INF1340 Midterm UN Data Cleansing Project

Yanke Mo

- Introduction:

The aim of this tidy data report is to clean the data of "United Nation: Trends in International Migrant Stock" based on five tidy data principles. The expectation of the cleaning process is to remove errors and anything that violated the tidy data principles, so that the final data can be applied for research or calculation directly.

The report will first analyze the content of the raw data. Based on the content of the raw data, a draft of the expectation will be carried out, and the raw data will be reorganized until it matches the expected result. The report will also explain the problems that appear in the data cleaning process, with the solutions of solving it.

- Raw Data and Expectations:

The raw data of "United Nation: Trends in International Migrant Stock" is stored in an excel file. It analyzes the migrant stock for all countries and regions based on genders and years. There are six tables in the files which contain data about migrant stock, each of them looking into migrant stock on a different scale. Every table is built in a highly similar structure, the migrant stock of every country and region are categorized first by gender, then by every five years starting from 1990 to 2015.

The data stored in excel file is well-organized, but problems appear when the data is imported in jupyter. The data that is imported in Jupyter violated the the five principles of tidy data, including having values as column's name, observation repeat, and cell contains more than one value. To produce a clean data, the raw data is expected to organized in ways that:

1. Every column is a unique variable.
2. Every row is a unique observation.
3. Each value should be stored in each cell.
4. Observation is not repeated in one table.
5. Observation has not been repeated in multiple tables.

In the meantime, the final data should be opened by each gender from 1990 to 2015, rather than opening data from year to gender. This process required splitting data into multiple tables in Jupyter, and the steps of approaching these expectations will be shown and explained in the following section.

- <u>Data Cleaning</u>:

➢ **<u>Step One: Remove Cover Page Error</u>**

The data cleaning process is started when tables are opened by pd.read.excel. Every table in the raw data is import as “Table_#”(# means the order of the table) into Jupyter. The first problem appears in the first 12 rows of every table. The first 12 rows of the original excel table is the cover page, and the cover page is imported as NaN into Jupyter.

At the same time, the column name of the tables has been imported as values (row 13 and row 14). This is a violation of tidy data principles #1, because the column names should not be stored in the cell, and column names should only be in one row. (pic1)

index	Unnamed: 0	Unnamed: 1	Unnamed: 2	Unnamed: 3	Unnamed: 4	Unnamed: 5	Unnamed: 6	Unnamed: 7	Unnamed: 8	Unnamed: 9
0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	NaN	NaN	NaN	NaN	United Nations	NaN	NaN	NaN	NaN	NaN
4	NaN	NaN	NaN	NaN	Population Division	NaN	NaN	NaN	NaN	NaN
5	NaN	NaN	NaN	NaN	Department of Economic and Social Affairs	NaN	NaN	NaN	NaN	NaN
6	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
7	NaN	NaN	NaN	NaN	Trends in International Migrant Stock: The 2015 Revision	NaN	NaN	NaN	NaN	NaN
8	NaN	NaN	NaN	NaN	Table 1 - International migrant stock at mid-year by sex and by major area, region, country or area, 1990-2015	NaN	NaN	NaN	NaN	NaN
9	NaN	NaN	NaN	NaN	POP/DB/MIG/Stock/Rev.2015	NaN	NaN	NaN	NaN	NaN
10	NaN	NaN	NaN	NaN	December 2015 - Copyright © 2015 by United Nations. All rights reserved	NaN	NaN	NaN	NaN	NaN
11	NaN	NaN	NaN	NaN	Suggested citation: United Nations, Department of Economic and Social Affairs (2015). Trends in International Migrant Stock: The 2015 revision (United Nations database, POP/DB/MIG/Stock/Rev.2015).	NaN	NaN	NaN	NaN	NaN
12	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
13	Sort order	Major area, region, country or area of destination	Notes	Country code	Type of data (a)	International migrant stock at mid-year (both sexes)	NaN	NaN	NaN	NaN
14	NaN	NaN	NaN	NaN	NaN	1990	1995	2000	2005	2010.0
15	1	WORLD	NaN	900	NaN	152563212	160801752	172703309	191269100	221714243.0
16	2	Developed regions	(b)	901	NaN	82378628	92306854	103375363	117181109	132560325.0
17	3	Developing regions	(c)	902	NaN	70184584	68494898	69327946	74087991	89153918.0

If the previous 12 rows of the cover page are removed, row 13 will automatically become the column names. Yet, the problem is that the attribute of years (1990 to 2015) will still be stored in the cell rather than the column headers. This problem will still violates tidy data principles #1, and it is necessary to figure out a method that can store the attribute of years back to column headers with the other attributes.

