01_make_clean_dataset

June 9, 2021

In this notebook we will import the raw data from the WGM, clean them and dummy-code them in order to make them compatible with the future analysis we will run.

[ ]:

```
```

1 Import the packages

[3]:

```
import pandas as pd
import numpy as np
import scipy.stats as stt
import networkx as nx
import matplotlib.pyplot as plt
```

[ ]:

```
```

2 Import the raw data

Main data:

`wgm_raw` the full database

`wgm_dic` a dictionary of what the database means -> the important columns are the `code`, `long question` and `short question`

Note: wgm_dic is not a dictionary data type, but a dataframe. This has been done as we need to convert between 3 different types of dataframe we will deal with:

- boolean (i.e. dummy coded)
- labels (i.e. very entry is
- numeric

The file `wgm2018.xlsx` is the raw file provided by the Wellcome Global Monitor: https://wellcome.org/reports/wellcome-global-monitor/2018

Instead, the `wgm2018_data_dic_mod.xlsx` is a file made by us to rename the questions and the answers in a more compact way for when dummy coding. You can find it here: https://github.com/just-a-normal-dino/wgm18_dic

```
[4]: # Import the raw data
     wgm_raw = pd.read_excel('wgm2018.xlsx', sheet_name=1)
     wgm_dic = pd.read_excel('wgm2018_data_dic_mod.xlsx')
     wgm = wgm_raw.copy()
```

Display the raw data

```
[5]: # wgm_raw.info()
     wgm.head()
```

```
[5]:    WP5       wgt         PROJWT FIELD_DATE  YEAR_CALENDAR  Q1  Q2  Q3  Q4  \
     0    1  0.652821  171769.597742 2018-01-08           2018   3   2   1   2
     1    1  0.695706  183053.484155 2018-01-08           2018   2   2   1   2
     2    1  0.523829  137829.328857 2018-01-08           2018   2   2   1  98
     3    1  0.764442  201139.215039 2018-01-08           2018   2   1   1   2
     4    1  3.327946  875645.512738 2018-01-08           2018   2   1   1   2

        Q5A  ... Age  AgeCategories  Gender  Education  Urban_Rural  \
     0    2  ...  72              3       2          3            1
     1    1  ...  72              3       1          2            2
     2    1  ...  85              3       1          2            1
     3    1  ...  54              3       1          3            2
     4    1  ...  20              1       1          2            2

        Household_Income  Regions_Report  Subjective_Income  WBI  EMP_2010
     0                 3               7                  2    4         6
     1                 3               7                  1    4         6
     2                 2               7                  3    4         6
     3                 5               7                  1    4         1
     4                 2               7                  1    4         6

     [5 rows x 60 columns]
```

Display the dictionary

```
[7]: wgm_dic.head()
```

```
[7]:             Code                                      Long question  \
     0            WP5                                            Country
     1            wgt  National weight, for analysis at the country l...
     2         PROJWT  Population weight (included factor to project ...
     3     FIELD_DATE                              Study Completion Date
     4  YEAR_CALENDAR                                    Year of survey

         Short question  Trust in science value  \
     0          Country                       0
     1       Nat weight                       0
     2       Pop weight                       0
```

```
3  Completion Date                      0
4      Survey Year                      0

                                        Ans dic  \
0  1=United States, 2=Egypt, 3=Morocco, 4=Lebanon...
1                           Scale (value of weight)
2                           Scale (value of weight)
3                                              Date
4                                              Year

                                              Notes
0                                               NaN
1  Use this weight for analysis at the country level
2  Use this weight for analysis which pools toget...
3                                               NaN
4                                               NaN
```

[ ]:

```

```

[ ]:

```

```

3 Clean the dictionary

Drop the notes column

[8]:

```
# Note: if you'll run this cell twice, you'll get an error as it cannot delete␣
 ↪it twice
wgm_dic.drop(columns="Notes", inplace=True)
wgm_dic.head()
```

