Delft University of Technology

Computer Science Project
EWI3615TU

Web Scraper for 510

Project Development and Documentation

Authors:

Samuël Verdaasdonk (4962079)
Robert Praas (5426537)
Haobai Guo (5180333)
Thomas Mouton (5427061)
Sofie Neuteboom (5426855)

January 22, 2021

Contents

Chapter 1

Introduction

1.1 The project

As a group of students from the Erasmus University, studying Econometrics, the most obvious and self-evident choice of a topic for our computer science project would most likely be concerned with any financial or standard economic phenomenon. However, to broaden our horizon and to discover what more can be done with our recently acquired programming skills, and possibly also with our econometric skills, we decided to investigate other potential endeavours. What if we could build a tool that could be applied to reality, with which we might also contribute towards a greater good? Luckily, we came into contact with the Data Analytics department of the Dutch Red Cross, 510. They were happy to hear that we would like to help them out with data scraping of the web. After having met with them, we gained a clearer picture of what they needed precisely and for what purpose. We had discovered our real-life use case for this project.

1.2 The Red Cross: 510

Before we dive into the challenge, let us begin by providing some more context to 510. As mentioned before, this organization is a sub-agency of the Red Cross. It deals with everything data-related that is valuable to the Red Cross. They describe their work as "contributing to open data [and] data analysis whilst using the latest techniques to build capacity in governments & NGOs" and underline the relevance and importance of this to "the understanding of humanitarian data" [1]. Their work is specifically fruitful for dealing with disasters in appropriate ways. For example, before a disaster takes place, they can construe a digital risk assessment, when a disaster is impending, they enable early action based on predictive impact analytics, when a disaster is ongoing, they can perform a predictive impact analysis and afterwards, they assess the appropriateness of the (possible) response(s).

Figure 1.1: 510

1.3 The challenge

19 November 2020, Rotterdam, the Netherlands

[1] https://www.510.global/what-we-do-3/

We are extremely lucky to live in a safe country. Of course, we have had our share of disasters, especially considering the flood disaster in 1953. However, we learned to protect ourselves against it, and we succeeded in this quite well, he renowned 'Delta Works' are now world-famous! The contrast with many, countries from the Southern hemisphere could not be bigger. Currently, 58 disasters are going on in the world. They range from hurricanes and volcano eruptions to floods and droughts, you name it. To be fair, we were all quite surprised by this. When we thought it could not be worse, it became apparent that these challenges got even less attention, because of the COVID-19 pandemic as this distracts us from other ongoing severe disasters.

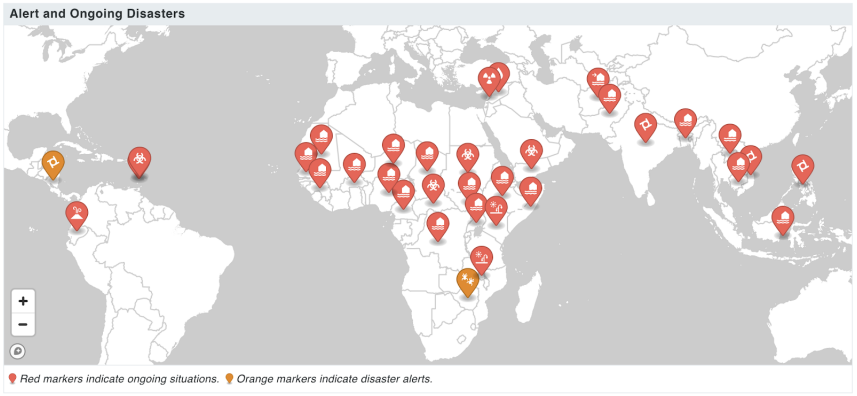


Figure 1.2: Live disasters: source: https://reliefweb.int/disasters?view=ongoing

Luckily, there is a lot we can do to predict, prepare for and deal with the disasters by using the information about the disasters from 1980 up until now. They are carefully reported at ReliefWeb [2], a website from the UN organization OCHA. It shows us which countries are affected and regularly updated with reports. These reports (from official organizations) contain heaps and heaps of impact data. To name a few, they record the number of people affected, money that is asked for by humanitarian organizations and the exact location of the disaster. Most of this information can be found in the regular UN reports, however, it is not readily available and especially the exact location of the disaster can be hard to find. Our tool should change this.

During the project we learned that most documents were not structured very well. Especially before 2012 information inside them could be stated all over the place. The document type that we judged of best use was that of the final reports of the IFRC (International Federation Red Cross). We used these type of documents to get additional information of around 700 disasters. 510 can use this data to predict the impact of future disasters and to make sure that the world can prepare accordingly. Also, as the reports are classified per country, it was quite hard to pinpoint exactly where a disaster was happening, which is very relevant to providing aid to the local communities are actually in need. To this end, we decided to extract this specific data as well, using text mining.

It was our first and foremost objective to create an impact data set of past disasters, which can also be arranged by country. We are proud to present the two first data sets, but also see tremendous opportunity for additions in the future.

1.4 Analysis of similar initiatives

Before we start scraping it is worthwhile to consider what kind of information and datasets are already available concerning natural disasters.

[2] https://reliefweb.int/disasters?view=all

1.4.1 PreventionWeb

PreventionWeb[3] provides a good summary of disaster datasets. Notably, there are different datasets with disasters affecting specifically US, Canadian or Australian citizens. Below we will outline the most important alternatives.

Emdat

The dataset from the centre for research on the epidemiology of disasters in Brussels is the closest aligned to the goals of our dataset. They describe it in the following way: "The database is compiled from various sources, including UN agencies, non-governmental organisations, insurance companies, research institutes and press agencies."[4]

Desinventar

This is a website from the United Nations as well. As they mention themselves, they focus on "the construction of databases of damage, losses and in general the effects of disasters" [5] Their dataset contains a lot of variables, but unfortunately, is not very up to date. Also, it lacks some variables that the Red Cross is specifically interested in.

Our world in data

This website uses various US sources, scattered per disaster. It is a good example of visualization, and we will therefore use it as an inspiration. [6]

Reliefweb

This is the source that we will use to scrape our data. It experiments with making information available in an easier way, yet most of their initiatives are in an early stage. [7]

All in all it seems that there is a lot of information available. Yet, 510 does not need just any disaster data, it needs data on specific variables from verified sources. Below we list the most important ones.

- Type of disaster
- Location of the disaster
- Date of disaster

In combination with other variables and official documents of the Red Cross itself, the UN and other NGOs, where extra information can be extracted from, this would compose a valuable impact data set. Since 510 is immediately called upon when a natural disaster occurs, it needs a tailored solution for the information they require. That is exactly what our product is meant for. Our dataset links our data to the source, gives the specifics of the disaster and provides some extra variables considering the budget, operation dates and people affected.

[3] https://www.preventionweb.net/risk/datasets
[4] https://www.emdat.be/
[5] https://www.desinventar.net/DesInventar/
[6] https://ourworldindata.org/natural-disasters
[7] https://labs.reliefweb.int/

Chapter 2

Project approach

This chapter will present our project approach. We will expand on our design strategy, its objectives, the critical features as well as on some risk factors and 'to-avoids'.

2.1 Design objective and strategy

The main objective of this project can be defined as retrieving data of the disasters that are documented on ReliefWeb. Eventually, this led to the creation of two datasets, which are explained in more detail below. Additionally, we also aimed to visualise some of our data. During the project, each group member worked on different features in different branches. For instance, one team member would create a webscraper, another the PDF miner, and so on. After we finalised all our code we made a well-structured master branch by merging our branches into the master. It should further be noted that we wrote our Python code in PyCharm.

2.1.1 Dataset I

The first step in creating the first dataset was finding and downloading the files from which the data could be mined on Reliefweb. After carefully comparing all the different types of reports that potentially contain the data we need, we decided to start with situation reports from the IFRC. This choice is mainly based on the fact that these reports all have the same structure. In these 'Situation reports' there is a distinction between different kind of formats (like 'Emergency plan of action' and 'final report'). Since final reports are most abundant and contain final information, we chose these for PDF and text mining. After this choice was made, a program (webscraper) was written to automatically search for the PDF download link of all final reports from the IFRC on Reliefweb. This webscraper also scraped some additional information about the disaster connected to this final report. Namely, the URL link of the page where the information was found, the disaster type(s), primarily affected country and other affected countries. Everything was stored in a CSV file, as this was simply the format that the Red Cross preferred. It should be noted that for the webscraping we used Selenium Webdriver.

Table 2.1: Variables contained in Dataset I

Variable Name
Url
Glide number
Disaster type
Operation start date
Operation end date
Budget
People affected
People assisted
Affected countries
Date
Cities affected

2.1.2 Data set II

During the midterm presentation for the Red Cross they clarified that they would also be fond of us working on another dataset, in addition to this first dataset. This dataset would have to consist of all the disasters on ReliefWeb. Together with the Red Cross, we carefully decided on which variables to include for each disaster. This resulted in the following variables: 'Disaster Name', 'Date of occurrence', 'Disaster Type' and 'Country'. Since all variables necessary for this dataset could be obtained from the webpage, no mining of pdf documents was necessary to this end. Again, the preferred format of storing the data was a CSV file.