The solution I give for this problem is to remove the entire rows from 0-15, the only data that is kept in the table are values, without column headers. The next step is to rename the entire column names. The good part about these two methods is that it helps me to remove the cover page error, and I get to write whichever variable I need as column names. (pic_2)

```
table_1_1 = table_01.drop(table_01.index[0:15])
table_1_1 #Drop 0-15 rows
```

	Unnamed: 0	Unnamed: 1	Unnamed: 2	Unnamed: 3	Unnamed: 4	Unnamed: 5	Unnamed: 6	Unnamed: 7	Unnamed: 8	Unnamed: 9	...	Unnamed: 13
15	1	WORLD	NaN	900	NaN	152563212	160801752	172703309	191269100	221714243.0	...	87884839
16	2	Developed regions	(b)	901	NaN	82378628	92306854	103375363	117181109	132560325.0	...	50536796
17	3	Developing regions	(c)	902	NaN	70184584	68494898	69327946	74087991	89153918.0	...	37348043

➢ Step Two: Rename Columns:

Based on the content of "United Nation: Trends in International Migrant Stock" , the column headers of the table one should be the names of areas and the migrant stock of the area from different genders and years. In order to ensure all attributes are indicated the column headers, I rewrite years, migrant stock, and gender in the column headers as follows.

```
[5] table_1_1.columns
```

```
[6] table_1_1.columns=['Order','Name of Country','Notes','Country Code','Type of Data(a)','1990 Migrant Stock Both Sex',
                       '1995 Migrant Stock Both Sex','2000 Migrant Stock Both Sex','2005 Migrant Stock Both Sex',
                       '2010 Migrant Stock Both Sex','2015 Migrant Stock Both Sex','1990 Migrant Stock Male',
                       '1995 Migrant Stock Male','2000 Migrant Stock Male','2005 Migrant Stock Male','2010 Migrant Stock Ma
                       '2015 Migrant Stock Male','1990 Migrant Stock Female','1995 Migrant Stock Female','2000 Migrant Stock
                       '2005 Migrant Stock Female','2010 Migrant Stock Female',
                       '2015 Migrant Stock Female']
```

```
[7] table_1_1 #rename the columns
```

	Order	Name of Country	Notes	Country Code	Type of Data(a)	1990 Migrant Stock Both Sex	1995 Migrant Stock Both Sex	2000 Migrant Stock Both Sex	2005 Migrant Stock Both Sex	2010 Migrant Stock Both Sex	...	2000 Migrant Stock Male	2005 Migrant Stock Male
15	1	WORLD	NaN	900	NaN	152563212	160801752	172703309	191269100	221714243.0	...	87884839	97866674
16	2	Developed regions	(b)	901	NaN	82378628	92306854	103375363	117181109	132560325.0	...	50536796	57217777
17	3	Developing regions	(c)	902	NaN	70184584	68494898	69327946	74087991	89153918.0	...	37348043	40648897

The data of migrant stock in table_01 are now categorized from every five years for each gender. (pic_3)

➢ Step Three: Separate Informations in Column Headers

The column headers are now able to explain the variable below them, but another problem appears. According to tidy data principle #2, one column should not contain more than one value. The column headers I just renamed violate this principle. The expected column headers should be able to carry year, migrant stock, and gender respectively, but achieving this format will require considerable effort.