[8]:

```
            Code                                      Long question  \
0            WP5                                            Country
1            wgt  National weight, for analysis at the country l...
2         PROJWT  Population weight (included factor to project ...
3     FIELD_DATE                              Study Completion Date
4  YEAR_CALENDAR                                     Year of survey

    Short question  Trust in science value  \
0          Country                      0
1       Nat weight                      0
2       Pop weight                      0
3  Completion Date                      0
4      Survey Year                      0

                                        Ans dic
```

```
0  1=United States, 2=Egypt, 3=Morocco, 4=Lebanon...
1                            Scale (value of weight)
2                            Scale (value of weight)
3                                               Date
4                                               Year
```

Make the code column as the index of the dictionary (and duplicate it so I can easily access it as a column)

[9]:

```
wgm_dic["Code_i"] = wgm_dic["Code"]
wgm_dic.set_index("Code_i",inplace=True)
wgm_dic.head()
```

[9]:

```
                        Code  \
Code_i
WP5                      WP5
wgt                      wgt
PROJWT                PROJWT
FIELD_DATE        FIELD_DATE
YEAR_CALENDAR  YEAR_CALENDAR

                                                   Long question  \
Code_i
WP5                                                      Country
wgt            National weight, for analysis at the country l...
PROJWT         Population weight (included factor to project ...
FIELD_DATE                                 Study Completion Date
YEAR_CALENDAR                                    Year of survey

                Short question  Trust in science value  \
Code_i
WP5                    Country                       0
wgt                 Nat weight                       0
PROJWT              Pop weight                       0
FIELD_DATE     Completion Date                       0
YEAR_CALENDAR      Survey Year                       0

                                                         Ans dic
Code_i
WP5            1=United States, 2=Egypt, 3=Morocco, 4=Lebanon...
wgt                                      Scale (value of weight)
PROJWT                                   Scale (value of weight)
FIELD_DATE                                                  Date
YEAR_CALENDAR                                               Year
```

Add a new columns which tells you if the value is a cathegory or not (`Categorical Ans`). This would be true if the answers are categorical (aka "nominal"). And it would be false for continuous numeric variables such as age.

[11]:

```
ans_col = wgm_dic["Ans dic"]
is_category = ans_col.apply(lambda el : "=" in el) # Almost all categorical␣
 ↪variables have a dictionary in the form of "ans x = y"
wgm_dic["Categorical Ans"] = is_category
wgm_dic.loc[["Age"],["Categorical Ans"]] = False # Manually removing Age

wgm_dic.head()
# print(wgm_dic.loc[is_category, ["Ans dic"]])
# print(wgm_dic.loc[wgm_dic["Categorical Ans"] == False, ["Ans dic"]])
```

[11]:

```
                        Code  \
Code_i
WP5                      WP5
wgt                      wgt
PROJWT                PROJWT
FIELD_DATE        FIELD_DATE
YEAR_CALENDAR  YEAR_CALENDAR

                                                   Long question  \
Code_i
WP5                                                      Country
wgt            National weight, for analysis at the country l...
PROJWT         Population weight (included factor to project ...
FIELD_DATE                                 Study Completion Date
YEAR_CALENDAR                                    Year of survey

                Short question  Trust in science value  \
Code_i
WP5                    Country                      0
wgt                 Nat weight                      0
PROJWT              Pop weight                      0
FIELD_DATE     Completion Date                      0
YEAR_CALENDAR      Survey Year                      0

                                                         Ans dic  \
Code_i
WP5            1=United States, 2=Egypt, 3=Morocco, 4=Lebanon...
wgt                                      Scale (value of weight)
PROJWT                                   Scale (value of weight)
FIELD_DATE                                                  Date
YEAR_CALENDAR                                               Year

               Categorical Ans
Code_i
WP5                       True
wgt                      False
PROJWT                   False
```

```
FIELD_DATE                    False
YEAR_CALENDAR                 False
```

```
[ ]:
```

