

TERM PROJECT DOCUMENTATION | ANALYSIS OF SUICIDE RATES

Data Engineering 2

MS in Business Analytics

TEAM

Julianna Szabo - 2003870

Maeva Braeckevelt - 2003872

Dominik Gulácsy – 2003374

Global Data Flow Outline

Data Sources:

- Kaggle
- Rest Countries to search for alpha-2 codes of countries
- World Bank API

Economic indicators considered:

- Unemployment Rate | SL.UEM.TOTL.MA.ZS and SL.UEM.TOTL.FE.ZS
- Population | SP.POP.TOTL
- Alcohol cons | SH.ALC.PCAP.FE.LI and SH.ALC.PCAP.MA.LI
- Smoker | SH.PR.V.SMOK.MA and SH.PR.V.SMOK.FE

Analytical Questions:

- What age group tends to have higher suicide rates?
- Which countries tend to have higher suicides?
- How can it be related to socio-economic indicators like Unemployment Rate, GDPPC, alcohol consumption, and cigarette consumption?

Statistical techniques:

- Histogram or boxplot
- Bar charts – percentage of total <- stacked bar charts
- Scatterplots

Contribution:

- Julianna: Visualization + cleaning + documentation + presentation
- Maeva: SQL + help with visualization + documentation + presentation
- Dominik: Data infrastructure + API + ETL + documentation

I. Data

The dataset we chose to work with is a dataset that collected data on suicide. We found it on Kaggle. We decided to work with this dataset mainly because it had a lot of data and a large variety. Countries, years, sex, ages, HDI, generation, suicide per 100k inhabitants, and GDP per capita were the variable available to help us for a robust analysis (details information on the variable is on the variable.xlsx file on Github). However, to better understand the suicide rate, we thought that other variables could be interesting. That's why we chose among the world bank data website: alcohol consumption, cigarette consumption, and unemployment rate.

All the engineering work we have done were aim to give us a visualization to answer these questions:

- What age group tends to have higher suicide rates?
- Which countries tend to have higher suicides?
- How can it be related to socio-economic indicators like Unemployment Rate, GDPPC, alcohol consumption, and cigarette consumption?

For alcohol and smoking habits, we decided to separate the male from the female. We made this choice because their consumption is usually not equal. Further, we chose to add the population from WDI for data quality purposes. The source of the population data was not mentioned on Kaggle, so

country	year	sex	age	suicides_r	populatio	suicides_	country_year	HDI	gdp	gdppc	generatio
Albania	1987	male	15-24 year	21	312900	6.71	Albania1987		2,156,624,900	796	Generatio
Albania	1987	male	35-54 year	16	308000	5.19	Albania1987		2,156,624,900	796	Silent
Albania	1987	female	15-24 year	14	289700	4.83	Albania1987		2,156,624,900	796	Generatio
Albania	1987	male	75+ years	1	21800	4.59	Albania1987		2,156,624,900	796	G.I. Gene
Albania	1987	male	25-34 year	9	274300	3.28	Albania1987		2,156,624,900	796	Boomers
Albania	1987	female	75+ years	1	35600	2.81	Albania1987		2,156,624,900	796	G.I. Gene

we opted for a universal and reliable source such as the World Bank.

Data citations:

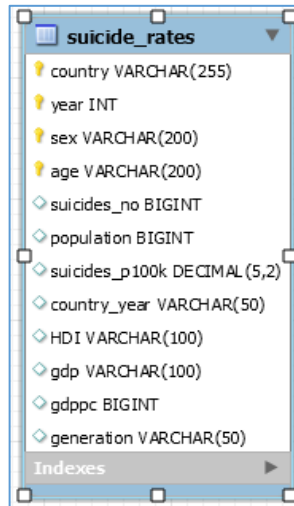
Suicide Rates Overview 1985 to 2016, Compares socio-economic info with suicide rates by year and country [Internet], Kaggle: Your Machine Learning and Data Science Community, 2018, [download on the 11/19/2020], License: World Bank Dataset Terms of Use, <https://www.kaggle.com/russellyates88/suicide-rates-overview-1985-to-2016/metadata>

World Bank, "Total population", "Unemployment rate male and female", "Alcohol consumption (male and female)", "Smoking prevalence (male and female)", World Development Indicators. The World Bank Group, [download on the 12/03/2020,] <https://data.worldbank.org/indicator?tab=all>

II. Data Resource Infrastructure

a. Kaggle Suicide Rates Dataset

First, we downloaded the csv file (data/static_kaggle/suicide_rate.csv) from [Kaggle](#), representing the core data in our workflow. We wrote an SQL file (data/static_kaggle/import_suicide_rates.sql) in which we imported the csv data file into a local MySQL database to check if it loads properly and explore the data.