The best method I could first consider is trying to separate the column headers based on string, by using the functions of assign and lambda. But before the functions can separate strings, the column headers should be pulled into the cell. I first used the melt function to pull the column headers which contain years, migrant stock, and gender into cells, (pic_4) and then I am able to use the functions of assign and lambda to separate years, migrant stocks, and gender into different cells. (pic_5)

```
[8] tidy_table_01 = table_1_1.melt(id_vars=["Order","Name of Country","Notes","Country Code","Type of Data(a)"],var_name="Year",value_name="Migrant Stock")
```

```
tidy_table_01
```

	Order	Name of Country	Notes	Country Code	Type of Data(a)	Year	Migrant Stock
0	1	WORLD	NaN	900	NaN	1990 Migrant Stock Both Sex	152563212
1	2	Developed regions	(b)	901	NaN	1990 Migrant Stock Both Sex	82378628
2	3	Developing regions	(c)	902	NaN	1990 Migrant Stock Both Sex	70184584
3	4	Least developed countries	(d)	941	NaN	1990 Migrant Stock Both Sex	11075966
4	5	Less developed regions excluding least develop...	NaN	934	NaN	1990 Migrant Stock Both Sex	59105261

```
tidy_table_01=(tidy_table_01.assign(year = lambda x: x.Year.str[:4].astype(str), Gender = lambda x: x.Year.str[18:].astype(str)).drop("Year",axis=1))
tidy_table_01.sort_values(by ='Order').reset_index().drop(columns = 'index') #Separate the Year cloumns into year, gemder, migrant stock
```

	Order	Name of Country	Notes	Country Code	Type of Data(a)	Migrant Stock	year	Gender
0	1	WORLD	NaN	900	NaN	152563212	1990	Both Sex
1	1	WORLD	NaN	900	NaN	93402426	2005	Female
2	1	WORLD	NaN	900	NaN	84818470	2000	Female
3	1	WORLD	NaN	900	NaN	172703309	2000	Both Sex
4	1	WORLD	NaN	900	NaN	79064275	1995	Female

After applying the melt function and the assign & lambda functions, the years, migrant stock, and gender are divided into different columns, and the column header of the table no longer violates the tidy data principles.

- ## Step Four: Reorganizing Repeated Observation

Even though the column headers are clean and organized, the observations are found to be constantly repeated on the left hand side of the table. Since there are three gender units (both sexes, male, and female) and five year units (1990, 1995, 2000, 2005, 2010, and 2015), labeling the observation with each gender by each year will automatically repeat every observation by 17 times, according to the new table. (pic_6) A constantly repeated observation violate the tidy data principle #4, and one observation is only allowed to be shown one time in a table.

index	Order	Name of Country	Notes	Country Code	Type of Data(a)	Migrant Stock	year	Gender
0	1	WORLD	NaN	900	NaN	152563212	1990	Both Sex
1	1	WORLD	NaN	900	NaN	93402426	2005	Female
2	1	WORLD	NaN	900	NaN	84818470	2000	Female
3	1	WORLD	NaN	900	NaN	172703309	2000	Both Sex
4	1	WORLD	NaN	900	NaN	79064275	1995	Female
5	1	WORLD	NaN	900	NaN	117584801	2015	Female
6	1	WORLD	NaN	900	NaN	74815702	1990	Female
7	1	WORLD	NaN	900	NaN	160801752	1995	Both Sex
8	1	WORLD	NaN	900	NaN	191269100	2005	Both Sex
9	1	WORLD	NaN	900	NaN	221714243	2010	Both Sex
10	1	WORLD	NaN	900	NaN	114613714	2010	Male
11	1	WORLD	NaN	900	NaN	97866674	2005	Male
12	1	WORLD	NaN	900	NaN	243700236	2015	Both Sex
13	1	WORLD	NaN	900	NaN	87884839	2000	Male
14	1	WORLD	NaN	900	NaN	77747510	1990	Male
15	1	WORLD	NaN	900	NaN	126115435	2015	Male
16	1	WORLD	NaN	900	NaN	107100529	2010	Female
17	1	WORLD	NaN	900	NaN	81737477	1995	Male
18	2	Developed regions	(b)	901	NaN	50536796	2000	Male