4 Define functions acting on the dictionary

As we will have three different dataframes in three different format (boolean, numeric and labels) here we define several functions to “translate” questions or answers from one dataframe to the others

```
[ ]:
```

Check if an element is in the series

```
[17]: def is_in(series,element):
          #Checks if the element is in the series. If so, it also returns the index of␣
       ↪where it is found
          try:
              ind = series[series == element].index[0]
              out = [True, ind]
          except:
              out = [False, None]
          return out
```

```
[ ]:
```

4.0.1 Translate questions

Find the index of a question (in format string) from the dictionary (`wgm_dic`)

```
[18]: def find_question_index(questions, in_format="Auto", out_format="Short"):
          # the question should be a string

          codes = wgm_dic["Code"]
          long = wgm_dic["Long question"]
          short = wgm_dic["Short question"]

          if type(questions) == type('abc'): # if it's a string

              isincode = is_in(codes,questions)
              isinlong = is_in(long,questions)
              isinshort = is_in(short,questions)

              if in_format == "Auto":
                  if isincode[0]: # if it's a code
                      ind = isincode[1]
```

```
            elif isinlong[0]: # if it's a long
                ind = isinlong[1]
            elif isinshort[0]: # if it's a short
                ind = isinshort[1]
            else:
                raise Exception("Question not found in any type!")

        elif in_format == "Code":
            if isincode[0]: # if it's a code
                ind = isincode[1]
            else:
                raise Exception("Question not found in the specified type!")

        elif in_format=="Short":
            if isinshort[0]: # if it's a code
                ind = isinshort[1]
            else:
                raise Exception("Question not found in the specified type!")

        elif in_format=="Long":
            if isinlong[0]: # if it's a code
                ind = isinlong[1]
            else:
                raise Exception("Question not found in the specified type!")

        else:
            raise Exception("Input data type not recognized")
    else:
        raise Exception("Invalid question type")

    return ind
```

[ ]:

```
```

Translate the questions (either a string or a list of strings) into any other format (short, long or code)

[19]:

```
def tanslateQuest(questions, in_format="Auto", out_format="Short"):
    # Translates a question from a format to another (Only Short, Long or Code)

    # questions should be either a list of strings or a string
    # The format can be only Long, Short or Code

    codes = wgm_dic["Code"]
    long = wgm_dic["Long question"]
    short = wgm_dic["Short question"]

    if type(questions) == type('abc'): # if it's a string
        questions = [questions] # make it as list

    ind_vec = list()
    out_vec = list()

    for quest in questions:
        ind = find_question_index(quest, in_format="Auto", out_format="Short")
        ind_vec.append(ind)

        if out_format == "Code":
            out = codes[ind]
            out_vec.append(out)

        elif out_format == "Short":
            out = short[ind]
            out_vec.append(out)

        elif out_format == "Long":
            out = long[ind]
            out_vec.append(out)

        else:
            raise Exception("Output format not recognized!")

    return [out_vec, ind_vec]
```

[ ]:

```
```

4.0.2 Tranlsate answers

You enter a question and it gives out the possible answers as dictionary type. Actually the real output is:

`[numNval_dict, num2val, val2num]`

where `numNval_dict` is the dictionary in both diretions (both num2val and val2num)

[20]:

```
def extractAns(question, question_in_format="Auto", question_out_format="Short",␣
 ↪ans_out_format="AShort"):
    # you can use only one question
    # Answers can be a list

    quest_index = tanslateQuest(question, in_format=question_in_format,␣
 ↪out_format="Code")[0][0]

    raw_dict = wgm_dic.loc[[quest_index], ["Ans dic"]]
    raw_dict = raw_dict.values[0][0]

    splitted = raw_dict.split(sep=', ')
#     print(splitted)

    num2val = dict()
    val2num = dict()
    numNval_dict = dict()

    for el in splitted:
        if len(el)<3:
            continue

#         print(el)
        [num, val] =el.split(sep='=')
        num = int(num)

        num2val[num] = val
        numNval_dict[num] = val

        val2num[val] = num
        numNval_dict[val] = num

    return [numNval_dict, num2val, val2num]
```