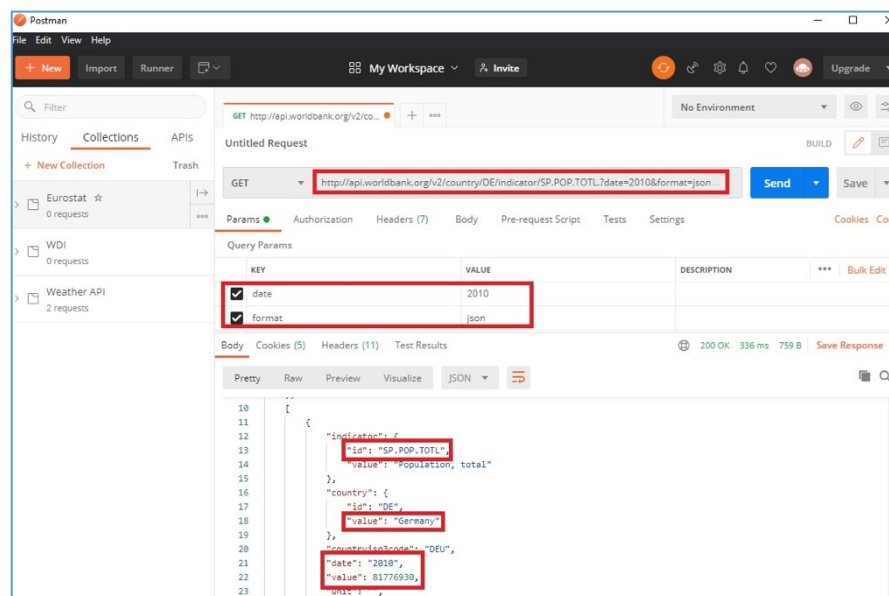


suicide_rates	
country	VARCHAR(255)
year	INT
sex	VARCHAR(200)
age	VARCHAR(200)
suicides_no	BIGINT
population	BIGINT
suicides_p100k	DECIMAL(5,2)
country_year	VARCHAR(50)
HDI	VARCHAR(100)
gdp	VARCHAR(100)
gdppc	BIGINT
generation	VARCHAR(50)
Indexes	

We wanted to make our workflow as reproducible and dynamic as possible, so we decided to host a database. We created a MySQL database instance using the RDS service of AWS. We used the AWS account provided by CEU. Our julmaedom-mysql-de2 named DB instance was configured as a db.t2.micro instance with 20 GiB storage. We set up a connection to the instance in MySQL Workbench. We exported our locally hosted DB data and structure in a Self-Contained sql file (data/aws/dump_suicide_rates.sql). Then we imported the data using this dump to our MySQL on AWS. We added a new user named grader that has only read access to simulate a real-life setup. Credentials for this user and connectivity details are provided in a text file (data/aws/grader_credentials.txt).