2.1.3 PDF and text miner

We mine data related to the variables in Table 2.1. To do this, we first divide the PDF file into smaller blocks and search through these blocks to find the keyword that is related to the target variable. After finding the block that contains the keyword, we take out the text from the entire block, analyze it and then find the variable that is associated with the keyword. We performed the above procedures for every variable in Table 2.1, stored them in a Python data structure and eventually wrote the data in the CSV file that is dedicated to our Data Set I.

2.1.4 Visualisation

Our priority in visualizing the data was creating a heat map. Additionally, we also wanted to see what other forms of reporting we could use for the data that we retrieved. Eventually, we ended up producing several bar plots, count plots and histograms in addition to the heat map. More on this can be found in the results section

2.1.5 Other report types

The IFRC is one of many organizations providing situation reports. Table 2.2 shows, there are 60.000+ situation reports from different organisations that can potentially be scraped in the future. Therefore, we can try to replicate the aforementioned sequence of 3 steps, with additional scrapers. These additional scrapers are necessary because these documents are structured differently than the IFRC document, or they are structured inconsistently. For example, we had hoped to continue to build a scraper for the 'Emergency plan of action' reports of IFRC, followed by scrapers for other structured reports like those from USAID. However, due to time constraints and the additional dataset that we had to produce after the mid-term presentation for the Red Cross, we chose to focus on one type of document thoroughly instead of obtaining a larger quantity and lower quality.

Table 2.2: Organizations and number of situation reports

IFRC 10,977
USAID 4,690
Unicef 6,856
UNOCHA 21,592
UNHCR 9,192
WFP 11,411
Fews net 12,085

2.2 Risk factors

At the beginning of this project, we already foresaw quite a few risk factors. Below, we state those that proved to be obstacles that we had to deal with, or which might cause problems in the future.

1. Double data because of the use of different data sources or organisations.
2. Bad document type identification, the code will fail because it isn't compatible with document types which are structured differently.
3. Obtain data which is solely focused on specific groups (like UNICEF focuses on children).
4. Missing data points (when documents do not include information on all variables).
5. Our program is solely based on ReliefWeb and its structure. If either ReliefWeb crashes or its structure changes the mining of PDF files will fail, and thus our database would not be updated.

Chapter 3

Requirements Engineering and Constraints

The web scraping tool for 510 has been designed in such a way that it attends to functional as well as to quality needs. At the same time, certain constraints specific to this project had to be taken into account.

3.1 Functional needs

All of the functional needs have been classified according to the MoSCoW method. In this way, it became easier for us to prioritise tasks. At the start of the project, we formulated the following requirements. In Chapter 16 Evaluation we evaluate whether the functional needs were achieved.

Must have:

- Build a web scraper, that searches *https://reliefweb.int* for PDF-documents and retrieves them or the relevant information that is in them with the help of the PDF Miner.
- Build a PDF miner, that can retrieve the content out of the PDF files.
- Build a program that extracts certain key values and facts (e.g. the number of deaths, total cost, type of disaster and location) from the content and adds it to a data set.
- As an end-product, create an impact data table with information about disasters of the past years gathered from ReliefWeb.
- Create heat maps for specific disasters or of all locations of disasters as a visualization of the retrieved data.
- Calculate and exhibit summary statistics of people affected, money needed, etc.
- Documentation to show a user how to use the tool.

Should have:

- Classification by type of disaster, showing which ones have the largest impact.
- Documentation of missing data.
- Automated way of dealing with reports that appear multiple times.

Could have:

- Information from additional report types, like 'appeals' or news and 'press releases'.
- Provide disaster summaries with a map and summary of the disaster.
- Time series plots of aggregated disaster data.
- Evaluation of disasters.

Won't have:

- Prediction of future disasters.
- Information from unofficial data sources.

3.2 Quality needs

One of the first issues related to quality we came across when working on the project was that it took a long time to scrape all the reports from ReliefWeb. It amounted to more than 5 seconds per link. Therefore, we set the goal of bringing this down to under 2.5 seconds, in which we eventually succeeded. We also discovered that most reporting types were highly unstructured. Consequently, we had to choose between scraping many reports with little information or scraping fewer reports with more information, we chose for the latter.

3.3 Constraints

One of the first constraints with which we dealt was that there are several documents for the same disaster that contain different, and perhaps even conflicting numbers. One way of dealing with this might be temporarily storing it as a different event, after which the data can be reviewed manually so that it can be decided whether it is indeed the same events and whether we take an average or deal with it in another way. However, eventually, we chose to solve this by only using the latest report of a disaster.

Another, perhaps even more important constraint, is that not every report contains exactly the same data and presents this data in the same way. We chose to mine mainly key figures that were easier to find than other information hidden in texts.

Furthermore, not always all the data for a specific disaster could be retrieved. We learned that with scraping it is worthwhile to scrape higher volumes and accepting missing data. An alternative is inspecting documents manually if time and scale permit.

Finally, text mining sometimes provided unexpected results like missing zero's in the number of people affected/assisted. On the other hand, sometimes more people were reported to be assisted than affected, for instance for the cholera disaster in Sudan. We assume this was reported because more people were helped than that got the disease.

Chapter 4

Design documentation set layout, with configuration management

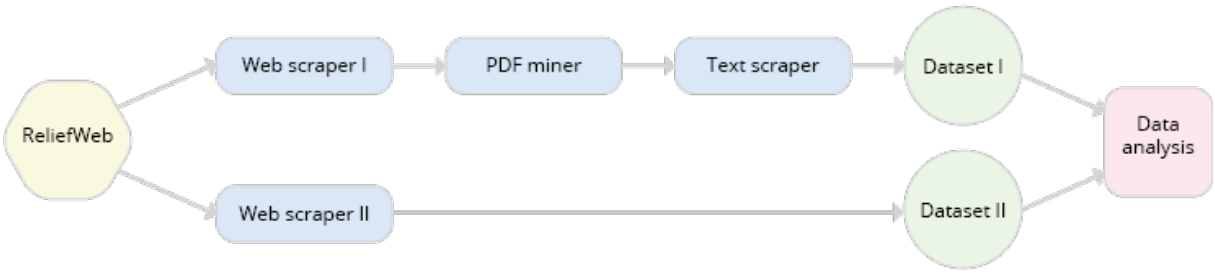


Figure 4.1: Pipeline

In order to fulfil our above-listed project requirements, we had come up with a design to implement them. We decided to fragment the project into the following subsections, which should be able to function independently from each other. Configuration management is done based on these definitions.

- Web Scraper I: Program that navigates through the website, searches for the information we need for Dataset I and collects the following attributes of each disaster: 'website URL', 'PDF download link', 'disaster types', 'primary country affected' and 'other countries affected'. It can be tested both automatically as well as manually and its specifications can be altered to search and extract for different information.
- Web Scraper II: Program that navigates through the website, searches for all disasters on the website present per disaster type. It then extracts the following information per disaster: 'Disaster Name' and 'Country of occurrence'. It also generates from these two variables the following extra variables: 'Disaster date' and 'Disaster type'. The last will be extracted from the initialization.
- PDF Miner: With the PDF documents at hand, it will extract all textual data from the PDF file. This should work based on the type of report it receives. This again can be manually tested. Moreover, in the future, it can be refined when the PDF file contains "tags" so as to convey more information on the text.
- The Text Scraper: Given HTML files with plain textual data, it extracts the variables of interest. The main focus of the text scraper is to extract the precise location, hence identifying and extracting the cities and their corresponding countries that are mentioned in the report.
- Data set I: This data set will contain information from both the web scraper I as the scraped PDF.
- Data set II: This dataset will contain information scraped by web scraper II
- Data analysis: given the created datasets we will then process the data to provide visualisation.

Each subsection of the project will allow testing and can be further refined or adapted if circumstances should change. By making fulfilling each subsection we should be able to attain all our above-mentioned requirements.

Chapter 5

Design Validation

In order to validate the design of our project we set quality standards below of which the project should comply with.

5.1 Design of the finished pipeline

- Extraction of disaster reports: Final Situation Reports published by the IFRC (International Federation of Red Cross)
- Use summary on ReliefWeb to retrieve affected countries, disaster type and link to PDF
- Scrape the main table from the PDF to get variables like operation dates, budget and people affected
- Get the location and date of disaster by packages/heuristics
- Use impact dataset to visualize disaster spread over countries

A simplified version of the pipeline is used to extract a dataset of all disasters on ReliefWeb. For this no PDF is used.

5.2 Configuration management

Web scraper

- Make sure every disaster from the past seven years is being selected, this number is chosen based on visual inspection of reports, which need to have a main information table to be informative
- For most disasters, multiple countries are affected. We scrape the most recent situation report for every country and treat the disaster per country as a different disaster. This is in alignment with the reporting. The scraped situation report must first be a final report from the IFRC and hopefully can later also have another structure/be from another situation
- the collection of scraped disasters reports must be able to be fed into the PDF miner
- if we scrape reports from different organizations, we will assign an ID to them similarly but treat them at first as different disasters.