[ ]:

```

```

Translate your answes from one format to the other (you need to specify the question, of course)

[21]:

```
def translateAns(question, answers, question_in_format="Auto",␣
 ↪question_out_format="Short", ans_out_format="Auto"):
    # you can use only one question
    # Answers can be a list
    # At the moment ans_out_format can be only Auto

    trans_Ans_dict = extractAns(question, question_in_format="Auto",␣
 ↪question_out_format="Short", ans_out_format="AShort")[0]

    if not type(answers)==type(list()): # Turn the answers in a list, so we can␣
 ↪iterate
        answers = [answers]

    translated_ans = list()
    for ans in answers:
        strans_ans = trans_Ans_dict[ans]
        translated_ans.append(strans_ans)
    if len(translated_ans) == 1:
        translated_ans = translated_ans[0]

    return translated_ans


#     quest_index = tanslateQuest(question, in_format=question_in_format,␣
 ↪out_format="Code")[0]

#     raw_dict = wgm_dic.loc[[quest_index], ["Ans dic"]]

    return raw_dict
```

[ ]:

```

```

5 Clean the labels in the database

Make a dictionary of all the indeces -> `index_dic`

[24]:

```
# need to create a dictionary
# wgm_dic["Code"]

list_of_codes = list(wgm_dic["Code"])

index_dic = dict()

for code in list_of_codes:
    short_vers = wgm_dic.loc[[code],["Short question"]].values[0][0]

    index_dic[code] = short_vers

# index_dic
```

[ ]:

```

```

5.0.1 Make the numeric version of the database

i.e. columns names (questions) are in version short, while all the answers are numeric

-> This dataframe will be called `wgm_numeric`

[27]:

```
wgm_numeric = wgm.rename(columns=index_dic)
wgm_numeric.head()
```

[27]:

```
   Country  Nat weight     Pop weight Completion Date  Survey Year  \
0        1    0.652821  171769.597742      2018-01-08         2018
1        1    0.695706  183053.484155      2018-01-08         2018
```

```
2        1    0.523829  137829.328857      2018-01-08         2018
3        1    0.764442  201139.215039      2018-01-08         2018
4        1    3.327946  875645.512738      2018-01-08         2018

   Know Science  Understand meaning Sci  Study disease is science  \
0             3                       2                         1
1             2                       2                         1
2             2                       2                         1
3             2                       1                         1
4             2                       1                         1

   Poetry is science  Learned Sci in Prim.School  ... Age Pers Age Coho  \
0                  2                           2  ...       72        3
1                  2                           1  ...       72        3
2                 98                           1  ...       85        3
3                  2                           1  ...       54        3
4                  2                           1  ...       20        1

   Gender  Education  Area Type  Income  Region  Subjective Income  \
0       2          3          1       3       7                  2
1       1          2          2       3       7                  1
2       1          2          1       2       7                  3
3       1          3          2       5       7                  1
4       1          2          2       2       7                  1

   Income Level  Employment
0             4           6
1             4           6
2             4           6
3             4           1
4             4           6

[5 rows x 60 columns]
```

[ ]:

```

```

5.0.2 Make the version with labels of the database

i.e. questions/columns as short and answers as val (not numeric)

-> wgm_labels

Note: some values are still numeric (such as the age) as it doesn't make any sense to change it. However, all the categorical questions will be changed

[28]:

```
wgm_labels = pd.DataFrame() # empty df

list_of_questions = list(wgm_dic["Short question"])
```

```
list_of_catheg_questions = list()

def translate_column(var):
    try:
        out = ans_dic[var]
    except:
        out = "Empty"
    return out

for quest in list_of_questions:

    if not wgm_dic.loc[wgm_dic["Short question"] == quest, ["Categorical Ans"]].
 ↪values[0][0]:
        # if it's not a cathegorical variable
        # Just copy it the way it is
        wgm_labels[quest] = wgm_numeric[quest]
    else:
        list_of_catheg_questions.append(quest)

        entire_col = wgm_numeric[quest]

        ans_dic = extractAns(quest)[0]

        entire_col_text = entire_col.apply(translate_column)

        wgm_labels[quest] = entire_col_text
```

