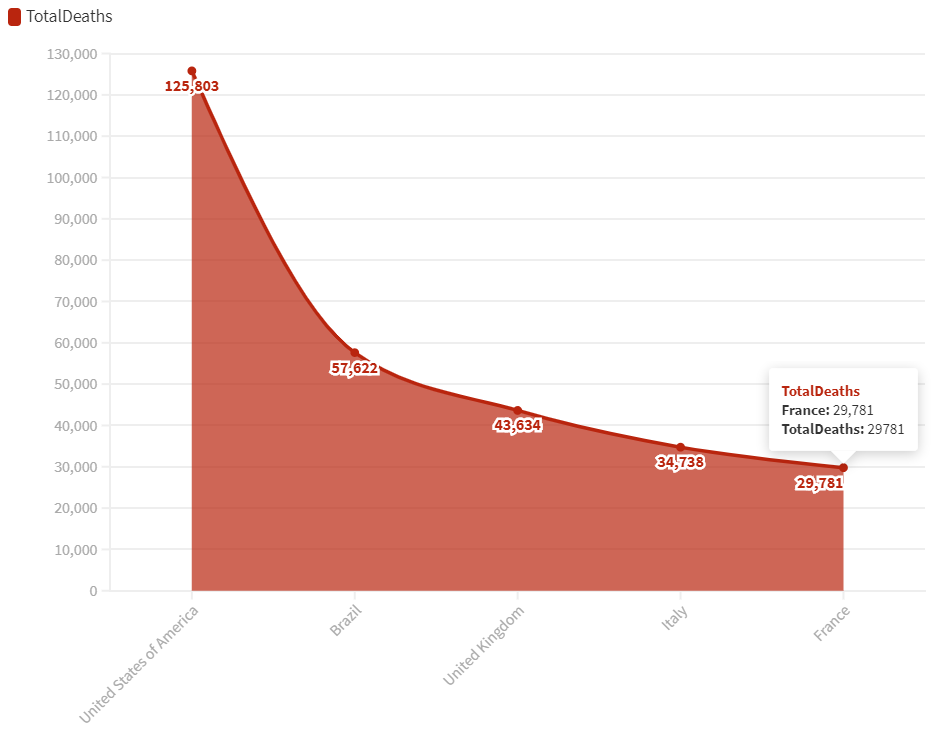
**VISUALIZATIONS**

We’re considering **other visualization tools such as Flourish and PowerBI** for creating visualization dashboards in addition to Tableau which would be embedded into our web-pages.

1. ***Flourish***

Flourish is a tool that allows to create visualization dashboards/storytelling scenarios in a more fancy way. It provides certain map visualizations, to name some: survey maps, projection maps. We tried to create a map visualization similar to that done in Tableau to sort of compare which one is better. Some visualization we tried were as follows:

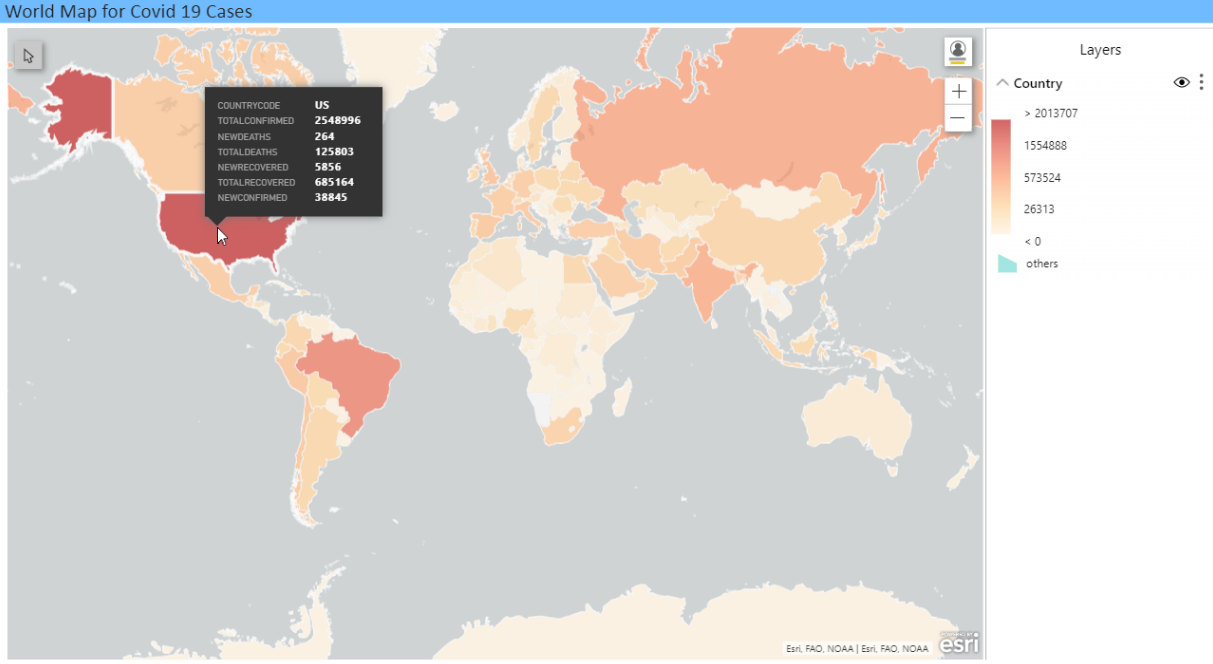


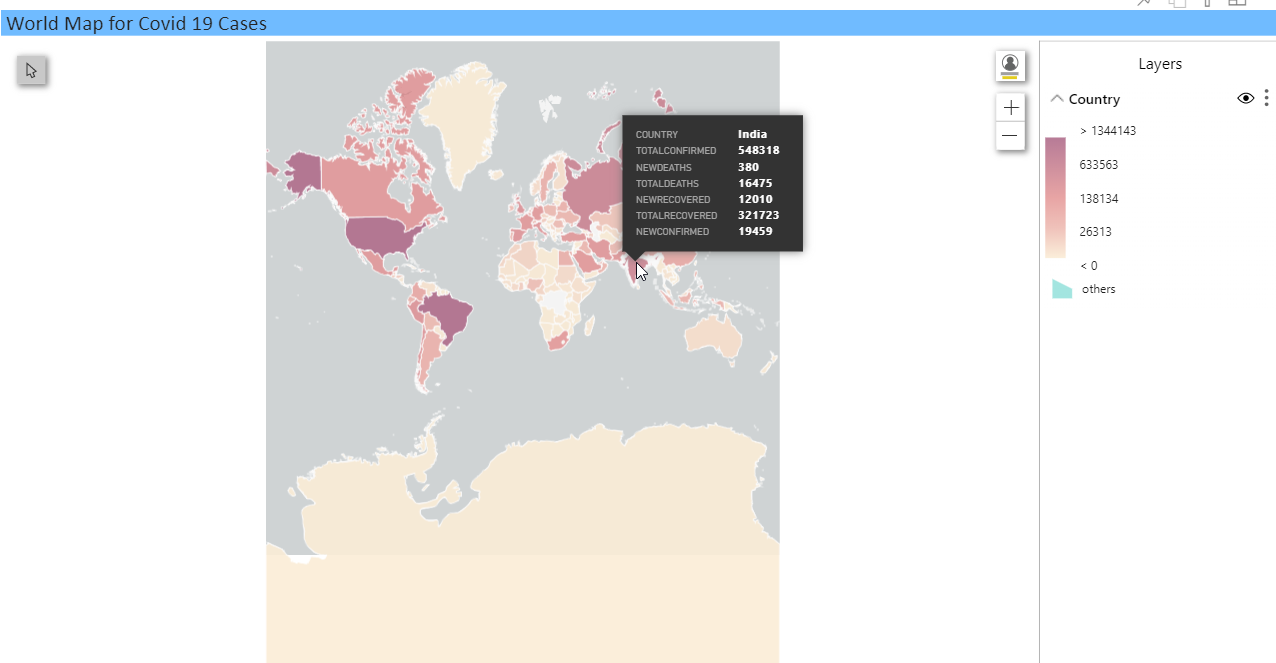


It provides a plain and bland view of the map with the statistical data displayed accurately, but it takes in a lot of unnecessary parameters for the visualization to be created and follows a very tedious approach. When dealing with a map, it asks for values like the geometry of a country, latitude/longitude values which makes it complicated unncessarily. Flourish could be used as a tool by those who write blogs, maintain fancy websites or someone who likes to create fancy stories from data.

1. ***PowerBI***

Tried the map visualizations with PowerBI and below are some examples of the same. PowerBI is an equally powerful tool as Tableau when it comes to delivering visualizations in terms of bar graphs, maps, pie charts, trends. It helps to create visualizations similar to that of Tableau.

******



Conclusion: After having done a trial and error application of each of the above three visualization tools, we’ve decided to go ahead with Tableau as our main visualization tool because of the ease and simplicity it provides when trying to create a visualization. It takes in just the right amount of columns required to create a visualization without having the need to use data columns which aren’t required at all. Also, after having embedded the maps created using Tableau and PowerBI into our web-pages, it was evident that the one’s created using Tableau provide a clear and concise view of the maps and also look better than the ones made using PowerBI. So for now, we’re sticking with Tableau as the main tool for visualization whereas we’re keeping PowerBI and Flourish as backup options, just in case we come up with some additional visualizations which can be made better-looking with PowerBI and Flourish.

