T-764-DATA – Spring 2021

Project 2: Data Quality

Fabien Cogez
Reykjavik University
fabien20@ru.is

Basile Ozenne
Reykjavik University
basile20@ru.is

Ragnar Stefánsson
Reykjavik University
ragnars15@ru.is

February 14, 2021

Abstract

This second Big Data project puts the emphasis on data manipulation. We are given a data set made up of 9 tables gathering data on volleyball practice in Iceland. The data set provided allows replicates, contains spurious and missing information. Our goal is to "clean" it and figure out ways to identify players that might be registered several times under different names.

I. Introduction

In order to identify individuals and organisations in Iceland, the government uses *kennitala*, a unique national identification number administered by the Registers Iceland. When it comes to volleyball, players are identified with their names and birthdays. As namesakes with identical birthdays are allowed, this identifier is not considered as a key. Hence a player can register several times under his/her true name or nickname, for different clubs or in the same club...

This makes the data quite hard to analyse as you can't endure the risk of counting several times the same person because they sign up several times. By giving a quick glance to the data set, we can also see that constraints one might expect are not respected (for example, team ordinal numbers given in the team names instead of in the designated field). For these reasons we need to adapt this data set through process called data cleansing and mapping.

The paper is structured as follows. Section 2 presents the background needed to understand the stakes of this project. Section 3 explains the challenges and our contributions to overcome them. Section 4 shows the results of our works and section 5 gives our conclusions and explores possible future works.

II. Background

Before diving into the core of the project, we need to better understand what data mapping and data cleansing mean.

Data mapping is essential when it comes to data integration process. It is a procedure of extracting data fields from a single or multiple source files in order to match them to their related target fields.

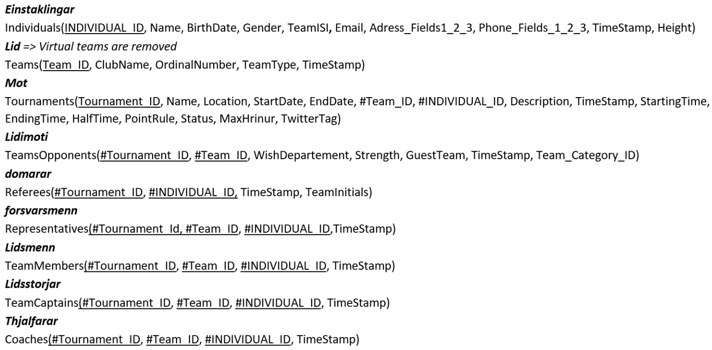
Data cleansing is a process in which errors, spurious information or missing values are fixed. There are many conceivable options but the main ones are correcting or deleting the error [1], get rid off unusable information and fill in blank fields. The major concern encountered is the synonym management. In the provided data set, Icelandic names may be written using a Latin alphabet convention. Dealing with a specific nomenclature by detecting synonyms (e.g. only working with Latin characters) considerably simplifies the data set.[2]

As the given data set may contain irrelevant information regarding the purpose of the project, we are also tempted to simplify it by removing these unwanted information from the data record. The term *data scrubbing* is commonly used to illustrate this concept. The process of data scrubbing gives us the opportunity to work on a data record that is free of irrelevant information that we do not actually require. The methods for data scrubbing are multiple, but most of them need the use of an extraction list on which the scrubbing application work on to lead to cleaned data set.[3]

III. Methods

In this part, we introduce the methods we have implemented to deal with the provided data set.

To begin with, we needed to take a closer look at the data set to get an overview of the tables and their relations. Here are the contents of the .csv files ordered in tables. The names are translated for a better understanding.