b. REST Countries API

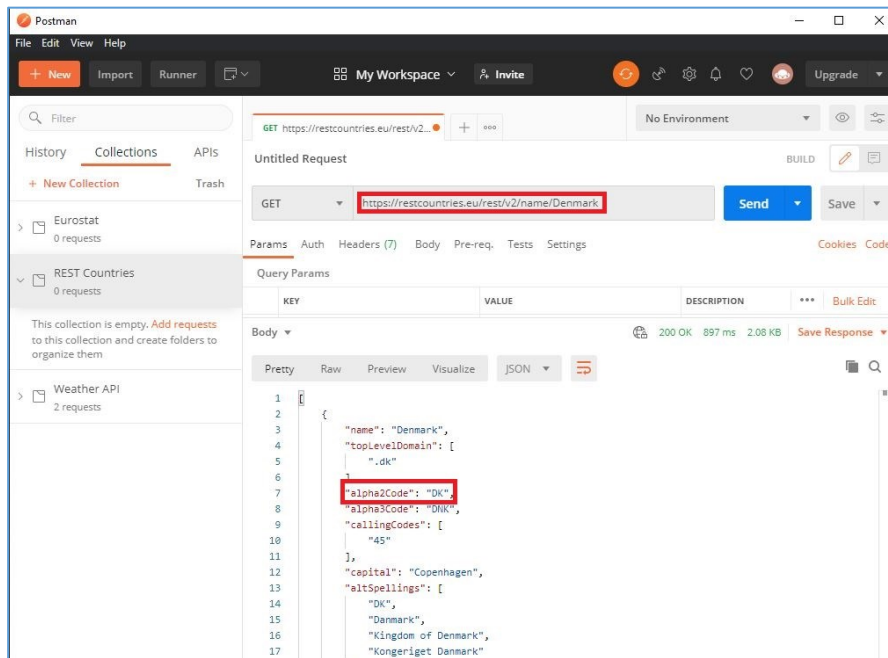
To download more data for the countries in the suicide_rates.csv file and to be able to join the subsequent data tables properly, we needed to identify countries universally. We achieved this by using country codes, more specifically alpha2 country codes. To associate every country in our data with a code, we used the REST Countries API to search for it. This functionality is called “name” and



available through <https://restcountries.eu/rest/v2/name/{name}>. This API does not require an access key, so it was easy to get it working. See below an example for Denmark:

c. World Bank API & WDIs_of_interest file

Since our analytical goal was to investigate suicide rates based on some other variables that we considered meaningful and relevant, we wanted to get more data on the countries. To do this, we looked at the world development indicators offered by the World Bank. We managed to access these



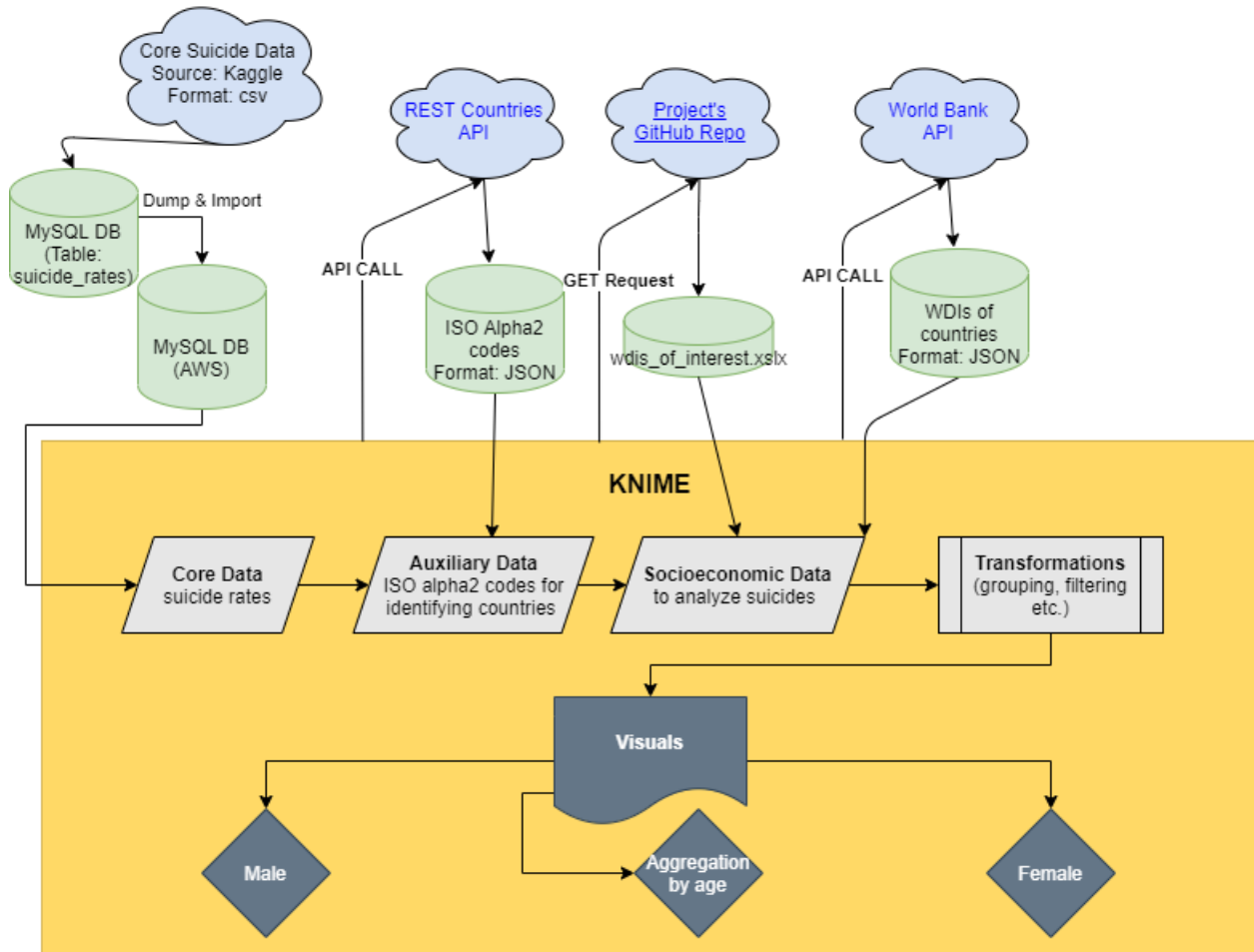
indicators via the [World Bank's public API service](#). We picked indicators using this overview [list](#) by applying posteriori logic during the selection process. See an example for Germany's population data in 2010:

We wanted to keep record of our choices of indicators, so instead of putting them as plain text somewhere in the KNIME workflow, we created an [Excel file](#) and collected the indicators' names and WDI reference codes, e.g.: SP.POP.TOTL for total population. This step cleared up our input data handling process and made it easy to later experiment with other variables as well. We opted for an Excel file since it's more user-friendly, and the content of the file is just for administrative purposes. It functions much more like a UI rather than actual data.

III. ETL Pipeline

We carried out the loading, cleaning, and analysis of our data in KNIME. We aimed to create a dynamic, well-integrated, and reliable workflow file that had a robust structure. The first step toward this was to take all input data sources and turn to hosted solutions. By hosting our MySQL DB on an AWS instance, we mainly fulfilled this motive. However, we also took advantage of making this project

available in the form of a Github repository online by pointing KNIME nodes in our workflow to the link of the [WDIs of interest Excel file](#) in the repo. Now let's see how we dealt with data loading and preprocessing in KNIME.



First, we created a MySQL connector node to connect to our remote MySQL DB that contained the suicide_rates data. Then we added two Query nodes: one for retrieving the data itself and one to use the REST Countries API on. The first problem we faced was that the REST Countries API could not recognize Saint Vincent and Grenadines because it was stored as Saint Vincent and the Grenadines. We addressed this issue in the string manipulation node, where we constructed the necessary URLs to call the API service. We ran the API on the countries extracted on the country codes and joined it with the original data based on the country column. Later on, we realized that the year column in the original data was stored as a number, which caused problems during the construction of URLs at World Bank's API, so we converted it to a string.

The next phase in the ETL process was to read in the World Bank's code of the WDIs from the WDIs_of_interest file. We had quite some WDIs in the file. This is a problem because it would have been quite tedious to do a branch of nodes for each WDI to get the World Bank API data. This solution would not have been scalable and dynamic. So we decided to attach the Excel reader node to a Table Row To Variable Loop Start. We ended the loop with a Loop End (Column append) node to iterate through the WDIs and add the results of the API calls to the output table. In the loop body, we created

an empty table to which we attached the WDIs as a flow variable. Then we constructed the URL for the World Bank API using a string manipulation node. We built up the structure of the URL the following way:

```
join("http://api.worldbank.org/v2/country/all/indicator/",strip($${SWDI_id}$),"?date=",string($${lmin_year}$),"-",string($${lmax_year}$),"&per_page=12000","&format=json")
```

As can be seen, we specified the specific WDI, the date range, the number of results per page, and the response format. Like the series of nodes dealing with REST Countries API, we sent a get request, extracted the JSON's relevant values, and dropped unnecessary columns. However, in this case, we had only one row but a table of data in our JSON to be extracted. We used the list option in the JSON path node to overcome this. We got this way the list of all countries, years, and WDI values in 3 distinct columns. We transformed the shape of the data by applying an ungroup node, which turned the data into a nice tabular structure. After some renaming, we set up a conditional node cluster in which we investigated whether the WDI is the first in the loop. We needed this not to have duplicated columns of countries and years when appending the resulting columns to the output table. Using the conditional block, we only append these columns for the first time, and later we only add the given WDI column. So when the loop ends, we have the country, date, and WDI columns. Finally, we joined this data table with the table from the first join. At this point, we had all the data we needed. Actually, even more.