PDF and Text Miner

- extract text from the scraped PDF text files are saved and can be fed into the text scraper
- Text scraper search text based on a predefined structure and variables for similarly structured documents save the variables found to the impact dataset

- if we scrape documents that are not similarly structured, search for variables based on IDs (variable names)
- find a more specific location than country if available
- find the date of the disaster if available

Impact dataset

- this is the most important final product
- this is a CSV file
- if we find it easy to add more variables, we will not hesitate to do so here we check whether data is saved in right format, else go back to the miner and/or scraper to solve these problems

Analysis

- here we visualize the retrieved data
- we show the amount of disasters per country
- we use a heat map and other plots to make the data better interpretative

Chapter 6

Implementation planning

To make sure we stay on track during our project and work as efficiently as possible, there are several means with which we hope to achieve this. They are explained below.

6.1 Communication

To make sure we are all up-to-date with the project's progress and make sure that we are able to attain our objectives, we have a weekly SPRINT meeting in which we set a weekly planning to achieve the next tasks. During this meeting, we also evaluate the previous week, which is documented in a weekly retrospective, such that we can avoid or ameliorate the inadequacies and continue the ways of working that are already going well. Besides the weekly SPRINT meeting, we have a shorter meeting during the week in which we discuss issues that have come up and talk about decisions that have to make together. Besides, group meetings we also have a chatroom where we keep each other updated about what has been done so far or when we plan to work on certain tasks, so as to have a nice continuous overview of the group's progress.

6.2 Efficient division of tasks

Also during the weekly SPRINT meeting, the SCRUM master will divide the tasks among the team members or subsets of the team. The tasks with the highest priority are assigned first. The team members themselves then put these tasks on the KANBAN in GitLab to keep track of progress and comment on tasks whenever necessary. Mostly one task is assigned to two people and then is further evaluated and split up among members individually.

6.3 Code Reviewing

If a task is finished, the team member who finished it will ask another team member to review the code. This team member can then share his feedback on GitLab, our chatroom or via any other form of communication. Upon review, usually, the author then improves his code also taking into account the output given by pylint so as to improve the functioning and the readability of the code. After this, the parts that have been finished can be included in the final report.

Chapter 7

Testing approach, test planning, validation reporting

7.1 Testing approach

It is important to implement testing for the programs to ensure that they meet the functional requirements. Also, for the web scraper, it is very important to implement (small) tests and program exceptions (how to handle) for when errors occur. This is because of the fact that the web scraper runs for quite some time and you do not want the code to stop running. On the other hand, automated testing for data scraping can be rather tricky since we do not have any data that can be used for validating purposes and it is quite hard to write tests for programs that contain web drivers. The solution to this problem is manually scraping part of the data and using these data to test the programs. Of course, whenever we implemented small functions which do not depend on the continuation (the code will not stop running because of this error) of a program we could write small tests to validate its functionality. An example of functions like this is writing a line to a CSV file.

7.2 Test planning

Before we started writing the code we could, of course, only guess about some of the errors we would face. Therefore we focused beforehand on the following: The tests would target two main functionalities of the programs. The first functionality is finding and downloading the desired type of files from the appointed websites and its corresponding information like for instance 'Primary country affected'. To test this part, we manually found and downloaded the files with the desired type together with the other desired data elements from three randomly picked months. These files later were used as a reference to evaluate the performance of the programs related to the first functionality.

The idea of tests for the second functionality (pdf scraper) is similar. We manually scraped one report for all the desired variables. By comparing the results from the pdf miner and what we manually obtained, this report is later on used as the benchmark for us to evaluate the performance of our pdf scraper.

7.3 Validation

In this project we tried to write as much of automated tests as possible. However, due to the usage of complex packages, we had to test a lot of code manually. In the end, we have tested all functionality of the code but due to the many usages of manual testing, we only covered 67% of lines in the project file. Also due to reoccurring errors with the Gitlab pipeline, we skipped 5 tests. These test will pass when they will run without any Gitlab restrictions.

Testing has proven to be worthwhile for this project. Some irregularities in the web scraping were found. Some examples of this are errors due to missing content on web pages and the web scraper randomly catching errors. However, due to testing, the code kept running which was very valuable.

Moreover, the PDF miner was found to extract the wrong date of the disaster if it was on the same page as the table. Due to the testing we discovered this fault and could change our code. It also proved to be fruitful for the visualization, as bugs were discovered in the functions for refactoring the data through this. If this would not have been done, the data would not have been represented accurately.

Chapter 8

Results

8.1 General

We expected to build a Minimum Viable Product that scrapes disaster reports, mines impact data from it and stores it in a database. Based on this MVP, more text mining procedures could be made to make more data available and the impact data found can be analyzed and used for prediction. Let's discuss our results below:

- We built two web scrapers: one obtaining PDFs from the IFRC and one getting a dataset of all disasters on ReliefWeb.
- We built a text miner that mined variables from the main table of the IFRC Final reports.
- We found ways to obtain harder to get variables like the location and date of the disaster.
- We inspected the dataset with sanity checks and updated our code accordingly to account for errors
- We visualized both datasets with plots, graphs and maps

8.2 Data sets

Overview During our project we created two distinct datasets, the first being our initial main dataset which we will call dataset 1. It consists of all the final reports on disasters filed by the International Red Cross extracted from https://reliefweb.int/, the dataset contains final disaster reports from 2014 until present date. The second data set, named hereafter as dataset 2, consists of all recorded disasters types listed on https://reliefweb.int/ regardless what organisation has filed the report. Data set 2 is meant to enrich dataset 1 and give a more global view of disasters, while dataset two is more specific.

Data set 1 Data set 1 contains information on most variables around 700-800 final reports, for each report if available it lists the URL link to the PDF of the final report, the type(s) of disaster that has occured, the operation start date, operation end date, the allocated budget in Swiss francs, number of people affected by disaster, number of people assisted during the disaster, the country in which the disaster occured and the cites which have been mentioned in the report. The dataset has been saved in a CVS file for ease of use.

Below we show a representation in Excel. It showcases most variables are found most of the time.

Url	Glide n¬∞∞	Type of disas	Operation st	Operation er	budget	people affec	people assist	Affected Cou	Date	cities
https://reliefweb.int/sites		['Flood']	31 March - 3	15 October 2	3500000	235000	153417	['Syrian Arab	31/03/2019	
https://reliefweb.int/sites		['Epidemic',	14 May 202(	30 Septemb	322948	33000	6128	['Kazakhstan	22/12/2020	['Turkestan', 'Kazakhstan']
https://reliefweb.int/sites		['Epidemic',	27 May 202(	30 October 2	[]	50000	20639	['Somalia']	18/12/2020	
https://reliefweb.int/sites		['Epidemic',	20/12/2019	30/06/2020	592672	65316	23405	['Sri Lanka']	23/12/2019	['Polonnaruwa', 'Batticaloa']
https://reliefweb.int/sites		['Epidemic']	19 Novembe	19 March 20	344125	9110984	1095000	['Democratic	10/12/2020	
https://reliefweb.int/sites		['Flash Flood	7 Decembe	30 July 2020	133844	15000	4487	['Malaysia']	11/12/2020	
https://relief	OT-2020-000014-GTM		25 January 2	[]	174436	[]	[]	['Guatemala	02/12/2020	['Izabal', 'Chiquimula']
https://reliefweb.int/sites		['Epidemic',	28 May 202(	31 August 2(	170820	2690	2690	['Tajikistan']	30/11/2020	['Vahdat']
https://reliefweb.int/sites		['Cold Wave'	5 February 2	31 August 2(	315292	800000	6291	['Pakistan']	30/11/2020	['Pishin']
https://reliefweb.int/sites		['Flood', 'Lan	26 April 201	12 Months u	305000	25000	5000	['Comoros']	27/11/2020	
https://reliefweb.int/sites		['Epidemic',	06 Decembe	06 June 202(	223977	250000	10000	['Djibouti']	25/11/2020	['Djibouti', 'Arta']
https://reliefweb.int/sites		['Tropical Cy(	08 Decembe	07 March 20	146491	1	2467	['Madagasca	09/12/2019	['Mahajanga']
https://reliefweb.int/sites		['Tropical Cy(	08 Decembe	8 January 20	76903	0	50	['Comoros']	24/11/2020	
https://relief		['Epidemic']	[]	[]	[]	[]	[]	['Ethiopia']	23/11/2020	
https://reliefweb.int/sites		['Drought', 'E	31 January 2	31 July 2020	303325	33157	28400	['Viet Nam']	18/11/2020	['Bac Lieu', 'Tien Giang']
https://reliefweb.int/sites		['Epidemic']	2 March 202	31 July 2020	263806	[]	[]	['Paraguay']	18/11/2020	['AsunciV≥n', 'Central']
https://reliefweb.int/sites		['Epidemic']	05 February	31 May 202(	284572	4272358	989701	['Nigeria']	17/11/2020	['Kano', 'Kaduna']
https://reliefweb.int/sites		['Flood']	26 October 2	30 March 20	152657	26083	30060	['Ghana']	16/11/2020	
https://reliefweb.int/sites		['Flash Flood	[]	22 February	193051	12900	10851	["CV¥te d'Ivo	16/11/2020	
https://reliefweb.int/sites		['Epidemic',	1 May 2020	31 July 2020	375388	70000	21330	['Democratic	01/05/2020	
https://reliefweb.int/sites		['Volcano']	15 January 2	15 July 2020	498602	736802	21768	['Philippines'	06/11/2020	['Batangas']
https://reliefweb.int/sites		['Cold Wave'	18/01/2020	[]	[]	[]	[]	['occupied Pa	26-12-2020:	
https://reliefweb.int/sites		['Epidemic',	[]	[]	[]	[]	[]	['Bangladesh	18/05/2020	['Satkhira', 'Khulna']
https://reliefweb.int/sites		['Epidemic',	12 Decembe	11 June 202(	344152	399840	137592	['Democratic	04/11/2020	
https://reliefweb.int/sites		['Drought', 'E	21 February	31 July 2020	[]	[]	[]	['Argentina']	29/10/2020	['Tartagal', 'Rivadavia']
https://reliefweb.int/sites		['Epidemic',	23 October 2	23 January 2	78140	25582	15095	['Cameroon']	27/10/2020	
https://reliefweb.int/sites		['Epidemic']	[]	[]	[]	[]	[]	['Turkey']		