[29]:
```
wgm_labels.head()
```

[29]:
```
         Country  Nat weight     Pop weight Completion Date  Survey Year  \
0  United States    0.652821  171769.597742      2018-01-08         2018
1  United States    0.695706  183053.484155      2018-01-08         2018
2  United States    0.523829  137829.328857      2018-01-08         2018
3  United States    0.764442  201139.215039      2018-01-08         2018
4  United States    3.327946  875645.512738      2018-01-08         2018

  Know Science Understand meaning Sci Study disease is science  \
0     Not much             Some of it                      Yes
1         Some             Some of it                      Yes
2         Some             Some of it                      Yes
3         Some              All of it                      Yes
4         Some              All of it                      Yes

  Poetry is science Learned Sci in Prim.School  ... Age Pers  Age Coho  \
0                No                         No  ...       72       50+
1                No                        Yes  ...       72       50+
2               (DK)                        Yes  ...       85       50+
3                 No                        Yes  ...       54       50+
4                 No                        Yes  ...       20  15 to 29

   Gender  Education                          Area Type      Income  \
0  Female   Tertiary  Lives in rural area or small town  Middle 20%
1    Male  Secondary    Lives in city or suburb of city  Middle 20%
2    Male  Secondary  Lives in rural area or small town  Second 20%
3    Male   Tertiary    Lives in city or suburb of city     Top 20%
4    Male  Secondary    Lives in city or suburb of city  Second 20%

             Region Income Level  \
0  Northern America  High income
1  Northern America  High income
2  Northern America  High income
3  Northern America  High income
4  Northern America  High income

                                    Subjective Income  \
0                        Getting by on present income
1             Living comfortably by on present income
2   Finding it difficult/very difficult to get by...
3             Living comfortably by on present income
4             Living comfortably by on present income

                           Employment
0                    Out of workforce
1                    Out of workforce
2                    Out of workforce
3  Employed full time for an employer
4                    Out of workforce

[5 rows x 60 columns]
```

[ ]:

```
```

6 Boolean version of the database (aka dummy coded)

i.e. each columns represents one combination of question and answers (e.g. “Vaccines:Trust”) the valus in the cells are then just booleans. This is useful for performing dichotomous analysis

[31]:

```
wgm_bool = pd.DataFrame()

list_of_attitudes = list()

for quest in list_of_catheg_questions: # for each question
```

```
    num2val_dic = extractAns(quest)[1]

    for key in num2val_dic: # for each anwer
        val = num2val_dic[key]
        full_str = quest+":"+val

        list_of_attitudes.append(full_str)

        col = wgm_labels[quest] == val

        wgm_bool[full_str] = col

# num2val_dic
# full_str
```

[ ]:

```

```

7 End of cleaning

Save the files to your favourite format

[33]:

```
# Excel
# filename = "wgm2018_cleaned"
# Excel_writer = pd.ExcelWriter(filename+".xlsx", engine = 'xlsxwriter')
# wgm_dic.to_excel(Excel_writer, sheet_name='Dictionary')
# wgm_numeric.to_excel(Excel_writer, sheet_name='Numeric')
# wgm_labels.to_excel(Excel_writer, sheet_name='Labels')
# wgm_bool.to_excel(Excel_writer, sheet_name='Booleans')

# Pickle
basename = "wgm2018_clean_"
wgm_dic.to_pickle(basename+"dictionary"+".pkl")
wgm_numeric.to_pickle(basename+"numeric"+".pkl")
wgm_labels.to_pickle(basename+"labels"+".pkl")
wgm_bool.to_pickle(basename+"boolean"+".pkl")

# Read
# print(pd.read_pickle(basename+"boolean"+".pkl")
```

The files are now ready to be used in the following codes