To reduce the repetition of observation, I decide to keep either years as column headers or keep gender as column headers. This method can largely reduce the number of repeats to either three times (repeated by gender unit) or five times (repeated by year unit), but the dilemma is which attribute I keep as column names. After careful consideration, I chose to keep the years in the table.I keep the observation repeated by years, and separate the table_01 into three tables to identify each gender (both sexes, female, and male).

```
[14] #Split the data in different table, start fromn row 1590, tidy_table_1_Both_sexes = Migrant Stock for Both Sexes
     split_point = 1590
     print(split_point)
     tidy_table_01_Both_sexes = tidy_table_01.iloc[:split_point]
     tidy_table_01_Both_sexes
```

```
[15] tidy_table_01_Both_sexes.columns=['Order','Name of Country','Notes','Country Code','Type of Data(a)',"Migrant Stock for Both Sexes","year","Gender"]
     tidy_table_1_Both_sexes = tidy_table_01_Both_sexes.drop("Gender",axis=1)
     tidy_table_1_Both_sexes
```

	Order	Name of Country	Notes	Country Code	Type of Data(a)	Migrant Stock for Both Sexes	year
0	1	WORLD	NaN	900	NaN	152563212	1990
1	2	Developed regions	(b)	901	NaN	82378628	1990
2	3	Developing regions	(c)	902	NaN	70184584	1990
3	4	Least developed countries	(d)	941	NaN	11075966	1990
4	5	Less developed regions excluding least develop...	NaN	934	NaN	59105261	1990
...	...	...	...	...	...	...	...
1585	261	Samoa	NaN	882	B	4929.0	2015
1586	262	Tokelau	NaN	772	B	487.0	2015
1587	263	Tonga	NaN	776	B	5731.0	2015
1588	264	Tuvalu	NaN	798	C	141.0	2015
1589	265	Wallis and Futuna Islands	NaN	876	B	2849.0	2015

```
[ ] #Split the data in different table, start fromn row 1590, tidy_table_1_male = Migrant Stock for Male
    split_point = 1590
    split_male_female = 3180
    tidy_table_01_Male = tidy_table_01.iloc[split_point:split_male_female]
    tidy_table_01_Male
```

```
[ ] tidy_table_01_Male.columns=['Order','Name of Country','Notes','Country Code','Type of Data(a)',"Migrant Stock for Male","year","Gender"]
    tidy_table_1_Male = tidy_table_01_Male.drop("Gender",axis=1)
    tidy_table_1_Male
```

```
[ ] #Split the data in different table, start fromn row 3180, tidy_table_1_female = Migrant Stock for female
    split_male_female = 3180
    tidy_table_1_female = tidy_table_01.iloc[split_male_female:]
    tidy_table_1_female
```

```
[ ] tidy_table_1_female.columns=['Order','Name of Country','Notes','Country Code','Type of Data(a)',"Migrant Stock for female","year","Gender"]
    tidy_table_1_female = tidy_table_1_female.drop("Gender",axis=1)
    tidy_table_1_female
```

The reason I give for splitting the table by gender rather than years is that the raw data in excel gives priority to gender, and the attribute of year is assigned below it. That means if there is an order in explaining the migrant stock by the attributes of gender and years, the order has to be each gender from every five years since 1990, instead of each year from gender of both sexes, male, and female. (pic_10)

Sort order	Major area, region, country or area of destination	Notes	Country code	Type of data (a)	Female migrants as a percentage of the international migrant stock					
					1990	1995	2000	2005	2010	2015

After splitting table_01 into three sub-tables based on gender, the observations are still repeated by years. I then bring the attribute of years back to the column headers, so that the observations in the sub-table are only going to appear once, and the migrant stock of every observation from 1990 to 2015 are stored in the same row as the observation. (pic_11)

```
table1_Bothsexes = tidy_table_1_Both_sexes.reset_index().groupby(['Order','Name of Country','Country Code','year'])['Migrant Stock for Both Sexes'].aggregate('first'
table1_Bothsexes
```

1 to 25 of 265 entries Filter

Order	Name of Country	Country Code	1990	1995	2000	2005	2010	2015
1	WORLD	900	152563212	160801752	172703309	191269100	221714243.0	243700236.
2	Developed regions	901	82378628	92306854	103375363	117181109	132560325.0	140481955.
3	Developing regions	902	70184584	68494898	69327946	74087991	89153918.0	103218281.
4	Least developed countries	941	11075966	11711703	10077824	9809634	10018128.0	11951316.
5	Less developed regions excluding least developed countries	934	59105261	56778501	59244124	64272611	79130668.0	91262036.
6	Sub-Saharan Africa	947	14690319	15324570	13716539	13951086	15496764.0	18993986.
7	Africa	903	15690623	16352814	14800306	15191146	16840014.0	20649557.
8	Eastern Africa	910	5964031	5022742	4844795	4745792	4657063.0	6129113.
9	Burundi	108	333110	254853	125628	172874	235259.0	286810.
10	Comoros	174	14079	13939	13799	13209	12618.0	12555.

```
table1_male = tidy_table_1_Male.reset_index().groupby(['Order','Name of Country','Country Code','year'])['Migrant Stock for Male'].aggregate('first').unstack()
table1_male
```