Here some summary statistics of the dataset 1:

Month	Budget Allocated
01	1.46836e+07
02	4.28888e+07
03	4.48854e+07
04	1.18849e+08
05	3.96543e+07
06	2.06511e+07
07	1.9429e+07
08	3.45842e+07
09	9.33911e+07
10	2.72227e+07
11	1.88617e+07
12	2.1535e+07

Disaster Type	Number of Occurrences
Epidemic	5
Cold Wave	34
Drought	75
Earthquake	28
Epidemic	191
Fire	5
Flash Flood	127
Flood	292
Insect Infestation	3
Land Slide	115
Mud Slide	17
Other	29
Severe Local Storm	40
Storm Surge	6
Technological Disaster	4
Tropical Cyclone	95
Volcano	21
Wild Fire	6

Month	Number of Affected People
01	9.04184e+06
02	6.87747e+06
03	6.27233e+06
04	1.24657e+07
05	1.99169e+07
06	1.12053e+07
07	3.20337e+07
08	4.5788e+07
09	8.90884e+06
10	5.79263e+06
11	1.39213e+07
12	2.13603e+07

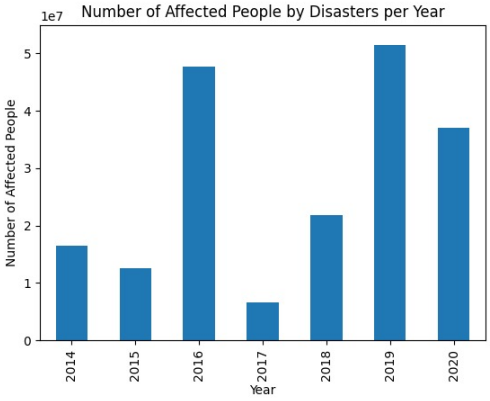
Data set 2 Data set2 contains information on all reported disaster on https://reliefweb.int/ and contains information on 6031 countries that have been affected by disasters, meaning that a disaster that effects several countries, will be listed several times. This has been done to get a better overview on the impact of disasters have on countries, as it helps to control for the magnitude of a disaster. Each data point in the dataset contains the official disaster name, date the disaster started, the type of disaster and the country it occured. Below we show the dataset when disasters are not yet merged:

name	date	disaster_type	country	
Mongolia: Dzud - Dec 2020	Dec/20	Cold Wave	Mongolia	
occupied Palestinian territory: Cold Wave - Jan 2020	Jan/20	Cold Wave	oPt	
Afghanistan: Cold Wave - Jan 2020	Jan/20	Cold Wave	Afghanistan	
Pakistan: Cold Wave - Jan 2020	Jan/20	Cold Wave	Pakistan	
Mongolia: Dzud - Jan 2020	Jan/20	Cold Wave	Mongolia	
Moldova: Cold Wave - Jan 2019	Jan/19	Cold Wave	Moldova	
Algeria: Cold Wave - Jan 2019	Jan/19	Cold Wave	Algeria	
Middle East: Floods and Cold Wave - Dec 2018	Dec/18	Cold Wave	Iran (Islamic Republic of)	
Middle East: Floods and Cold Wave - Dec 2018	Dec/18	Cold Wave	Iraq	
Middle East: Floods and Cold Wave - Dec 2018	Dec/18	Cold Wave	Jordan	
Middle East: Floods and Cold Wave - Dec 2018	Dec/18	Cold Wave	Lebanon	
Middle East: Floods and Cold Wave - Dec 2018	Dec/18	Cold Wave	Syrian Arab Republic	
Mongolia: Dzud - Jan 2018	Jan/18	Cold Wave	Mongolia	
Morocco: Cold Wave - Jan 2018	Jan/18	Cold Wave	Morocco	
Afghanistan: Avalanches and Floods - Jan 2017	Jan/17	Cold Wave	Afghanistan	
Algeria: Cold Wave - Jan 2017	Jan/17	Cold Wave	Algeria	
Morocco: Cold Wave - Jan 2017	Jan/17	Cold Wave	Morocco	
Pakistan: Floods and Heavy Snowfalls - Jan 2017	Jan/17	Cold Wave	Pakistan	
FYR Macedonia: Cold Wave - Jan 2017	Jan/17	Cold Wave	North Macedonia	
Belarus: Cold Wave - Jan 2017	Jan/17	Cold Wave	Belarus	
Mongolia: Dzud - Dec 2016	Dec/16	Cold Wave	Mongolia	
Morocco: Cold Wave - Feb 2016	Feb/16	Cold Wave	Morocco	
Mongolia: Dzud - Jan 2016	Jan/16	Cold Wave	Mongolia	
Peru: Cold Wave - Jul 2015	Jul/15	Cold Wave	Peru	
Papua New Guinea: Drought and Frost - Aug 2015	Aug/15	Cold Wave	PNG	

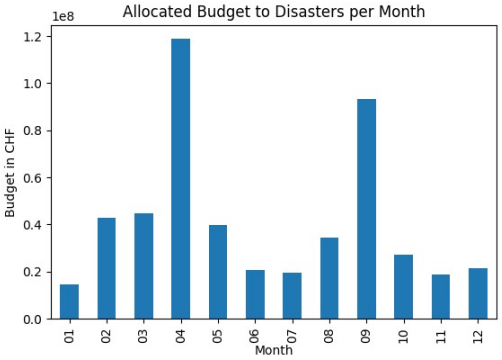
8.3 Visualisation

At the very beginning of this project, it was still quite indeterminate how we would visualize our data. However, throughout the project, more and more ideas and possibilities concerning this presented themselves. For example, the idea of creating a heat-map was something we were all very enthusiastic about, in addition to producing interesting graphs about the occurrences of (different types of) disasters over the years.

It should further be noted that we intended to enrich our data through visualising it, as this was one of the course's objectives. During the process, however, it became apparent that through doing so, we could also 'check' the accuracy of our data. Let's exhibit a few of our graphs to explain this in more detail.



(a) Number of affected people by disasters per year



(b) Allocated budget to disasters per month

Figure 8.1: Plots of dataset I

To start off, the more complicated data from the dataset I, which contains all the data scraped from Final Reports, has also been used to illustrate some more detailed aspects of disasters that have occurred in the past. This is nicely illustrated in 8.1b, which shows the allocated budget in Swiss francs per month, for the different years, which may imply that for during spring and autumn most disasters occur, which may be explained by the change in seasons and hence change in temperatures. In addition, 8.1a shows the number of people affected by disasters per year, however there seems to be no clear pattern although there are some strong changes, which could further be investigated in.

Concerning the visualisation of the data-set from the CSV file called `all_disasters`. This data set has been scraped by the web scraper and includes the variables `name`, denoting the name of the disaster, `date`, denoting the month and year of the disaster, `country`, denoting the country in which the disaster occurred, and `disaster_type`, denoting the type of the disaster. To use this data set, irregular observations for which the scraper did not manage to scrape the proper date, have been corrected manually. If a time period was scraped, this was automatically converted to the several months of these years.

In the graph below, the number of disasters per country over all years has been exhibited. Here, it should be noted that this graph by no means extensively reports all the disasters that have happened from 1980 onward, it only includes those that have final reports on ReliefWeb, but more on this later. The thing that one notices immediately when looking at the graph, is that the Philippines has had to endure the most disasters by far. If we think about this, this makes sense. As the Philippines is an archipelago, located between tectonic plates, this causes a lot of earthquakes, volcanic eruptions, floods, etc. Additionally, it has a maritime, tropical climate, meaning it has to deal with monsoon seasons. In other words, tropical storms are no exception.

