Bryan Zhu
1004892342
INF1340 Midterm Submission Writeup

Cleaning the UN Migrant Stock Dataset

Introduction

For this project, we needed to clean a dataset published by the United Nations titled "*Trends in International Migrant Stock: The 2015 Revision*". This dataset is in the form of an Excel file, and consists of 6 data tables, along with a contents sheet, an annex sheet, and a notes sheet. The data within the 6 data tables are not properly organized and violate various principles of tidy data. The purpose of this project is to clean these data sets in order to make them more suitable for further data analysis and visualization. Cleaning of the data sets will be done by following the principles of tidy data, which are designed to provide a standardized method of organization within data sets for data scientists. By doing this, it will be easier to perform further analysis using datasets that are uniformly standard.

Methods

The principles of tidy data are as followed:

Principle 1: Column names need to be informative, variable names and not values

Principle 2: Each column needs to consist of one and only one variable

Principle 3: Variables need to be in cells, not rows and columns

Principle 4: Each column needs to have a singular data type

Principle 5: A single observational unit must be in one table

The first step is to look through the UN data sets to see if/how they violate any of these principles of tiny data. After this, they can begin to be cleaned. The 6 data sheets have very

similar formatting, with the exception of Table 4 having only one sex (females) and Table 6 conducting different measurements instead of using different sexes. Thus, they can all be cleaned using similar procedures.

First and foremost the standard libraries used for data cleaning need to be imported into Google Colab:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import plotly
```

Next, the Excel file containing the UN migrant data is loaded into Colab:

```
[92] from google.colab import drive
     drive.mount('/content/drive')
     migrantdata = pd.ExcelFile('/content/drive/My Drive/UN_MigrantStockTotal_2015.xlsx')
     #import the UN Migrant Data excel file from Google Drive
     #rename the file as "migrantdata"
```

The first table is loaded from the Excel file. The first 14 rows of the Excel file consist of various UN titles and logos, so they are ignored for data cleaning purposes:

```
[93] table1 = pd.read_excel(migrantdata, sheet_name = "Table 1", header = [0,1], skiprows = 14)
     table1.head()
     #reading Table 1 from migrantdata
     #skip rows function to skip all the UN header rows
```

	Sort\norder	Major area, region, country or area of destination	Notes	Country code	Type of data (a)	International migrant stock at mid-year (both sexes)					...	International migrant stock at mid-year (male)				International migran		
	Unnamed: 0_level_1	Unnamed: 1_level_1	Unnamed: 2_level_1	Unnamed: 3_level_1	Unnamed: 4_level_1	1990	1995	2000	2005	2010	...	2000	2005	2010	2015	1990	1995	2(
0	1	WORLD	NaN	900	NaN	152563212	160801752	172703309	191269100	221714243	...	87884839	97866674	114613714	126115435	74815702	79064275	84
1	2	Developed regions	(b)	901	NaN	82378628	92306854	103375363	117181109	132560325	...	50536796	57217777	64081077	67618619	42115231	47214055	52
2	3	Developing regions	(c)	902	NaN	70184584	68494898	69327946	74087991	89153918	...	37348043	40648897	50532637	58496816	32700471	31850220	31
3	4	Least developed countries	(d)	941	NaN	11075966	11711703	10077824	9809634	10018128	...	5361902	5383009	5462714	6463217	5236216	5573685	4
4	5	Less developed regions excluding least develop...	NaN	934	NaN	59105261	56778501	59244124	64272611	79130668	...	31986141	35265888	45069923	52033599	27464255	26276535	27

5 rows × 23 columns

From here, we can see a number of violations. There are columns that are uninformative, there are multiple column headers, there are variables in the column headers and not in the cells themselves, and there are multiple variable types within some columns. We can then begin the data cleaning process.

First, the “Notes” column can be dropped as it is uninformative within this data set:

```
table1.drop("Notes", inplace = True, axis = 1)
table1.head()
#drop the Notes column as it is uninformative
```