***Sprint 5+6:***

***-Plans for coming week: With the scope of information being provided by the chatbot to be restricted to Ireland.***

***-Upon the overall world data by country which we had gathered we decided to prepare data of Ireland regarding the pandemic in greater detail to demonstrate the current scope of the project.***

***-The refined Irish data is aimed to demonstrate what type of data of each country we envision the final version of the application would contain for users to access. -Working on Tableau with further visualizations for the newly curated data pertaining to Ireland.***

***-These are the final sources of data, which will be used to train the bot.***

* ***Johns Hopkins API***
* ***Integration of the summary data from the two main APIs, Yatko and John Hopkins respectively and the final merged Dataset set for each country which will be later used in the visualization. (YATKO API + Johns Hopkinns )***
* ***BING***
* ***Ireland Data from data.gov.ie(HSPC)***

***Data Source 4: Ireland focussed data - data.gov.ie***

This data source provides us with the data focussed on Ireland only. It provides the statistical data for Ireland’s 26 counties separately as well as for the country as a whole.

**Dataframe1** : It consists of the country’s population data (pertaining to each county) , confirmed cases of COVID-19 till date and Cases by population data.  
**Dataframe2:** This data frame consists of statistical data revolving around the **main types of cases.  
Dataframe3:** This data frame consists of information regarding **Hospitalized Cases**.  
**Dataframe4**: This data frame consists of **Gender-wise** COVID-19 cases differentiation.  
**Dataframe5**: This data frame consists of the **reason of** **spread categories**.  
**Dataframe6**: This data frame consists of the **County-wise Confirmed Case related statistics**.

Apart from this real time data, we look forward to collecting static data (which requires no updation) pertaining to the symptoms of COVID, healthy habits and so on.

We’d also be getting information from some additional APIs and integrating the required datasets to enhance the availability of data from various APIs to be put in one place.

***Evaluation+last sprint:***

***-All scripts finalized for the Data Collection and Data Pre-processing Activities.***

***-Scripts triggered automatically via cron jobs using Windows Task Scheduler to be run everyday at a fixed time.***

***-All Visualization Dashboards Completed using Tableau and embedded onto the website successfully.***

***-Test Evaluation Scripts***

***Creation of a User Test Case script template and a peer review template using google forms.***

***-Identification of the scope of the bot and the collection of Amazon lex metrics to evaluate the performance of the chatbot.***

Test Scripts:

Once we had planned on the intents on the Lex chatbot, we were able to create test scripts for our chatbot. So we had planned out 30 different user test scenarios on how we can ask our bot questions and how it will respond. These user test cases where to check if the functionality of the bot is correct or not. We even made a Functional document on how the bot behaves and all the different functions of an intent. The test script created has each scenario sketched out with multiple steps, where we test the bot throughout without missing any functionality. Once the test script was ready, each member was assigned some test cases to execute and record the output, if the output deferred from the expected results they were noted in the remarks column in the test script.

So once the script was ready and once done executing them, the changes were supposed to be incorporated into the peer review document. This way we documented the changes to be made in the original test script and keep a track of the mistake in the test script. A peer review document is used to track the changes made in the test scripts.

**Major challenges along the way, the challenges tackled and how each challenge affected the vision of the project**

* The major challenge faced by us was to handle the data which was coming from the API’s.
* Data had gaps which were not encountered at the same time, as the data was updated we had to check for new data gaps and handle them accordingly.
* The other challenge was again to merge the data from two different data sources, where we had to get the format same for both the datasets (Unnecessary columns were dropped and date formats were made similar in all the data frames and then merged them accordingly.
* Next set of issues were seen in the visualizations, where we initially started with Tableau and made as many as visualization possible. Later when the scope was refined we had to redefine the visualizations according to the scope.
* We also struggled on choosing the proper software for visualization and we did explore other options as mentioned above. The thing with tableau is that the tableau online account is not free and we had to request for extension from the Tableau Academic team.
* The reason to do so was because we could only connect our database on Azure cloud to Tableau online and later create visualization using the tables.
* Once the dashboard was made all we had to do was to embed the dashboard using the tableau online script and put the same in our website CSS file.

-We also had some issues regarding countries with commas in them, i.e. Iran, Republic of , Ireland, Republic of. So in the Cloud database , data of one country was getting stored in the other countries data, we handled that by saving the csv file with a delimiter. By doing so the country with multiple words was saved in one column and the confusion was avoided later. If the delimiter was not added the country with two words in its names was saved in two different columns and the data for one country was represented by another.  
-When combining two different data frames several countries were represented in a different way, for example one dataframe had countries addressed as UK, USA; we then handled them by maintaining a single way of representing : United Kingdom, United States of America; other countries like Congo had two different names assigned for same values, CAR = Central African Republic. Similarly, we took care of the countries like Namibia (countryCode “NA” was getting counted as a null value when reading the dataset).

**Structure of data architecture, major sources, and how they were gathered**