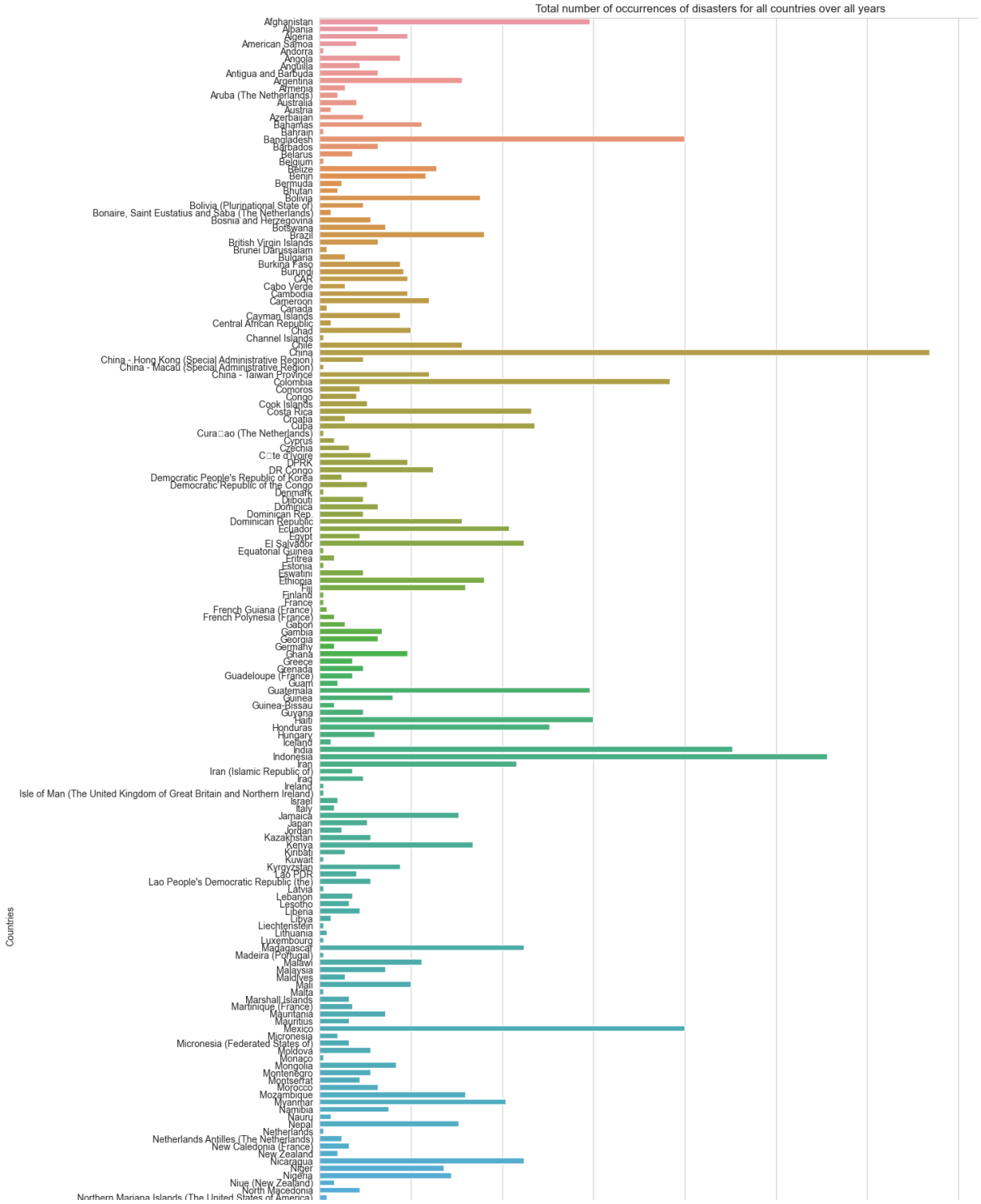


Figure 8.2: The number of disasters over all years per country (continued on the next page)

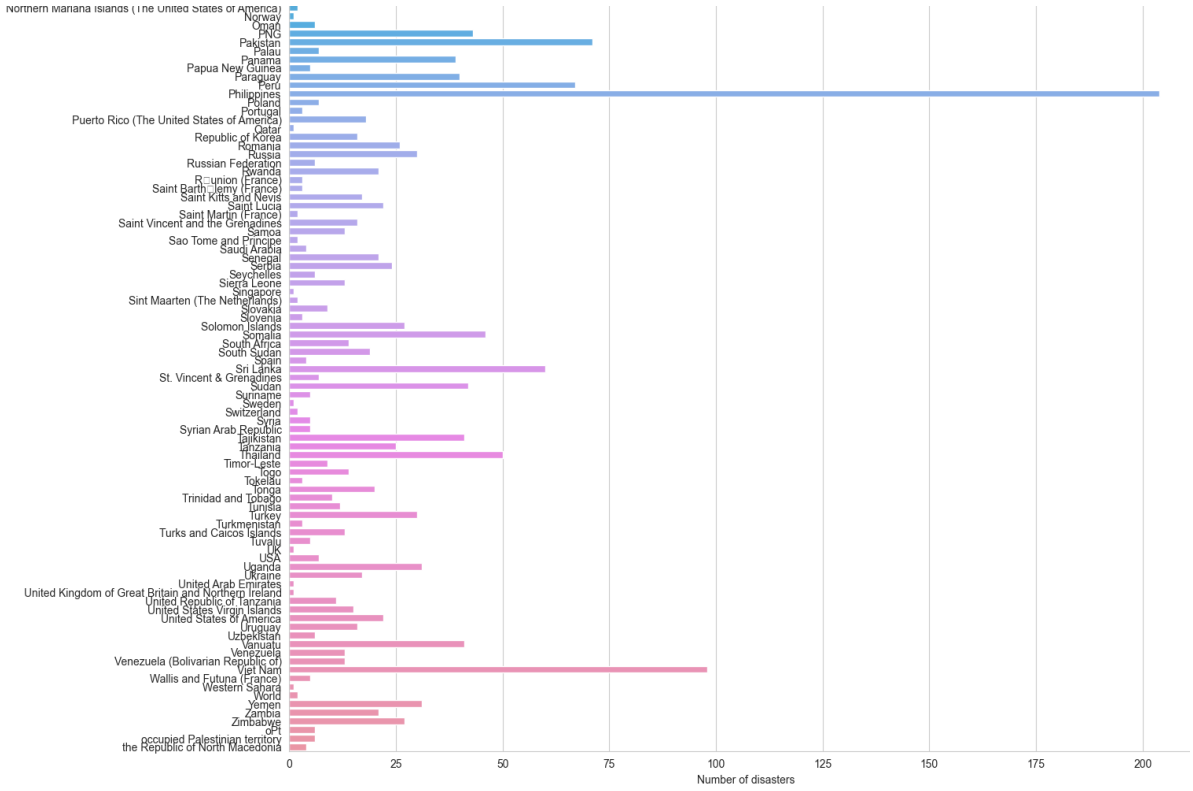


Figure 8.3: The number of disasters over all years per country (continued from the next page)

As has been touched upon in the previous page, it is important to remember that in this section, but also in the rest of the report, we are not dealing with an extensive data set that covers all of the countries that have happened over the past 40 years. Although it is quite unusual to exhibit graphs that do not accurately represent real-world data, let us still do it here, to emphasize the importance of this point. Below you will find a graph representing the number of disasters that were ongoing in each year. One is inclined to think that 2009 must have been a terrible year, especially compared to the other years. Nevertheless, after doing some research, one might start to believe the opposite. In fact, the UN International Strategy for Disaster Reduction reported that there were only 245 natural disasters recorded during that year, whereas in 2005, this amounted to 434!

So, how come that we have such a big misrepresentation in this graph? Well, this is precisely because we are scraping from the website of the Red Cross, which only reports disasters for which help is needed, international attention is turned towards and where countries cannot suffice in aid themselves so that they must receive international aid. Nevertheless, this does not mean that our investigation has become redundant. Quite the opposite, actually, as other projects are going on simultaneously, to scrape disasters from different sources. Together with these projects, a final, complete and clean data set will be achieved.