```
/usr/local/lib/python3.7/dist-packages/pandas/core/generic.py:4150: PerformanceWarning: dropping on a non-lexsorted multi-index without a level parameter may impact performa
  obj = obj._drop_axis(labels, axis, level=level, errors=errors)
```

Sort\norder	Major area, region, country or area of destination	Country code	Type of data (a)	International migrant stock at mid-year (both sexes)						...	International migrant stock at mid-year (male)				International migrant		
Unnamed: 0_level_1	Unnamed: 1_level_1	Unnamed: 3_level_1	Unnamed: 4_level_1	1990	1995	2000	2005	2010	2015	...	2000	2005	2010	2015	1990	1995	20
0	1	WORLD	900	NaN	152563212	160801752	172703309	191269100	221714243	243700236	...	87884839	97866674	114613714	126115435	74815702	79064275	84
1	2	Developed regions	901	NaN	82378628	92306854	103375363	117181109	132560325	140481955	...	50536796	57217777	64081077	67618619	42115231	47214055	52
2	3	Developing regions	902	NaN	70184584	68494898	69327946	74087991	89153918	103218281	...	37348043	40648897	50532637	58496816	32700471	31850220	31
3	4	Least developed countries	941	NaN	11075966	11711703	10077824	9809634	10018128	11951316	...	5361902	5383009	5462714	6463217	5236216	5573685	4
4	5	Less developed regions excluding least develop...	934	NaN	59105261	56778501	59244124	64272611	79130668	91262036	...	31986141	35265888	45069923	52033599	27464255	26276535	27

5 rows × 22 columns

Next, because there are two column headers, they should be combined into one singular header:

```
[95] table1.columns = table1.columns.map("{0[0]}{0[1]}".format)
     table1.head(2)
     #merge the 2 rows of column headers into just one row
```

Sort\norderUnnamed: 0_level_1	Major area, region, country or area of destinationUnnamed: 1_level_1	Country codeUnnamed: 3_level_1	Type of data (a)Unnamed: 4_level_1	International migrant stock at mid-year (both sexes)1990	International migrant stock at mid-year (both sexes)1995	International migrant stock at mid-year (both sexes)2000	International migrant stock at mid-year (both sexes)2005	International migrant stock at mid-year (both sexes)2010	International migrant stock at mid-year (both sexes)2015	...	Internati migrant st at mid- (male):
0	1	WORLD	900	NaN	152563212	160801752	172703309	191269100	221714243	243700236	...	8788
1	2	Developed regions	901	NaN	82378628	92306854	103375363	117181109	132560325	140481955	...	5053

2 rows × 22 columns

These headers themselves have strange variable names, so for readability these headers are renamed:

```
table1 = table1.rename (columns={
    "Sort\norderUnnamed: 0_level_1": "Sortorder", "Major area, region, country or area of destinationUnnamed: 1_level_1": "Country/Region/Area",
    "Country codeUnnamed: 3_level_1": "Country Code", "Type of data (a)Unnamed: 4_level_1": "Data Type",
    "International migrant stock at mid-year (both sexes)1990": "b1990",
    "International migrant stock at mid-year (both sexes)1995": "b1995",
    "International migrant stock at mid-year (both sexes)2000": "b2000",
    "International migrant stock at mid-year (both sexes)2005": "b2005",
    "International migrant stock at mid-year (both sexes)2010": "b2010",
    "International migrant stock at mid-year (both sexes)2015": "b2015",
    "International migrant stock at mid-year (male)1990": "m1990",
    "International migrant stock at mid-year (male)1995": "m1995",
    "International migrant stock at mid-year (male)2000": "m2000",
    "International migrant stock at mid-year (male)2005": "m2005",
    "International migrant stock at mid-year (male)2010": "m2010",
    "International migrant stock at mid-year (male)2015": "m2015",
    "International migrant stock at mid-year (female)1990": "f1990",
    "International migrant stock at mid-year (female)1995": "f1995",
    "International migrant stock at mid-year (female)2000": "f2000",
    "International migrant stock at mid-year (female)2005": "f2005",
    "International migrant stock at mid-year (female)2010": "f2010",
    "International migrant stock at mid-year (female)2015": "f2015",})
#rename the column headers to shorten them for readibility
table1.head(2)
```

	Sortorder	Country/Region/Area	Country Code	Data Type	b1990	b1995	b2000	b2005	b2010	b2015	...	m2000	m2005	m2010	m2015	f1990	f1995	f
0	1	WORLD	900	NaN	152563212	160801752	172703309	191269100	221714243	243700236	...	87884839	97866674	114613714	126115435	74815702	79064275	8481
1	2	Developed regions	901	NaN	82378628	92306854	103375363	117181109	132560325	140481955	...	50536796	57217777	64081077	67618619	42115231	47214055	5283

2 rows × 22 columns

However, there are still columns that have variables in the column headers. Namely, the sex and year are still in the headers, but they should be variables in cells. We can use the melt function from pandas to move these variables into the cells:

```
[97] table1 = table1.melt(id_vars = ["Sortorder", "Country/Region/Area", "Country Code", "Data Type"], var_name = "demographic", value_name = "Migrant Stock")
     table1.head()
     #melt the sex/year (demographic) so that they no longer values in the column header
     #sex/year (demographic) are now variables in table cells
```

	Sortorder	Country/Region/Area	Country Code	Data Type	demographic	Migrant Stock
0	1	WORLD	900	NaN	b1990	152563212
1	2	Developed regions	901	NaN	b1990	82378628
2	3	Developing regions	902	NaN	b1990	70184584
3	4	Least developed countries	941	NaN	b1990	11075966
4	5	Less developed regions excluding least develop...	934	NaN	b1990	59105261

Now, there is the issue of sex and year being in the same column within this data table. Each column can only have one variable and one data type within it. So, we need to split the "demographic" column into two separate columns:

```
[98] table1 = table1.assign(Sex = lambda x: x.demographic.str[0].astype(str), Year = lambda x: x.demographic.str[1:].astype(str)).drop("demographic",axis=1)
     table1.head()
     #split the demographic variable into separate unique sex and year variables
```

	Sortorder	Country/Region/Area	Country Code	Data Type	Migrant Stock	Sex	Year
0	1	WORLD	900	NaN	152563212	b	1990
1	2	Developed regions	901	NaN	82378628	b	1990
2	3	Developing regions	902	NaN	70184584	b	1990
3	4	Least developed countries	941	NaN	11075966	b	1990
4	5	Less developed regions excluding least develop...	934	NaN	59105261	b	1990

Lastly, for improved comprehensibility the variables under the “sex” column are renamed for easier viewing:

```
table1 = table1.replace(to_replace = ["b", "m", "f"], value = ["both", "male", "female"])
table1.sample(10)
#stylized so the sex variable is more descriptive
```

	Sortorder	Country/Region/Area	Country Code	Data Type	Migrant Stock	Sex	Year
2153	34	Congo	178	B	152666	male	2000
1826	237	United States of America	840	B	11372985	male	1990
1410	86	South-Eastern Asia	920	NaN	9867722	both	2015
999	205	Trinidad and Tobago	780	B	44812	both	2005
3628	184	Aruba	533	B	12300	female	1995
3752	43	Morocco	504	C	25801	female	2000
0	1	WORLD	900	NaN	152563212	both	1990
3678	234	Canada	124	B	2508673	female	1995
4256	17	Mauritius	480	C	11648	female	2010
2454	70	Togo	768	C R	102877	male	2005

Table 1 of the UN migrant data set has now been cleaned. The cleaning process for Tables 2 and 3 is repeated nearly identically, with certain variable names being changed. Below are the final tables for Tables 2 and 3:

```
[106] table2 = table2.replace(to_replace = ["b", "m", "f"], value = ["both", "male", "female"])
      table2.sample(10)
      #stylized so the sex variable is more descriptive
```

	Sortorder	Country/Region/Area	Country Code	Total Population	Sex	Year
262	263	Tonga	776	95.152	both	1990
3535	91	Malaysia	458	10195.405	female	1995
1307	248	Micronesia	954	502.49	both	2010
2695	46	Western Sahara	732	269.743	male	2010
975	181	Caribbean	915	40028.199	both	2005
3639	195	Guadeloupe	312	210.496	female	1995
391	127	Europe	908	727778.44	both	1995
2428	44	Sudan	729	16050.47	male	2005
2170	51	South Africa	710	22102.957	male	2000
4265	26	United Republic of Tanzania	834	22982.543	female	2010

```
table3 = table3.replace(to_replace = ["b", "m", "f"], value = ["both", "male", "female"])
table3.sample(10)
#stylize the sex variable to be more descriptive
```

	Sortorder	Country/Region/Area	Country Code	Data Type	Migrant Stock % of Total Population	Sex	Year
3409	230	Uruguay	858	B	3.264831	female	1990
3687	243	Fiji	242	B	1.645345	female	1995
1157	98	Southern Asia	5501	NaN	0.841261	both	2010
1134	75	Tajikistan	762	B	3.668731	both	2010
2477	93	Philippines	608	C R	0.307262	male	2005
2972	58	Gambia	270	B	10.299118	male	2015
951	157	Croatia	191	B R	13.231283	both	2005
3121	207	United States Virgin Islands	850	B	52.786665	male	2015
3110	196	Haiti	332	B	0.429907	male	2015
2060	206	Turks and Caicos Islands	796	B	..	male	1995

Table 4 is also cleaned similarly, however only one sex (female) is specified within this table. So, the step of splitting the combined sex/year column can be skipped:

```
[118] table4 = table4.melt(id_vars = ["Sortorder", "Country/Region/Area", "Country Code", "Data Type"], var_name = "Year",
                           value_name = "Female Migrants as % of International Migrant Stock")
      table4.head()
      #melting the year so that it is a variable rather than a column header
      #Instead of having sex as a column header, the Migrant Stock header states that the data includes only females
```

	Sortorder	Country/Region/Area	Country Code	Data Type	Year	Female Migrants as % of International Migrant Stock
0	1	WORLD	900	NaN	1990	49.03915
1	2	Developed regions	901	NaN	1990	51.123977
2	3	Developing regions	902	NaN	1990	46.592099
3	4	Least developed countries	941	NaN	1990	47.261155
4	5	Less developed regions excluding least develop...	934	NaN	1990	46.466684

Table 5 returns to the familiar formatting of Tables 1-3, so it is similarly cleaned as those tables:

```
table5 = table5.replace(to_replace = ["b", "m", "f"], value = ["both", "male", "female"])
table5.sample(10)
#stylized the sex variable to be more descriptive
```

	Sortorder	Country/Region/Area	Country Code	Data Type	Annual Rate of Change of Migrant Stock	Sex	Years
3562	118	Lebanon	422	B R	1.413896	female	2005-2010
1100	41	Egypt	818	B R	10.167201	both	2010-2015
874	80	China, Hong Kong Special Administrative Region	344	B	0.426942	both	2005-2010
3454	10	Comoros	174	B	-1.08327	female	2005-2010
1036	242	Melanesia	928	NaN	0.344728	both	2005-2010
1167	108	Western Asia	922	NaN	4.393455	both	2010-2015
1731	142	Estonia	233	B	-6.020975	male	1995-2000
684	155	Andorra	20	C	3.536511	both	2000-2005
2720	71	Asia	935	NaN	-0.717339	female	1990-1995
431	167	Slovenia	705	B	-0.393833	both	1995-2000

Table 6 is similar to Tables 1, 2, 3, and 5, but instead of the variable being “sex” in the top row, it consists of three different measurement variables. So, the data is cleaned in the same manner, but the final data table looks like this:

```
table6 = table6.replace(to_replace = ["s", "p", "r"], value = ["Estimated Stock", "Percentage of Stock", "Rate of Change (annual)"])
table6.sample(10)
#styled measurement variable to be more clear as to what the refugee column indicates
```

	Sortorder	Country/Region/Area	Country Code	Data Type	Refugees	Measurement	Years
4186	212	Guatemala	320	B R	-25.783308	Rate of Change (annual)	2005-2010
1234	175	Liechtenstein	438	B	92	Estimated Stock	2010
271	7	Africa	903	NaN	5949953	Estimated Stock	1995
2833	184	Aruba	533	B	0.0	Percentage of Stock	2010
2511	127	Europe	908	NaN	2.747442	Percentage of Stock	2005
3596	152	United Kingdom of Great Britain and Northern I...	826	B	11.752859	Rate of Change (annual)	1995-2000
520	256	Polynesia	957	NaN	0	Estimated Stock	1995
1914	60	Guinea	324	C R	86.803571	Percentage of Stock	1995
3827	118	Lebanon	422	B R	0.302694	Rate of Change (annual)	2000-2005
2949	35	Democratic Republic of the Congo	180	B R	21.945266	Percentage of Stock	2015

Discussion

The basic strategy for data cleaning is applicable to all six tables within this UN dataset due to the high degree of similarity between them. Figuring out the process and syntax for cleaning Table 1 took a lot of trial and error, and was relatively time-consuming. However, once the first table was cleaned, the cleaning process for the subsequent tables was a look quicker since the steps involved were very similar to the steps involved for cleaning Table 1. In many instances the steps were identical with the exception of changing some of the names within the lines of code.

The primary violations were that there were variables in the column headers, and that many columns consisted of more than one variable. This was present in all 6 data sheets, which was fixed using the “melt” function from pandas to pivot column variables into the cells and using the “assign” function to split columns consisting of two variables into two unique columns.

One thing that I wasn't sure about was my process for Table 6. Table 6 had the problem of columns consisting of two variables, for example both year and estimated refugee stock. There were three different measurements in the topmost row, to which I used the melt functon to pivot into the table cells, similar to the other data sets. I'm not sure if this also constitutes as a violation of tidy data principle 5, which states that a single observational unit must be in one table. If so, then I should have split Table 6 into smaller constituent tables so that I wouldn't have to declare what was being measured as a column variable in my cleaned table. I couldn't quite figure out how to successfully split the table however, so I went forward and cleaned the table in the same process as the other tables. But upon reflection, I feel like splitting the table might have been the more correct option.

Overall, while determining the correct pathing and syntax for Table 1 was challenging, it paved the way for subsequent cleaning of the rest of the data tables. The structural similarities between tables allowed for much of the code to be reused. Although somewhat tedious to have to undergo so much repetition, it simplified the process greatly.

Conclusion

This project provided good practice in identifying what tidy data is and how to spot violations of tidy data principles. I learned a lot about not only what tidy data is, but also how to utilize Python from this project. Data cleaning is tedious and non-prestigious, but still a crucial part of the data science process. Proper tidy data allows for further analyses, some of which may be quite sophisticated, but none of which would be possible without first knowing how to produce a cleaned data set.