Data Science - Group Delivery Guide
January 2021 - The Bridge

INSTRUCTOR: Gabriel Vázquez Torres
gabriel@thebridgeschool.es

TEACHER: Clara Piniella Martinez
clara.piniella@thebridgeschool.es

TEACHER: Diomedes Barbero Martínez
diomedes@thebridgeschool.es

Delivery explanation

This group delivery aims to practice different concepts about EDA and APIs.

Groups

There are six groups:

- **Group A**:
- **Group B:**
- **Group C:**
- **Group D**:
- **Group E**:
- **Group F**:

Countries assigned

- **Group A**: Argentina, Russia, Colombia, Chile and Spain
- **Group B:** India, Peru, EEUU, Francia and Spain
- **Group C:** Mexico, Netherlands, Brazil, Iran and Spain
- **Group D**: Portugal, Venezuela, Turkey, UK and Spain
- **Group E**: Poland, South Africa, Ukraine, Indonesia and Spain
- **Group F**: Czechia, Canada, Romania, Belgium and Spain

Requirements

The next requirements are mandatories:

1. Each group must choose a person that will do the presentation. This time, the presentation must show the technical part of the project.
2. All the participants must write code tasks. For this, in each function, apart from that the function must be well documented, it must contain the name of the student or students that have done the exercise. It must appear with the syntax "@alias_of_github".
3. It is mandatory that each group uses the software *trello* (or other related) to manage the tasks in different status: TODO, DOING, REVIEW and DONE. Every student must review the tasks of the coworkers.
4. The delivery must be sent before 25/01/2021 at 23:59.
5. The delivery must be sent in a *.zip* file by email/classroom with this structure:
    a. A folder **src/** that contains all the source code.
    b. A folder **documentation/** that contains all the documents related to documentation (pdf, presentation, ...).
    c. A folder **resources/** that contains other useful content (images,...)
    d. A folder **reports/** that contains all related to created reports such as figures, html, pdf, etc
    e. A folder **notebooks/** that contains notebooks for your tests.
    f. A folder **data/** that contains the data of the project (optional).
    g. A folder **src/utils/** that contains all the modules used by the *main* file.
    h. A file **src/main.ipynb** that contains all the functionality. This file only can contains imports to your **src/utils/*** modules.

i. There are, at least, these modules inside **src/utils/** :

   i. “folders_tb.py” that contains the generic functionality related to open, create, read and write files.

   ii. “visualization_tb.py” that contains the generic functionality related to pandas, matplotlib, seaborn and other libraries focus on visualizations.

   iii. “mining_data_tb.py” that contains the generic functionality related to collect data, clean data and others (wrangling methods such us working with multiples jsons)

   iv. “apis_tb.py” that contains the generic functionality related to working with APIs.

   v. Others that the students need.

j. A file **src/services/api/server.py** that contains the functionality to start the Flask API. There are two GET functions:

   i. One that must allow to receive an ***group_id*** and will return a json with one key called “token” with the S value (explained below). This function must only return the S value in a json if the ***group_id*** received is equal to N (explained below). Otherwise, it must return a string with a message of error.

   ii. Another that must allow you to receive a ***token_id*** value and, if ***token_id*** is equal to S, return the json that contains *the json of your group*. Otherwise, return a string with a message of error.

      1. N is the letter of your group concatenated with the sum of the ages of the participants. Examples: “A103”, “C86”.

      2. S is the letter of your group concatenated with the sum of the DNIs of the participants of the group starting with the letter of your group. Example: “B85918591859851”, “D635154795182”.

k. A file **src/services/argparse/console.py**. This file will be executed by command line adding *"-j 18"* and will show *the json of your group*. If you pass another number or argument, it will show an error. The logic of this file must be done using classes (OOP).

6. The json returned (df) is different by group:

   a. **Group A**: it must return a json with one key *"n_c_averages"* that represents the mean of the "new_cases" per day of all of your countries.

   b. **Group B**: it must return a json with one key *"n_d_averages"* that represents the mean of the "new_deaths" per day of all your countries.