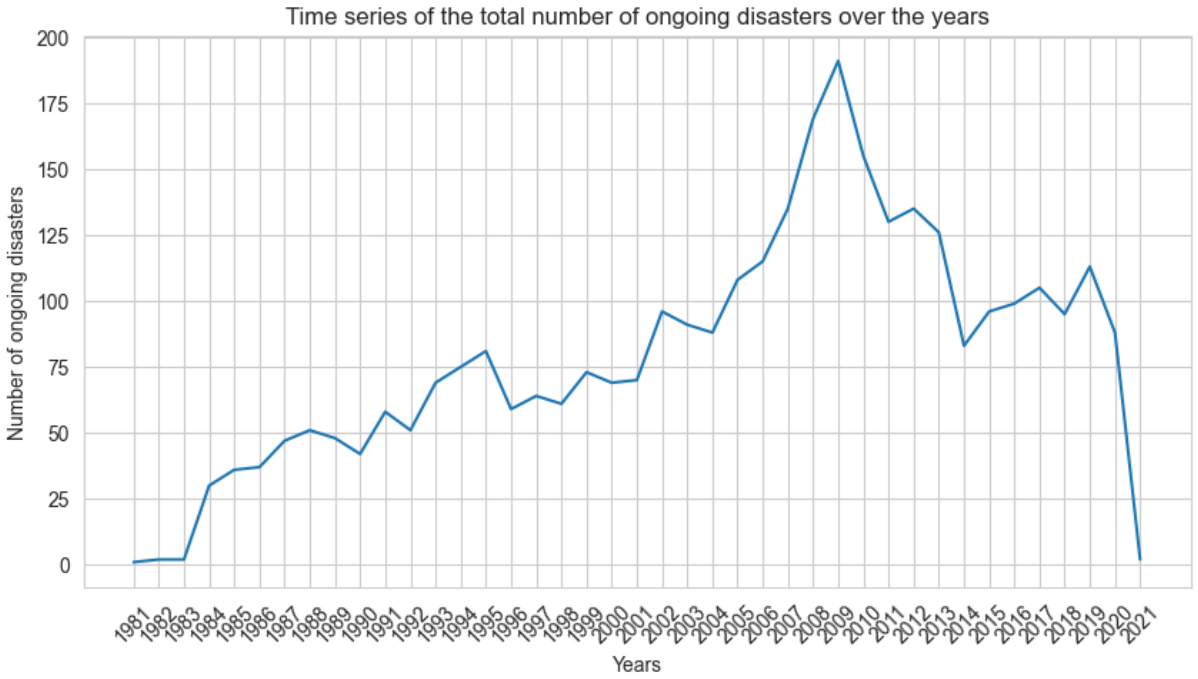


Figure 8.4: The number of disasters (reported on ReliefWeb) over all years

Also, the previous point does not imply that we cannot produce interesting graphs with the data that we retrieved. For example, we can analyze which types of disasters were reported and therefore assisted in most, during the past 40 years. In the colourful plot below, one can the proportion of a type of a disaster during a certain year, to the other disasters. A few interesting trends can be derived from this. First of all, one can derive that the proportion of flash floods and land slides increased enormously over the years. A possible reason for this might be climate change. As many of the glaciers in mountain rich areas are melting, countries like Bangladesh and India have to deal with more water in their rivers, falling from the sky and at the sea. Combined with the deforestation in this country, and other countries alike, causes an increase in landslides as the roots are not able to secure ground. Another interesting trend to note, especially in current times, is that bit by bit, epidemics started to be more common. Perhaps this goes hand in hand with population growth and a pandemic ultimately was doomed to occur, but such a statement would have to be confirmed (or falsified) by the research that is going on today.

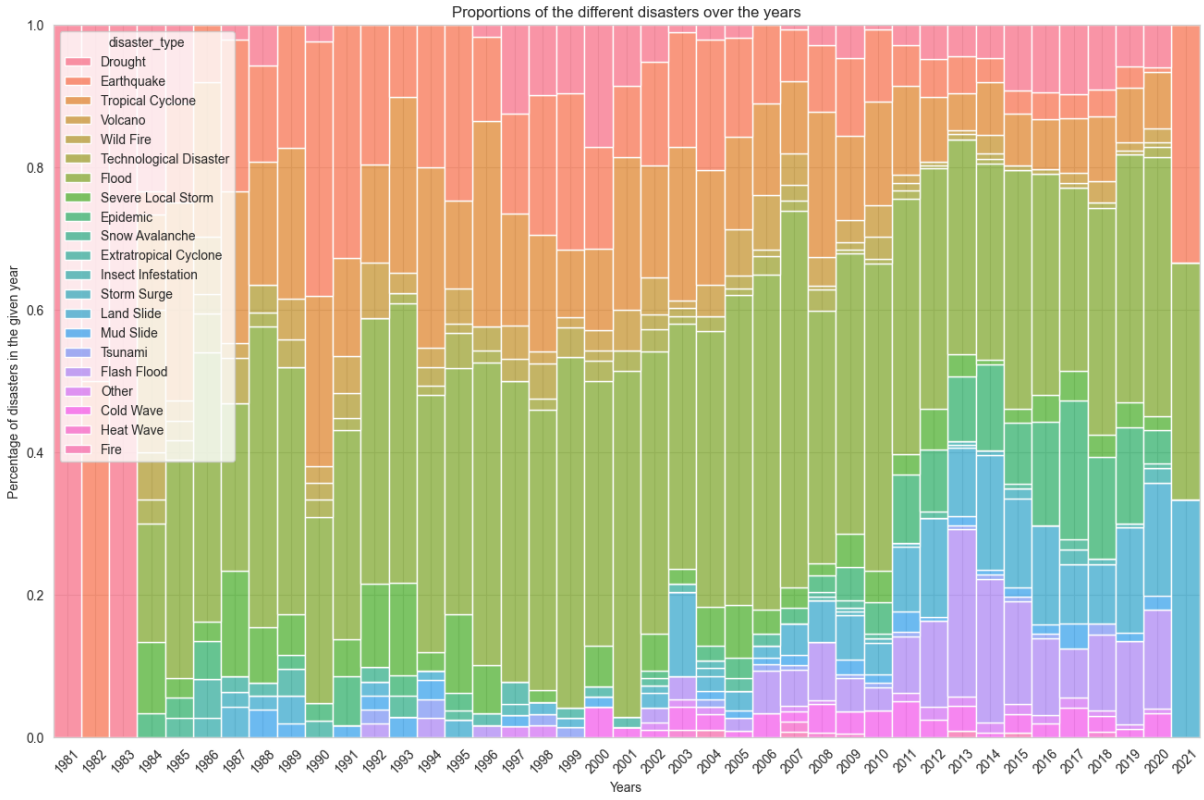


Figure 8.5: The approximate proportions of disasters for every year

As an extra illustration of our data, we exhibit the number of occurrences of a specific disaster type for all countries. From the graph below, we can deduce in which countries the most tropical cyclones occur. As one might expect, these happen most often in countries in tropical regions and for example not in the Netherlands.

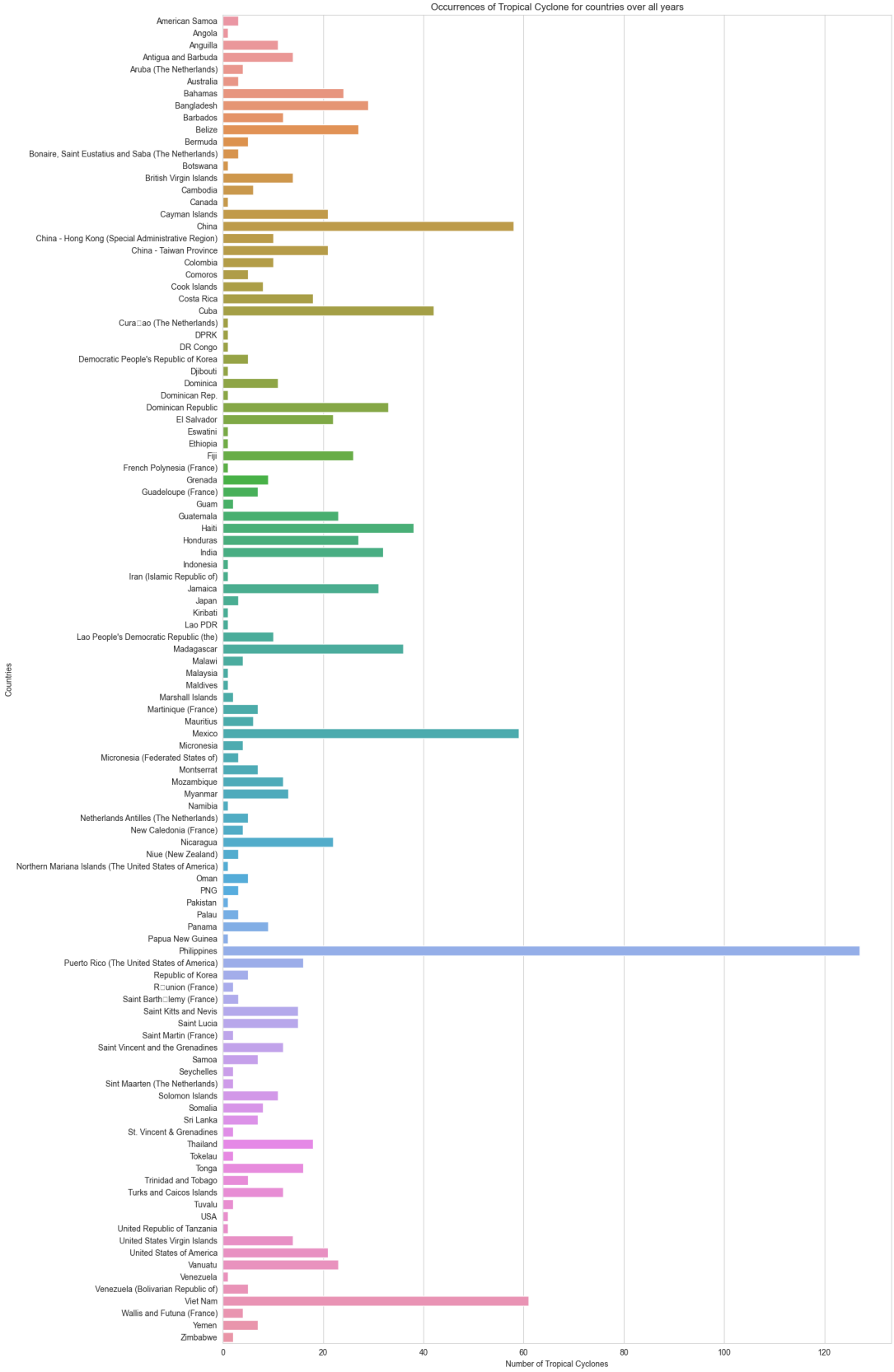


Figure 8.6: The number of tropical cyclones that occurred for every country

In addition, the count of disasters per country from the second dataset was used to produce a heatmap, showing which countries are hit by the disasters the most according to ReliefWeb. Striking are the high numbers for Asian and South-American countries. The Philipinnes were leading in disaster counts with 204 disasters counted.