1 to 25 of 265 entries Filter

Order	Name of Country	Country Code	1990	1995	2000	2005	2010	2015
1	WORLD	900	77747510	81737477	87884839	97866674	114613714.0	126115435.0
2	Developed regions	901	40263397	45092799	50536796	57217777	64081077.0	67618619.0
3	Developing regions	902	37484113	36644678	37348043	40648897	50532637.0	58496816.0
4	Least developed countries	941	5843107	6142712	5361902	5383009	5462714.0	6463217.0
5	Less developed regions excluding least developed countries	934	31641006	30501966	31986141	35265888	45069923.0	52033599.0
6	Sub-Saharan Africa	947	7745306	8036824	7210452	7444048	8188581.0	10099486.0
7	Africa	903	8279564	8616931	7856358	8231437	9039314.0	11123423.0
8	Eastern Africa	910	3071189	2585053	2480584	2529460	2366216.0	3109176.0

```
table1_female = tidy_table_1_female.reset_index().groupby(['Order','Name of Country','Country Code','year'])['Migrant Stock for female'].aggregate('first').unstack()
table1_female
```

1 to 25 of 265 entries Filter

Order	Name of Country	Country Code	1990	1995	2000	2005	2010	2015
1	WORLD	900	74815702	79064275	84818470	93402426	107100529.0	117584801.0
2	Developed regions	901	42115231	47214055	52838567	59963332	68479248.0	72863336.0
3	Developing regions	902	32700471	31850220	31979903	33439094	38621281.0	44721465.0
4	Least developed countries	941	5236216	5573685	4721920	4432371	4560536.0	5493028.0
5	Less developed regions excluding least developed countries	934	27464255	26276535	27257983	29006723	34060745.0	39228437.0
6	Sub-Saharan Africa	947	6945013	7287746	6506087	6507038	7308183.0	8894500.0
7	Africa	903	7411059	7735883	6943948	6959709	7800700.0	9526134.0
8	Eastern Africa	910	2892842	2437689	2364211	2216332	2290847.0	3019937.0

- ➢ Step Five: Categorizing Observations

The rows and columns look clear after splitting the table and organizing the column headers. Yet, the category of the observation is not being classified. According to the tidy data principles four, one table should not contain more than one kind of observation. The observations in the tables above contains countries and continents, this is a violation of the tidy data principle #4. The solution is to build a new dataframe especially for the observations of continents and other regions, and keep the observations of countries in the original table. I first located the number of rows the non-country observation at, and used it to build a new dataframe. Then, the non-country observation will be dropped out of the original dataframe, so that the original data frame keeps only the one that is countries.

```
table01_Continent_or_region_BothSexes = table1_Bothsexes.loc[[1,2,3,4,5,6,7,8,29,39,47,53,71,72,78,86,98,106,127,128,139,153,170,180,181,108,217,232,238,239,242,248,256],
table01_Continent_or_region_BothSexes
```

