**GOFUNDME : Report of work**

Alban d’Hauthuille

Thomas Lance

Victor Bakache

Manon Krook

Hannes Marks

Ulrich Laitenberger

**INTRODUCTION:**

We decided to look into a crowdfunding company called GoFundme. This platform aims to allow anyone who wants to publish a project and raise funds. We therefore want to collect different data that would allow us to understand which factors are responsible for the success or failure of a project.

This platform is in the form of a web site whose url is : <https://www.gofundme.com/fr-fr>. It allows individuals or private companies to publish a project and call for donations.  To collect the features of a collect : number of donors, creation date, description, category, amount collected, amount targeted, town, title, etc.., we used Selenium to navigate on the website and then a parser, BeautifulSoup, to get theses features from the html code of each page. To analyse the data, we use matplotlib to have a first look, matrixes of correlation and mutliples OLS regression (statmodels) to analyze the influence of the differents features on the amount collected.

We found several conclusions, some obvious like the greater the mean donation of a collect is, the more successful is the collect; but others are surprising, for example the duration of a collect is not that much correlated to its success. But most and foremost, the correlation between the presence of ‘keywords’ in the description is very important !

**Data collection strategy:**

As planned in step 1, we had to go to the page of each collection and for that, we had to use Selenium to navigate and BeautifulSoup to retrieve the different characteristics. As blockages, we didn't have any capcha to deal with but if we didn't put a time.wait(), the loops failed. Also, it was sometimes complicated to locate exactly the tag of such or such figure displayed on the screen, so we had to try inspecting the code source of the webpages. For the number of donors for example, we had to try with Selenium and then BeautifulSoup, using first the class, then the XPATH and then the selector or the CSS path. Sometimes projects have no donors for example and so the tag linked to the number of donors does not exist, we had to use ‘Try’ loops. Moreover, the webscraping program was very long to run because of the navigation on the different collection categories and then on the page of each collection, about 10-15  minutes with fiber connection and powerful computers. That is why we decided to collect 96 collections per category, i.e. 18\*96=1728 collections in total. We had 1200 at the beginning but we decided to let the programme run longer to get more. To discuss 'How representative is our sample?' The answer is that we took the collections that would correspond to a person browsing the homepage of each category and looking at the most popular collections displayed by the platform. This means that they are the ones that a normal person would look at, without for example using a solidarity link that points to a specific collection. This being the case, such a collection would surely be displayed in the popular collections. *Figure1* belowshows our first DataFrame before cleaning.

Une image contenant table

Description générée automatiquement

Figure 1: The Data freshly scraped

After this scraping work, we had a lot of cleaning to do in preparation for the analysis, so we decided to create a notebook 'II' just for cleaning. First we checked that each column corresponded to the right row, for example the right amount associated to the right collect, we were sure of this thanks to our 'for' loop which scrapes the data as it returns 'None' if the tag does not exist in the html parser. It consisted of several steps: retrieving the data that was of type 'objects' in the DataDrame to transform it into strings or floats. We often had to tokenize the strings, remove the ., €, %, :, transform the string 1k into 1000 and 1M into 1 000 000 in the columns of 'amount collected', 'description', 'city' etc. We had to create a program that transformed the sentence 'Created 2 days ago' into a correct DateTime in the format 'MM-DD-YYYY' and that calculates the difference with today to get the duration of the collection (for OLS and plots). We also transformed the string column 'Description' into a tokenized word list column of length greater than 4 to avoid 'the', 'and, 'an', etc which are not interesting for the word study.last but not least, we removed the 4th quartile of each category on the 'Amount targeted' feature because there were false collections which distorted all the analyses, for example: "Give me 1B€ to buy watches"...

Finally, we had the DataFrame of the *Figure2*, ready for the analyse in the notebook III.

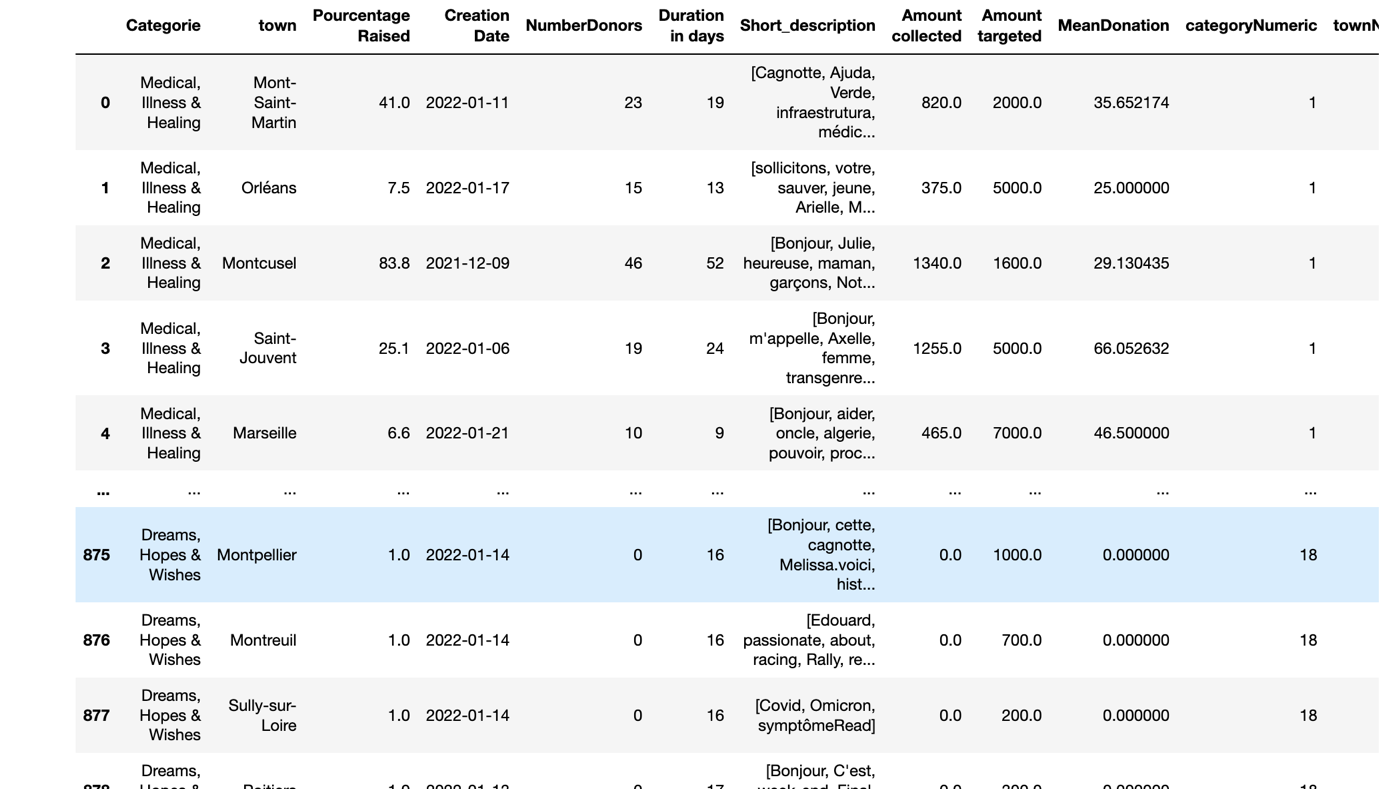


Figure 3 Data cleaned