Tidy 2015 United Nations Trend In International Migration Stock Dataset

Name:Hui Li

1 Dataset Description

There are six tables in the dataset. Six tables contain data of the international migrant stock, total population, international migrant stock as percentages of total population, female migrants as percentages of the international migrant stock, annual rate of change of the migrant stock, and estimated refugee stock, respectively. Each table is grouped by country codes, mid-years, countries, locations, and genders.

2 Import Useful Packages For Tidying

```
[ ]  # Import needed libraries

     import pandas as pd
     import altair as alt
     import numpy as np
```

3 The Process of Tidying Each Table

1) Import useful packages for tidying the data

```python
# Import table 1 from the UN 2015 data

data = pd.ExcelFile('UN_MigrantStockTotal_2015.xlsx')
df1 = pd.read_excel(data,'Table 1')
df1.head(25)
```

	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 201...	NaN	NaN	NaN	NaN	NaN

2) In this part, I also looked at the first 25 rows of each table to examine if the first few rows are useful. After executing df1.head(25), I found that the first 13 rows are not useful, containing many NaN values. As a result, I deleted those rows, maintained other rows, and reindex starting from row 13.

```python
# Keep necessary parts and delete unuseful ones
df1 = df1[df1.columns[1:23]].iloc[13:281]
df1 = df1.reindex(df1.index.drop(13))
df1
```

	Unnamed: 1	Unnamed: 2	Unnamed: 3	Unnamed: 4	Unnamed: 5	Unnamed: 6	Unnamed: 7	Unnamed: 8	Unnamed: 9	Unnamed: 10	...	U
14	NaN	NaN	NaN	NaN	1990	1995	2000	2005	2010.0	2015.0	...	
15	WORLD	NaN	900	NaN	152563212	160801752	172703309	191269100	221714243.0	243700236.0	...	8
16	Developed regions	(b)	901	NaN	82378628	92306854	103375363	117181109	132560325.0	140481955.0	...	5
17	Developing regions	(c)	902	NaN	70184584	68494898	69327946	74087991	89153918.0	103218281.0	...	3
18	Least developed countries	(d)	941	NaN	11075966	11711703	10077824	9809634	10018128.0	11951316.0	...	

3) From the previous part, I observed that columns “Unnamed: 2” and “Unnamed: 4” contain useless information and null values. Hence, I deleted those rows. Moreover, I reset the indices from 0 to 265 because there were 266 rows after deletion.

```python
# Drop Unnamed:2 and Unnamed:4 (Both contain many NaN values)
# We delete 14 indices, so we maintain 266 indices
df1 = df1.drop(['Unnamed: 2','Unnamed: 4'],axis=1)
df1_index = pd.Series(range(266))
df1= df1.set_index([df1_index])
df1
```

	Unnamed: 1	Unnamed: 3	Unnamed: 5	Unnamed: 6	Unnamed: 7	Unnamed: 8	Unnamed: 9	Unnamed: 10	Unname
0	NaN	NaN	1990	1995	2000	2005	2010.0	2015.0	1
1	WORLD	900	152563212	160801752	172703309	191269100	221714243.0	243700236.0	77747
2	Developed regions	901	82378628	92306854	103375363	117181109	132560325.0	140481955.0	40263
3	Developing regions	902	70184584	68494898	69327946	74087991	89153918.0	103218281.0	37484
4	Least developed countries	941	11075966	11711703	10077824	9809634	10018128.0	11951316.0	5843
...	...	...	...	...	...	...	...	...	
261	Samoa	882	3357	4694	5998	5746	5122.0	4929.0	1
262	Tokelau	772	270	266	262	258	429.0	487.0	

4) I realized that row 0 contained some null values. I replaced those values with corresponding categories of the columns as well as creating meaningful column names.

```python
# Create meaningful column names
df1.columns = ['Country/Location', 'Code','International Migrant Stock at mid-year (Both Sexes)','International Migrant Stock at mid-year (Both Sexes)',
               'International Migrant Stock at mid-year (Both Sexes)', 'International Migrant Stock at mid-year (Both Sexes)','International Migrant Stock at mid-year (Both Sexes)'
               'International Migrant Stock at mid-year (Both Sexes)', 'International Migrant Stock at mid-year (Male)', 'International Migrant Stock at mid-year (Male)',
                'International Migrant Stock at mid-year (Male)', 'International Migrant Stock at mid-year (Male)',
               'International Migrant Stock at mid-year (Male)', 'International Migrant Stock at mid-year (Male)', 'International Migrant Stock at mid-year (Female)',
                'International Migrant Stock at mid-year (Female)', 'International Migrant Stock at mid-year (Female)',
               'International Migrant Stock at mid-year (Female)', 'International Migrant Stock at mid-year (Female)', 'International Migrant Stock at mid-year (Female)']
# replace NaN value with the corresponding column name
df1["Country/Location"] =  df1["Country/Location"].fillna("Country/Location")
df1["Code"] = df1["Code"].fillna("Code")
df1
```

Country/Location	Code	International Migrant Stock at mid-year (Both Sexes)	International Migrant Stock at mid-year (Both Sexes)	International Migrant Stock at mid-year (Both Sexes)	International Migrant Stock at mid-year (Both Sexes)	International Migrant Stock at mid-year (Both Sexes)	International Migrant Stock at mid-year (Both Sexes)	International Migrant Stock at mid-year (Male)	International Migrant Stock at mid-year (Male)	Internation Migrant Sto at mid-ye (Mal

5) From the data overview, there were mainly three gender groups (e.g. Both Sexes, Male, and Female). As a result, I further tidied the data by each specific gender group. I created a new table for this gender group containing necessary information (e.g. Country/Location, country code, International Migrant Stock) from the original table.