Einstaklingar
Individuals(INDIVIDUAL_ID, Name, BirthDate, Gender, TeamISI, Email, Adress_Fields1_2_3, Phone_Fields_1_2_3, TimeStamp, Height)
Lid *=> Virtual teams are removed*
Teams(Team_ID, ClubName, OrdinalNumber, TeamType, TimeStamp)
Mot
Tournaments(Tournament_ID, Name, Location, StartDate, EndDate, #Team_ID, #INDIVIDUAL_ID, Description, TimeStamp, StartingTime, EndingTime, HalfTime, PointRule, Status, MaxHrinur, TwitterTag)
Lidimoti
TeamsOpponents(#Tournament_ID, #Team_ID, WishDepartement, Strength, GuestTeam, TimeStamp, Team_Category_ID)
domarar
Referees(#Tournament_ID, #INDIVIDUAL_ID, TimeStamp, TeamInitials)
forsvarsmenn
Representatives(#Tournament_Id, #Team_ID, #INDIVIDUAL_ID,TimeStamp)
Lidsmenn
TeamMembers(#Tournament_ID, #Team_ID, #INDIVIDUAL_ID, TimeStamp)
Lidsstorjar
TeamCaptains(#Tournament_ID, #Team_ID, #INDIVIDUAL_ID, TimeStamp)
Thjalfarar
Coaches(#Tournament_ID, #Team_ID, #INDIVIDUAL_ID, TimeStamp)

Figure 1: *Tables in normal form*

In the beginning, we tried to use spark with Scala but after following the resources given in the sundogsoftware tutorials among others, we face too many issues with it, as it took a long time to understand how RDD, data sets, and data frames worked in Spark so we could use the data correctly. Then we had problems due to how many things looked similar but behaved very differently like spark.session vs spark.context to initialize and import files. Then we tried to make different kinds of SQL requests on the data, but it was a real pain, as Scala just did pretty much what it wanted, so we gave up on this. In particular we were unable to use the native $dataset.[sqlrequest]$ to do "select" when aggregating, and using $spark.sql([sqlrequest])$ didn't allow very complex requests written in SQL. To recap this first experience with spark was quite unpleasant and very challenging.

All of our results were created using Python since we did not manage to get Spark to work.

************************ The minor changes we did at the start was to correctly refill the Radnumer column in the table lid with correct entries from the end of the name field (if it contained a letter or a number that was not known to be in a team's name) and to reformat and sort the Phone number fields in the Einstaklingar table, to get the data ready for future analysis. ************************

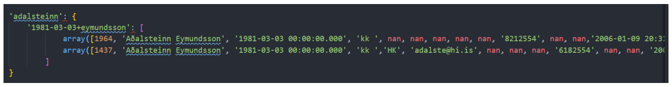
The issue with the lid table was that most people wrote their Radnumer in the field dedicated to their club name. The challenge was to come up with a way to extract the Radnumer in the club name, not knowing whether it was there or not, and to not confuse it with a single character from an initial of the club's name. After looking at some of the data we found out that Radnumer values are always either a single number, or a letter in the range a-f/A-F. Knowing this and the format most clubs used in their names, we where able to create the function $correctRadNumbersFromEntries()$.It checks if the club's name entry ended with a single character separated from the rest, and if this character is respecting the usual convention of Radnumer, it's then removed from the name and put into the Radnumer colunm.

After this, we also rearranged the Phone Fields, and cleaned up these entries from the Einstaklingar table, as some people wrote their phone numbers using different conventions, and some entries weren't filled in the right order. To resolve this issue, we made the $movePhoneFieldsEntries()$ and $convertPhoneEntry()$ functions that respectively reordering the fields to have the firsts filled first, and keep only the numbers from the entries. This way we expected to have the phone fields easily accessible, as now if the first field is empty, you know the others will be too, and if you want to fetch the phone numbers, you should now be able to directly convert them to integers without worrying about dashes and spaces in the field.

These 2 contributions help make the data more readable and accessible for later analysis.

Then comes the main contribution of our work : the creation of a mapping that gives a quite precise list of candidates for merging data. A great challenge of this cleaning task was to come up with a way to detect the duplicate entries in the Einstaklingar table. The main problem being that Icelanders have a limited supply of first names, can write their last names in different ways when identifying themselves as for example : Gunnarsdóttir vs gunnarsd. And on top of that, another problem of this base was that sometimes entries are in all capitals, sometimes only initials are given, and sometimes the middle name is given (or not).