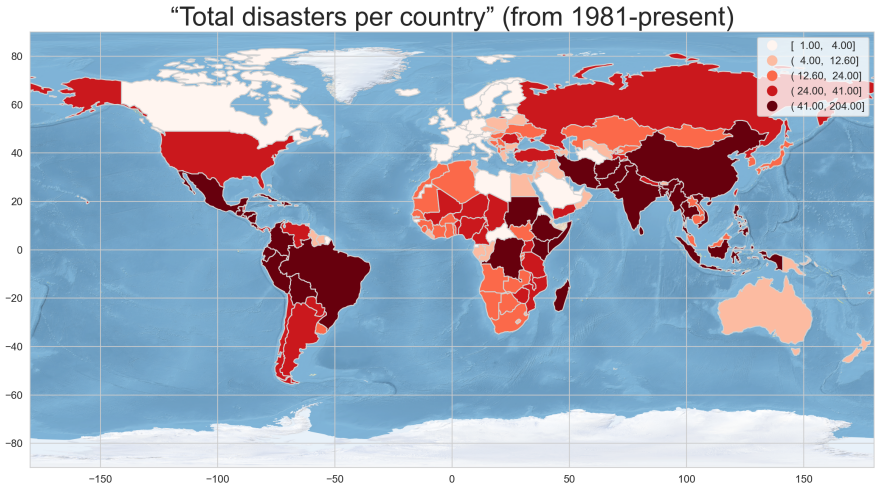


Figure 8.7: Disasters per country heatmap

Chapter 9

Continuation of the project

The goal of this chapter is to give a detailed explanation of the improvements and continuation of this project. In this chapter, we will also refer to the code such that it is clear where one exactly can start. Each major part of the project will be discussed separately. This is done to improve and suggesting further extensions of these parts.

9.1 Design of the finished pipeline

We basically succeeded in the first and last item of the list below. All steps in between are suggestions for improvement.

- Extraction of PDF: IFRC - Situation Report - Final Report, create base data set and data visualization
- add additional scrapers like IFRC - Situation Report - Emergency Plan of Action
- Add PDF sources from different organizations which have a central structure
- Add other PDF sources which are not structured the same (by means of "information searching by search words"
- Create a filter that identifies how the PDF is structured, so the filter tells which PDF analyser is needed
- Combine different data sources; make sure there are no double data/link data sources to the same disaster.
- impact table creation and visualization

9.2 General improvement

We are glad to have shown a proof of concept of scraping disaster and we have determined the most important next steps to be taken for this project. There could be several potential improvements to our project. First of all, our PDF scraping and text mining can be done for the reporting type 'Appeal' which are earlier versions of the 'Final reports' that we used. There are over 2000 appeals from the Red Cross on ReliefWeb which can provide valuable insights into the past disaster responses. The challenge for Appeals is the possibility of there being multiple Appeals for the same disaster, which needs to be taken into account. Moreover, Appeals are early-stage analyses which mean that they are not as accurate as the final reports.

Secondly, the performance of the PDF miner that we developed is only guaranteed when scraping the aforementioned variables from the Final reports on ReliefWeb. So far, these are the only variables needed for the project. However, there is potentially other useful information contained in the Final reports and other types of disaster reports. Our PDF miner could be improved to have high performance in more general settings.

Thirdly, because of irregularities in the documents, data available can seem to be missing in the datasets. In the future a program can be written that detects it and searches in a different way for the data.

Furthermore, more sophisticated data analyses and research questions can be added to this project.

We first envisioned using machine learning models ourselves to classify/predict disasters, yet we learned during the process that obtaining data, cleaning it and structuring it in an informative way, is the current most important challenge.

9.3 Web Scrapers

For this project we have made two different webscrapers namely `scraper_finalreports.py` and `scraper_alldisasters.py`. The first one is used to create 'Dataset I' and the second for 'Dataset II'. We will discuss them separately. A very important side-note is that currently, the code doesn't work because we faced a problem with the Gitlab pipeline (due to its specific settings). However, in the `README` file and in the code itself it is explained to make it function properly. It boils down to uncomment a few lines.

9.3.1 `scraper_finalreports.py`

There are a few things we would suggest in improving this code. Firstly, there should be added a function that checks the last scraped disaster such that the code won't have to run all over again. Secondly its error handling. At the moment only `NoSuchElementException` exceptions are incorporated in the code. This worked fine while running the code because it is programmed for a very basic case (Final reports of IFRC). However, when any changes in file types will be made one could think about more errors that could occur. Next to the code improvements, there are is a lot of room in extending the code and its functionalities. Firstly, this scraper can be extended to a few different extra document types which require changing the `URL` variable to another link. Or looping through a list of links for the extraction of multiple document types at once. Secondly, this scraper can also be extended to scraping all documents on Reliefweb and built and an identifier for each document type. Furthermore, a nice extension to this scraper would be scraping the 'Disaster Name' from the same website the 'PDF link', 'Disaster Type', 'Primary Country' etc. is scraped (which is done in the function `get_data`). With this new variable 'Disaster Name' one could connect the information retrieved by the `scraper_alldisasters.py` because that scraper also scrapes this variable. Hence two datasets could be (partially merged). The only problem to consider is that not every disaster on Reliefweb has a final report. Scraping different kinds of reports would therefore be necessary.

9.3.2 `scraper_alldisasters.py`

For this function, we suggest some improvement in the code as well. The first thing is actually the same as for the other web scraper, namely that there should be a function added that checks which disasters are already scraped such that the scraper doesn't have to go through all disasters again. Again, there is quite some room regarding extending this web scraper. Firstly, this web scraper could extract more information about each disaster (more variables). Also, in this web scraper some problems regarding the dataset could be handled immediately. For instance implement a country name check, which checks whether a country name isn't double (like 'opt' and 'Occupied Palestinian Territory').

9.4 PDF and Text Miner

General improvements consider 1. adding PDF miners for other types of reports: those from the UN and USAID are most abundant, so figuring out structures to obtain data from them is most worthwhile. A second major improvement could be 2. Use NLP to obtain certain parts of the text. Descriptions of the situation, the mention of numbers considering people affected per location, the effect of the disaster on sub-regions are all things that are valuable information, yet hard to find to be provided in a structured

form. The proposed way of doing is by searching for tags like In addition, we advise to update the code in this way:

9.4.1 `pdf_miner.py`

The `pdf_search` function plays a key role in obtaining variables from the main table of the final reports. It is designed specifically for this type of table to perform well and it should be checked whether it performs as well on other types of tables. That is exactly where the biggest wins can be in the future, modify this script for other tables and mine more data. In addition, we found that the same variables are named differently like 'budget' is found to be 'budget', 'operation budget', 'operational budget' and more. We managed to account for this for IFRC reports, yet the same should be done again for other reporting agencies. The `find_date` function uses the heuristic that the first sentence in the situation analysis describes what happened where and when. In that way, the first date after the title Situation Analysis is extracted. With manual checks we found this to be a successful heuristic, yet here again, better performance can be reached by analyzing the nature of the text and extracting a better pattern to be more certain of the date corresponding to the actual disaster.

9.4.2 `location_miner.py`

We found that the geograpy module is a pretty good way to find locations mentioned in a text and since the output is a dictionary it can easily be filtered based on a country. However, the mentioning of a city does not mean that a disaster was there. For instance, the capital name of a country can be mentioned to explain the decisions of the government (in the capital) or it can be used as a reference point to the actual location. In addition, cities are just one way to indicate a location. Therefore, more advanced processing can be used by for instance filtering on words with a capital letter and then using a city database and list of other geographical places like regions, zones and points of recognition. In this way, more locations can be found. To improve decision making about which location should be chosen, we think Natural Language Processing can be used to verify key sentences that explain we are talking about the location of the disaster. The location within that text is then most likely the location of the disaster.