This workflow setup is very efficient because by taking advantage of the loop, we had to send the least amount of get requests as possible as those can significantly increase runtime. Furthermore, a very high number of requests may result in blocking the IP on which the workflow runs (with the reason of violating fair usage policy or identifying the process as a DDOS attack). To further increase the efficiency dynamically, we specified the date range in the World Bank API URL based on the years in the core suicide_rates data. We accomplished this by taking the last and earliest years and putting them into variables, merging them, and attaching them to the string manipulation node in the loop.

Once we had a clean data table, we needed to clean it further for visualization. First, we filtered out all the missing values from any of the columns. Once that was done, we realized that most observations were for the year 2010. While we originally wanted to do a time series analysis, at this point, we realized we would not have the data to do so. Therefore, we decided to switch to a cross-sectional analysis using data from only the year 2010, after we confirmed the significant difference in the number of observations for that year using a histogram.

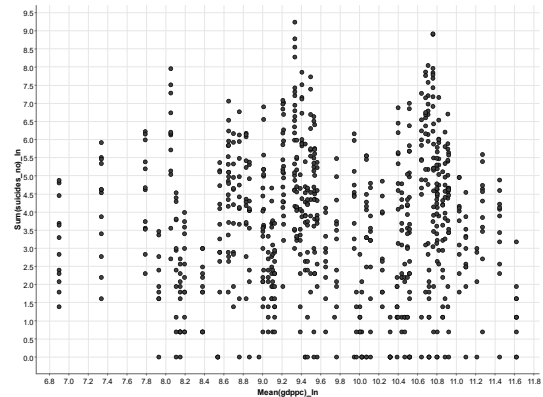
IV. Visualization

Once we got the data joined and cleaned, and could start the visualization. The main challenge here was the different variables that needed to be visualized to give us a good overview of the factors that may affect suicide rates. There were two main groups of variables. The first one was about the whole population, such as GDP per Capita, while others, such as smoking habits, were differentiated between male and female. Therefore, it required two different paths in KNIME.

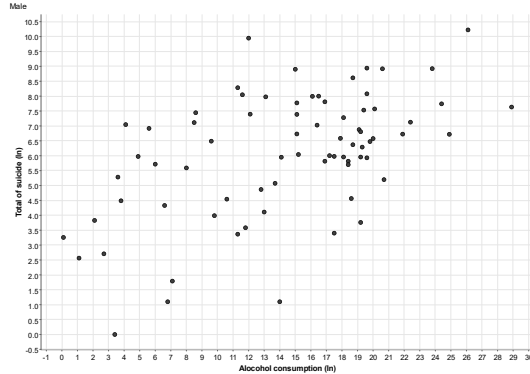
The aggregation process was relatively simple for the ones containing the whole population since it only needed to be grouped by one variable. We used the sum function for the number of suicides (since this is a flow variable) and the mean for the GDP per Capita and the population.

For the population, we created a simple scatter plot at first to see if there was any linear correlation. The scatter plot shows that there is a general positive correlation between population and the number of suicides. The smaller the population, the fewer suicides; however, larger populations do not necessarily have higher suicide rates.

We first created a scatter plot using level-level analysis for GDP per Capita but decided that a log-log transformation would best represent this data. It shows how a one percent change in GDP per Capita could affect the percentage number of suicides. It shows that there seems to be no correlation between percentage change in GDP per Capita does not affect the percentage change of the number of suicides.



Total of suicide per alcohol consumption (ln scale)



We needed to take a couple of extra steps for the variables that have a division by sex. We completed the same steps expect that we needed to aggregate including sex and then created two separate paths based on it.

For the unemployment and the smoking rates, both sexes showed no huge positive correlation to suicide rates. There was a tendency towards the center, meaning that people with an average smoking habit were most likely to commit suicide, but no clear correlation.

For alcohol consumption, we decided to do a log-log analysis. This showed significantly more interesting results. There is a cluster around the higher percentage of alcohol consumption and the percentage of suicides (graph shows males). Therefore, there is a correlation between the percentage change in alcohol consumption and the percentage change in the number of suicides. We cannot claim causality, but there seems to be a connection between these two variables, much more than any others.