To deal with this, we began our work by creating a dictionary which contained all the birthdays (filtered by sex) that were duplicates, then we created a function called $create_duplicated_entries()$ which would take all of these duplicates and retrieve the name of each person, then it would create a new dictionary containing the key $'nafn'$ and it's values as $'Fdagur + EftirNafn'$ by doing it this way we can easily identify all people with the same first name, birthday and last name (filtered by sex), but the problem with this is that if the name is missing a middle name or last name then those two become two different entries under that name key and this is something we would have to fix in future work.

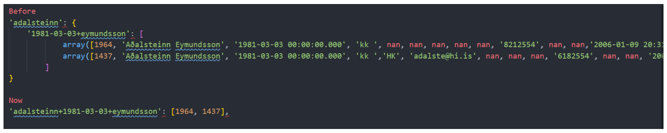


```
'adalsteinn': {
    '1981-03-03+eymundsson': [
            array([1964, 'Aðalsteinn Eymundsson', '1981-03-03 00:00:00.000', 'kk ', nan, nan, nan, nan, nan, '8212554', nan, nan,'2006-01-09 20:3
            array([1437, 'Aðalsteinn Eymundsson', '1981-03-03 00:00:00.000', 'kk ','HK', 'adalste@hi.is', nan, nan, nan, '6182554', nan, nan, '20
        ]
}
```

Figure 2: *Example of how the duplicated data was structured*

After combining all potential duplicates, we then might have some entries which are just single entries because the person with that birthday and of that sex had the same birthday as another person so we want to remove this person from the duplicated dictionary because there is no other another entry for that birthday with that first and last name so we call the function remove_single_entries which is responsible for finding all entries that were wrongly inserted into the dictionary.

We then use the function identifier_map_unique_ids to combine the *birthday+lastname* value key in duplication dictionary to each key .



```
Before
'adalsteinn': {
    '1981-03-03+eymundsson': [
            array([1964, 'Aðalsteinn Eymundsson', '1981-03-03 00:00:00.000', 'kk ', nan, nan, nan, nan, nan, '8212554', nan, nan,'2006-01-09 20:3
            array([1437, 'Aðalsteinn Eymundsson', '1981-03-03 00:00:00.000', 'kk ','HK', 'adalste@hi.is', nan, nan, nan, '6182554', nan, nan, '20
        ]
}

Now
'adalsteinn+1981-03-03+eymundsson': [1964, 1437],
```

Figure 3: *Duplication data before and after calling identifier_map_unique_ids()*

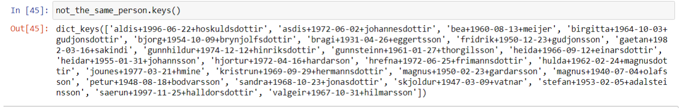
To avoid false positive while merging, we decided to further differentiate people on the basis of their activity. So that if there exists two different people in the duplication dictionary that have the same name and birthday but are not the same person then we have to remove them as duplicates, our main way of doing this was through our first experiment using the team member table (lidsmenn).

IV. Results

We began our main experiment by implementing a function called $find\_duplicates()$. This function was responsible for finding players with the same birthday and name that were playing a match within 20 minutes of each other (normally volleyball game are between 60 and 90 minutes), this way we could identify all the players that we were certain of being two different people and thus they did not belong together in the duplication dictionary.

Doing this time check is not really guaranteed to be accurate since it's hard to determine which data entries were inserted wrongly, for example if entries for the same person was entered few minutes apart but the person inserting it messed up and put in the wrong teamid or einstaklingsid and thus created a new team or person from scratch which is apparent if you look at the table teams (lid), which is just a bunch of wrongly entered team names.