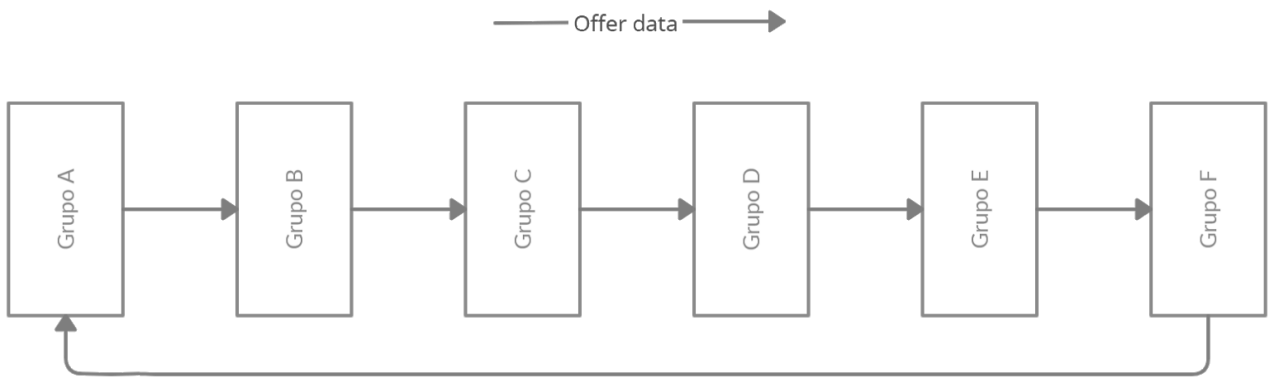
   c. **Group C**: it must return a json with one key *"t_c_averages"* that represents the mean of the "total_cases" per day of all your countries.

   d. **Group D**: it must return a json with one key *"t_d_averages"* that represents the mean of the "total_deaths" per day of all your countries.

   e. **Group F**: it must return a json with one key *"n_v_averages"* that represents the mean of the "new_vaccinations" per day of all your countries.

   f. **Group G**: it must return a json with one key *"n_t_averages"* that represents the mean of the "new_tests " per day of all your countries.

7. The groups are related as in the Figure 1:



Presentation

All groups must do a presentation about its project. The presenter of the group will use a presentation file to explain all the steps of the workflow with graphs.

The duration of the presentation won't be longer than 7 minutes so it is really important and necessary to explain the essential points of the work.

Evaluation criteria

For this delivery, there are different delivery options. Each group must choose what delivery they want to do. **C** is the minimum requirement for this delivery. There is a hierarchy in the options: **C→ B→ A→ A+***

It is not allowed to do:

- B without C
- A without B and C
- A+ without A, B and C

Option C

Apart from all requirements that are written in the **Requirements** section, there are the next mandatory exercises:

1. Document all steps. Structure your code to keep it cleaned using good practices.
2. Collect Coronavirus Data. It is mandatory that in each call, it collects the last updated data.
3. Determine and explain if the data is cleaned. If not, then clean it.
4. Create an API that returns a Json with the logic explained for your group. The flask server must be executed running the **src/api/server.py** file.
5. Get the jsons generated from your annexed group and plot it. First, try to connect to the private ip of your annexed group. If it is not possible because of physical issues, then simply use what they generate copying it. If your annexed group cannot give you the necessary json, then annotate it, use the json of another group.

6. Show different tendencies for each column in your dataset. Show, vertically, the start date and end date of the alarm state in each plot. If there is no alarm state, then show only the start date.
7. Draw the workflow of your program. You can use free tools.
8. Per country, which are the columns that are more related? find the correlation between columns with the *correlation matrix*.
9. Use a different github repository adding all group participants with write permissions. Use that repository to manage the delivery code and resources. It is mandatory that every student of the group does, at least, five commits/push.
10. Answer the questions:
    a. What position do your countries occupe respect to the number of total infected, total deaths and total recoveries?
    b. What can you conclude about your data study?
    c. Are there outliers or some rare data?

Option B

1. Draw, in different colors and vertically, the moments when the daily death curve increases and decreases.
2. Create with bars, lines, points and pie charts the daily deaths and infected.
3. Research to save each plot in local files. Save them on different folders for each country.
4. When are the worst moments to go to the countries? Answer this referring to the “per_million” columns.

Option A

1. Create a file that saves all plots in local files when you execute it by command line.
2. The main file must only import a class object from a module and execute all the logic.
3. Draw the correlation matrix using the top ten columns with the highest correlation.
4. Answer the questions:
    a. Can we conclude that the alarm state has had an effect on the improvement of the daily infected rate? explain why (we know it is more complex of what you can explain with your data)
    b. How is the progression going each ten days? And each month?

Option A+

There are different A+. The groups can do the ones they want:

1. How can you put your flask server with a public IP? realize that flask starts the server in a private net as default (localhost)
2. How can you put your flask server with a public URL?
3. There are more urls from where to collect Covid-19 data. Collect from one or more different urls and merge the new information by columns with the original. Try to find the country's populations.
4. In order to practice OOP and engineering/architecture concepts in computing, define **all** the functions inside classes and make the program functional using them. After that, use a program to create the class diagram.
5. Get the total deaths, new deaths, new cases and total cases for your countries using web scraping from this website.
6. Using flask and html/css, create something like the dashboard of this website using your countries.