Order	Name of Country	Country Code	1990	1995	2000	2005	2010	2015
1	WORLD	900	152563212	160801752	172703309	191269100	221714243.0	243700236.0
2	Developed regions	901	82378628	92306854	103375363	117181109	132560325.0	140481955.0
3	Developing regions	902	70184584	68494898	69327946	74087991	89153918.0	103218281.0
4	Least developed countries	941	11075966	11711703	10077824	9809634	10018128.0	11951316.0
5	Less developed regions excluding least developed countries	934	59105261	56778501	59244124	64272611	79130668.0	91262036.0
6	Sub-Saharan Africa	947	14690319	15324570	13716539	13951086	15496764.0	18993986.0
7	Africa	903	15690623	16352814	14800306	15191146	16840014.0	20649557.0
8	Eastern Africa	910	5964031	5022742	4844795	4745792	4657063.0	6129113.0
29	Middle Africa	911	1460530	2646108	1756687	1928828	2139979.0	2307688.0
39	Northern Africa	912	2403200	2081640	1885650	1782054	1921613.0	2159048.0
47	Southern Africa	913	1392359	1191582	1222314	1439426	2203306.0	3435194.0
53	Western Africa	914	4470503	5410742	5090860	5295046	5918053.0	6618514.0
71	Asia	935	48142261	46548225	49340815	53371224	65914319.0	75081125.0
72	Central Asia	5500	6630683	5890035	5183872	5238699	5262414.0	5393504.0

```
clean_table1_Bothsexes  =  table1_Bothsexes.drop([1,2,3,4,5,6,7,8,29,39,47,53,71,72,78,86,98,106,127,128,139,153,170,180,181,108,217,232,238,239,242,248,256],  inplace=False)
clean_table1_Bothsexes
```

		year	1990	1995	2000	2005	2010	2015
Order	Name of Country	Country Code						
9	Burundi	108	333110	254853	125628	172874	235259.0	286810.0
10	Comoros	174	14079	13939	13799	13209	12618.0	12555.0
11	Djibouti	262	122221	99774	100507	92091	101575.0	112351.0
12	Eritrea	232	11848	12400	12952	14314	15676.0	15941.0
13	Ethiopia	231	1155390	806904	611384	514242	567720.0	1072949.0

➢ **Step Six: Replacing Missing Value**

Other than considering the violation of tidy data principles, the missing value in all of the data are equally worth to concern. By closely looking into the raw data in excel file, I discover that there are considerable amounts of missing value, and the missing spaces are filled out by ".." in it.

23	Somalia	706	478294	19527	20087	20670	23995.0	25291.0
24	South Sudan	728	..	..	..	..	257905.0	824122.0
25	Uganda	800	558307	634620	634703	652968	529160.0	749471.0

Based on this feature, I decided to replace the cells that contain ".." with the string of "Missing_val".

23	Somalia	706	478294	19527	20087	20670	23995.0	25291.0
24	South Sudan	728	Missing_value	Missing_value	Missing_value	Missing_value	257905.0	824122.0
25	Uganda	800	558307	634620	634703	652968	529160.0	749471.0

Compared to replacing the value with the mean of the column's variable, I believe that replacing it with the string of "Missing_val" could be a better method, just so the real data can be filled in later on. Suppose the missing values are replaced by the mean of the column variables, the result may be less disturbing for overall data. However, it makes the missing value less visible once it is replaced, and it requires much more jobs to fill in the real data if the real data is available. At the sametime, if the researcher is interested in looking into data individually, the cells will fall to represent the real observation if it is being replaced by mean. Thus, I decided to replace the missing value with "Missing_val".

- <u>Discussion:</u>

This data cleaning report mainly focuses on removing error, avoiding five tidy data principles, and replacing missing value. Problems about errors and missing value are thoroughly taken care of, but I discover that I am incapable of avoiding all of the five principles about tidy data. The steps of reorganizing the column headers and splitting the tables into sub-tables ensure that the dataframes do not violate principles 1, 2, and 4, but the principle of 5 is unable to be satisfied under such changes. The decision of splitting the tables makes the same observation repeated once in every sub-table, and this is a violation of principle 5. However, if tables are not splitted, observations will be repeated multiple times in a table. This issue also violate principle #4. Thus, one of the problems I would continue to study after this report is to manage the conflict between principle 4 and principle 5.

- <u>Conclusion:</u>

The data cleaning process in this assignment is mainly working on removing errors, avoiding tidy data principle violation, and filling in missing values. Majority of the tasks are focused on correcting the violation of data. By rearranging the columns and splitting tables based on the attribute of gender, the data frame is able to avoid most of the violations in tidy data principle, except principle #5. In the process of cleaning the data, the conflict between principle #4 and principle #5 is discovered. This assignment is only able to solve the violation of principle #4, and the violation of principle #5 will be continually working on, until the conflict between principle #4 and principle #5 can be solved.