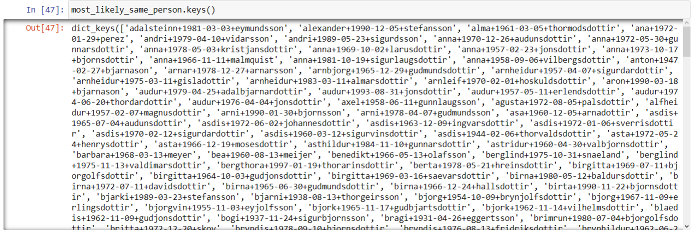
By doing this time checker we eliminated 25 name entries from being considered as duplications which left us with 482 entries that could still be considered as duplication, but for those 25 name entries we would have to manually determine if it was the same person or not in future work.



```
In [45]: not_the_same_person.keys()

Out[45]: dict_keys(['aldis+1996-06-22+hoskuldsdottir', 'asdis+1972-06-02+johannesdottir', 'bea+1960-08-13+meijer', 'birgitta+1964-10-03+
         gudjonsdottir', 'bjorg+1954-10-09+brynjolfsdottir', 'bragi+1931-04-26+eggertsson', 'fridrik+1950-12-23+gudjonsson', 'gaetan+198
         2-03-16+sakindi', 'gunnhildur+1974-12-12+hinriksdottir', 'gunnsteinn+1961-01-27+thorgilsson', 'heida+1966-09-12+einarsdottir',
         'heidar+1955-01-31+johannsson', 'hjortur+1972-04-16+hardarson', 'hrefna+1972-06-25+frimannsdottir', 'hulda+1962-02-24+magnusdot
         tir', 'jounes+1977-03-21+hmine', 'kristrun+1969-09-29+hermannsdottir', 'magnus+1950-02-23+gardarsson', 'magnus+1940-07-04+olafs
         son', 'petur+1948-08-18+bodvarsson', 'sandra+1968-10-23+jonasdottir', 'skjoldur+1947-03-09+vatnar', 'stefan+1953-02-05+adalstei
         nsson', 'saerun+1997-11-25+halldorsdottir', 'valgeir+1967-10-31+hilmarsson'])
```

Figure 4: *Key values of people who did not belong to the duplicated data*



```
In [47]: most_likely_same_person.keys()

Out[47]: dict_keys(['adalsteinn+1981-03-03+eymundsson', 'alexander+1990-12-05+stefansson', 'alma+1961-03-05+thormodsdottir', 'ana+1972-
         01-29+perez', 'andri+1979-04-10+vidarsson', 'andri+1989-05-23+sigurdsson', 'anna+1970-12-26+audunsdottir', 'anna+1972-05-30+gu
         nnarsdottir', 'anna+1978-05-03+kristjansdottir', 'anna+1969-10-02+larusdottir', 'anna+1957-02-23+jonsdottir', 'anna+1973-10-17
         +bjornsdottir', 'anna+1966-11-11+malmquist', 'anna+1981-10-19+sigurlaugsdottir', 'anna+1958-09-06+vilbergsdottir', 'anton+1947
         -02-27+bjarnason', 'arnar+1978-12-27+arnarsson', 'arnbjorg+1965-12-29+gudmundsdottir', 'arnheidur+1957-04-07+sigurdardottir',
         'arnheidur+1975-03-11+gisladottir', 'arnheidur+1983-03-11+almarsdottir', 'arnleif+1970-02-01+hoskuldsdottir', 'aron+1990-03-18
         +bjarnason', 'audur+1979-04-25+adalbjarnardottir', 'audur+1993-08-31+jonsdottir', 'audur+1957-05-11+erlendsdottir', 'audur+197
         4-06-20+thordardottir', 'audur+1976-04-04+jonsdottir', 'axel+1958-06-11+gunnlaugsson', 'agusta+1972-08-05+palsdottir', 'alfhei
         dur+1957-02-07+magnusdottir', 'arni+1990-01-30+bjornsson', 'arni+1978-04-07+gudmundsson', 'asa+1960-12-05+arnadottir', 'asdis+
         1965-07-04+audunsdottir', 'asdis+1972-06-02+johannesdottir', 'asdis+1963-12-09+ingvarsdottir', 'asdis+1972-01-06+sverrisdotti
         r', 'asdis+1970-02-12+sigurdardottir', 'asdis+1960-03-12+sigurvinsdottir', 'asdis+1944-02-06+thorvaldsdottir', 'asta+1972-05-2
         4+henrysdottir', 'asta+1966-12-19+mosesdottir', 'asthildur+1984-11-10+gunnarsdottir', 'astridur+1960-04-30+valbjornsdottir',
         'barbara+1968-03-13+meyer', 'bea+1960-08-13+meijer', 'benedikt+1966-05-13+olafsson', 'berglind+1975-10-31+snaeland', 'berglind
         +1975-11-13+valdimarsdottir', 'bergthora+1997-01-19+thorarinsdottir', 'berta+1978-05-21+hreinsdottir', 'birgitta+1969-07-11+bj
         orgolfsdottir', 'birgitta+1964-10-03+gudjonsdottir', 'birgitta+1969-03-16+saevarsdottir', 'birna+1980-05-12+baldursdottir', 'b
         irna+1972-07-11+davidsdottir', 'birna+1965-06-30+gudmundsdottir', 'birna+1966-12-24+hallsdottir', 'birta+1990-11-22+bjornsdott
         ir', 'bjarki+1989-03-23+stefansson', 'bjarni+1938-08-13+thorgeirsson', 'bjorg+1954-10-09+brynjolfsdottir', 'bjorg+1967-11-09+e
         rlingsdottir', 'bjorgvin+1955-11-03+eyjolfsson', 'bjork+1965-11-17+gudbjartsdottir', 'bjork+1962-11-14+vilhelmsdottir', 'blaed
         is+1962-11-09+gudjonsdottir', 'bogi+1937-11-24+sigurbjornsson', 'bragi+1931-04-26+eggertsson', 'brimrun+1980-07-04+bjorgolfsdo
         ttir', 'britta+1972-12-20+skov', 'bryndis+1978-09-10+bjornsdottir', 'bryndis+1976-08-13+fridriksdottir', 'brynhildur+1962-06-2
```