9.5 Visualisation

So far much of the data has been already illustrated using basic histograms, graphs and heat maps. What could further be done, is make more extensive use of heat maps, by subdividing into continents and focusing not on countries as a whole but on cities, which would give a more precise overview of disaster impact. Moreover, using the given number of assisted and affected people for disasters we already scraped, the difference could be computed, hence getting the number of unassisted people. The number of unassisted people could then again be visualised using a heat map to further study, in which areas there are most people that are not receiving much assistance. Furthermore, this could again be plotted against time and one could evaluate how this has changed over time. Moreover, using the data one could even try to run regressions and fit trend lines to the plots and even include forecast windows, so as to give an idea of how the situation may evolve over time. Moreover, one could also make use of third variables such as weather conditions since we realised most disasters in recent times especially are weather-related, for example, one could plot average temperatures with the number of disasters and visualise and analyse those relationships.

Chapter 10

Evaluation

10.1 Risk assessment before the project

Possible obstacles that we may encounter in our project are the fact that not all of our PDF documents will be in the same format since we will be looking at data from the past 20 years and different organizations issue differently formatted documents. However, we may alter the algorithm for different time segments, when the format should change. Moreover, not all documents are in English, so maybe we may refine our algorithm to detect if it is not in English and possibly design it in a way that it can still collect our target variable, one possible solution would be to have the algorithm search differently depending on the language since the languages used are quite limited mostly English, sometimes French and Spanish this should be possible. However, these are risky assumptions considering the length of the project.
Furthermore, another potential difficulty would be the implementation of a PDF "miner", e.g something that extracts the text from the PDF documents, most PDF documents are tagged which should facilitate the extraction. Finally, there could be a memory-related problem as the PDF documents from the past 20 years may take up a lot of space, however, we may analyze the data in smaller time segments to limit the memory used, if this may prove to be a problem.

10.2 Evaluation of results

As aforementioned, the results are put into two data files. The first file contains the data obtained from the around 800 final disaster reports. As not all the final reports contain the data we need, there are some missing data points in the first data file. Nevertheless, we still have obtained a sufficient amount of data that provides insight into the influences of the past disasters and the level of involvement of the charity. The amount of data we obtained were also sufficient for as to produce valuable visualizations as showed in the relevant section. The second data file contains the data of all the documented disasters and we collected over 6000 disasters. Since this file contains only disasters and the country name, it is free from the missing data issue. The huge amount of data also provides us visualizations from which we can have an accurate overview of disasters that happened in each country. Overall, we have collected a sufficient amount of data for simple data analyses.

10.3 Requirements engineering

Here we inspect the result of our functional needs. We are happy to mention that we accomplished all must-haves, which together form the MVP of 'The Red Cross Disaster Web Scraper Text Miner.' Regarding the should haves, these were less relevant to the project than projected beforehand. Classifying disaster types was not framed clearly enough, although we still managed to show which ones occur the most, and more importantly, we have shown which countries have the most disasters according to ReliefWeb. We took care of missing data by tweaking our text miner to minimize this. Nevertheless, some data is simply not reported. Right now the best solution is to dive manually into a document if data of a specific disaster is required. Luckily we kept the PDF link, so it is very easy to access from our

dataset. Lastly, reports do not occur more than once because of the way ReliefWeb is structured, so that was solved by itself. Moreover, from the could haves we provide time series plots of disaster data and are almost ready to scrape appeal documents to enrich our dataset.

During the project, we learned why a tool like this is not there yet. The consequent data gathering of ReliefWeb is already a great accomplishment considering the many disasters, different updates, and immediate nature of the emergency relief sector.

Some highlights of the things we learned:

- Different organizations use different formats and even the same organization with the same type of document can differ per country or over time. However, in our opinion, the quality of the information from these verified documents is worth the trouble to obtain.
- The biggest bottleneck was indeed the structure of the PDF, which was best provided by the IFRC final reports. We succeeded in mining the necessary information from the tables, yet more advanced NLP is required to get more information out of the larger texts.
- Contrary to an earlier project with one app, we now learned that a project with branches that are depending on each other, it is critical to continuously monitor bottlenecks. Namely, if there are no PDF links scraped, not text can be mined. If no text is mined, no dataset is created. Else if the dataset is not cleaned it cannot be visualized and such. Updating other team members on the status of your work is key to keep the flow of improvement flowing.

All in all, we are grateful for the responsibility we got for this project. We learned a lot and we hope this is only the start of a data-driven journey with impact.

Appendix A

Selected log of progress

A.1 Web scraper I

8-1-21.Update by Samuel and Thomas. As before Samuel and Thomas did some brainstorms on how to program the web scraper. We managed to create a working web scraper quite fast. However, the main problem we faced was that the program runs slow. From 8-1-21 we added the feature that the web scraper writes the following variables to the CSV file (which is partially the input for the pdf scraper): html link, pdf downlaod link, disaster type, primary country of occurence, other affected countries. Also, we managed to improve the speed of the web scraper. Hence it would take about 6 hours to scrape all 187 pages (of 20 disasters each). For now we ran the web scraper to extract all disasters starting from January 2012. We still need to figure out how we solve the problem stated in 13.1 (only from 2012 there is this standard pdf form present). If we figure out how to solve this problem we can manually run the web scraper from 2012 until the first disaster recorded on Reliefweb TODO: Code a function such that the web scraper remembers the last scraped disaster (such that it only has to scrape the new disasters once Reliefweb gets updated).

16/01/20 Update Thomas: Added exception handler that were more specific and towards the potential errors. Moreover, the variables got better, more clear names, redundant code was excluded so to make it more efficient and a tiny bit faster. In terms of readability and aesthetics a lot of improvement has been made, now the code is better structured and much cleaner and easy to read. Finally, the failing pipeline has been fixed and the code quality according to pylint has massively been improved from around 8/10 to 10/10.

A.2 Webscraper II

The Red Cross asked us quite late in the project whether it would be possible to create a dataset for all disasters on reliefweb. Together with some visualisation. We discussed that a heat map of all disasters will be a good visualisation. For this we want to have an overview per country how many disasters happened, when and what kind of disaster type it is. Reliefweb offers an overview of all disaster happened per country. However, Reliefweb doesn't offer a clear overview of all disaster types when there are multiple disasters present for 1 disaster (for example: earthquake in combination with tsunami). To fix this, Samuel started to program a webscraper that goes through all disaster types and stores each disaster and connects is to a country (for each affected country there is made a "new" disaster and the disaster type gets stored. If per country there is a disaster present which has multiple disaster types, the disaster will be merged to one disaster and the disaster types will be merged so we can see a list of all disaster types for this one disaster.

A.3 Text miner

8-1-21 Pair programming Sofie and Robert

We delete pdf after usage otherwise it fills pycharm. For location we know that under the title 'Situation Analysis' the first sentence standardly describes the disaster: When What Where —> task for

us to separate it and to put it in the table. We assume the date in the first sentence is the date of the disaster

16-1-21 Haobai: We improved the accuracy of scraping and shortened the scrapped results by removing redundant parts. We adjust the input variable such that the code is more robust to the documents where the desired information are phrased differently. We restructured the codes for PDF scrapping by classifying codes for different functionalities into relevant function, such that the code has clear hierarchy and structure. We wrote automated test for the functions for PDF mining. Next steps: - Heatmap/summary stats (Sofie) - Location from date sentence (Robert)

Red Cross said SQL was not relevant ; focused on relevant parts (Assumption was different from reality)

- Ft Haobai: Notebook schoonmaken, documenteren, pipeline laten werken

- 13/01/2020 separate functions for clearness

14-1 Dataset cleaning: start with date found TODO - add the sentence the date occurs in, such that manually can be checked whether this is what we want (or the first paragraph of descriptions) We experimented with getting the location, it is mentioned in the first paragraph,but no clear heuristic found. Geopgrapy can extract locations but is confused by all the helping countries (US, NL, Swiss frequently mentioned) Number of people assisted: get only the number look at integer, see if thousand/million is missing; Solved not detecting zero's from main table. Same for affected; also get numbers for summary stats 10-12 During our first sprint meeting, wee divided the group into two subgroups. One group (Samuël and Thomas) to work on the WebScraping and the other subgroup (Robert, Haobai, Sofie) worked on the pdf miner. WebScraper: - Works, extracts the links from the website Miner: - Works if you just provide it with the link - But get very big rectangle when you look for something that is contained in a paragraph and not in the table What went well: - The WebScraper and WebMiner already work quite well What could be improved: - The speed of the program, specifically of the WebScraper - Branching - Maybe more specific tasks, although this was difficult because we were still starting and now have a better idea of the tasks at hand. To do: - Connect the scraper and the miner, make sure output scraper is input miner. - Decide how we can retrieve the location from the text blocks. - For the WebScraper add the download link, disaster type and the name of a country to a CSV file. - Improve the speed of the WebScraper (2-2.5 sec per page per link). - Text mining output is also a csv file (for now).

A.4 Data Visualisation

19/01/2020 Thomas: Cleaned the dataset 1 from the final reports, processed the data to create histograms and summary statistics, both on monthly and yearly basis, but also on the entire dataset. Thereby helping to better illustrate the number of people that have been impacted by disasters and when during the year they tend to occur most often.

19/01/2020 Sofie and Robert: Cleaned the dataset 2, that contained information on all recorded disasters, came up with a few graphs depicting the evaluation of number and kind of disaster types over time. Also, heatmaps were created to better illustrate the number of disaster occurrences geographically.