Figure 5: *Key values of people who are still considered as duplications*

$$duplicate\_dict = 492$$
$$dict\_removed\_single\_entries = 290$$
$$dict\_name\_entries = 528$$
$$len(dict\_einstaklingar\_teammember\_info) = 516$$
$$len(not\_the\_same\_person.keys()) = 25$$
$$len(most\_likely\_same\_person) = 482$$
$$len(merged\_list) = 482$$

V. Conclusion

We did not manage to implement everything we wanted, we would have liked to have implemented a way to further check if the entries in $find_duplicates()$ that were determined to not be the same person, were correctly identified or not.

We also did some cleaning in the teams (lids) table and in future works we would have liked to clean up the names more and merged all the entries of teams that were duplicated entries, from there on, we would then be able to change all the lidIDs in team members (lidsmenn) table to the correct teamID and thus we could be even more certain that our $find_duplicates()$ function was identifying duplicates correctly, since our implementation can not check if two teams are in fact the same team just wrongly inserted.

One of our problems in our duplication keys identifiers is that if there is a missing middle name or last name for a person then it might created 5 different keys for the same person (firstname->lastname, firstname->middlename->lastname, firstname->middlename, firstname) based on the same birthday, so if the team table were to be correctly cleaned up then we could connect it to the team members table which we could then iterate over every entry and connect all names together (based on that maybe two entries with same first name and birthday played at the same time, but then one of those entries also played another day with a person with the same birthday and first name, that way we could create a spider web of connections based on days they played for a team, but this is impossible if the team member table links to incorrectly inserted names in the table teams).

References

[1] Jules J. Berman. "Principles and Practice of Big Data - Preparing, Sharing, and Analyzing Complex Information - Second Edition". In: *no journal* (2018), p. 78.

[2] Jules J. Berman. "Principles and Practice of Big Data - Preparing, Sharing, and Analyzing Complex Information - Second Edition". In: *no journal* (2018), pp. 24–25.

[3] Jules J. Berman. "Principles and Practice of Big Data - Preparing, Sharing, and Analyzing Complex Information - Second Edition". In: *no journal* (2018), p. 69.