Tidy Data by Genders (Both Sexes)

```python
# Create a separate table with necessary information on both sexes
df1_BothSexes = df1[['Country/Location', 'Code','International Migrant Stock at mid-year (Both Sexes)']]
```

6) I created a new column “Gender” and set the value to be “Both” to indicate this new table is for “Both Sexes” gender group. Then I reindexed the table by the first row and set the first column as headers

```
# Add a new column "Gender" and specify the values to be "Both" for future merging.
df1_BothSexes['Gender'] = 'Both'

# Reindex the table by the first row
df1_BothSexes.iloc[0,8] ='Gender'

# First column as the headers
df1_BothSexes.columns = df1_BothSexes.iloc[0]
df1_BothSexes = df1_BothSexes.reindex(df1_BothSexes.index.drop(0))
```

7) I realized that in the header, some mid-years are in the form of decimal. As a result, I change the type to string in order to change into the form of integer. I reordered the header “Gender” to the first header to indicate that this table was for “Both Sexes” gender group. Then, I set new indices.

Note: if mid-years are in the form of “2000-2005”. I did not change the type. In this situation, I simply reordered the headers.

```
# Change the type to string in order to easily change year to integer form instead of decimal form
df1_BothSexes.columns = df1_BothSexes.columns.astype(str)
df1_BothSexes.rename(columns = {"2010.0":"2010","2015.0":"2015"},inplace = True)

# Reorder the headers in order to show that this table is for both sexes
df1_BothSexes = df1_BothSexes[['Gender','Country/Location','Code','1990','1995','2000','2005','2010','2015']]

# Set index
df1_BothSexes.set_index(['Gender','Country/Location','Code'], inplace=True)
df1_BothSexes.columns.names = ['Mid-year']
```

8) I changed the table format from wide to long and specified the column to be “`International Migrant Stock (Both Sexes)`”. This column could be different depending on the table and gender group.

```
# Change the dataframe from wide to long format
df1_BothSexes_long = df1_BothSexes.stack().to_frame()
df1_BothSexes_long.columns = ["International Migrant Stock (Both Sexes)"]
```

9) I removed empty spaces in the new table and set the values in the column to be numeric. If the values were percentages, I rounded up to three decimal places. Then, I created a pivot table based on the gender column.

```python
# Remove empty spaces
df1_BothSexes_long.replace(regex=True,inplace=True,to_replace=r'\D',value=r'')

# Change "International Migrant Stock" to numeric values
df1_BothSexes_long['International Migrant Stock (Both Sexes)'] = df1_BothSexes_long['International Migrant Stock (Both Sexes)'].apply(pd.to_numeric)

# Pivot gender column of this dataframe (Both Sexes)
df1_BothSexes_long = pd.pivot_table(df1_BothSexes_long, values='International Migrant Stock (Both Sexes)', index=['Country/Location','Code','Mid-year'],columns = 'Gender')
df1_BothSexes_long
```

		Gender	Both
Country/Location	Code	Mid-year	
Afghanistan	4	1990	57686.0
		1995	71522.0
		2000	75917.0
		2005	87300.0
		2010	102246.0
...	...	...	...
Zimbabwe	716	1995	431226.0
		2000	410041.0
		2005	392693.0
		2010	397891.0
		2015	398866.0

1575 rows × 1 columns

In the tidy process, steps 2-4 were the same for all tables. I used steps 5-9 for each gender group. If there were three gender groups in the table, I did this process three times.

10) Eventually, I merged all gender group data frames and created corresponding meaningful column names.

```python
# Merging three data frames
df1_all = [df1_BothSexes_long,df1_Male_long,df1_Female_long]
df1_merge = pd.concat(df1_all,axis = 1)
df1_merge.columns = ['International Migrant Stock (Both Sexes)',
                     'International Migrant Stock (Male)','International Migrant Stock (Female)']
df1_merge
```

			International Migrant Stock (Both Sexes)	International Migrant Stock (Male)	International Migrant Stock (Female)
Country/Location	Code	Mid-year			
Afghanistan	4	1990	57686.0	32558.0	25128.0
		1995	71522.0	39105.0	32417.0
		2000	75917.0	42848.0	33069.0
		2005	87300.0	49274.0	38026.0
		2010	102246.0	57709.0	44537.0
...	...	...	...	...	...
Zimbabwe	716	1995	431226.0	246012.0	185214.0
		2000	410041.0	233843.0	176198.0
		2005	392693.0	223970.0	168723.0
		2010	397891.0	226967.0	170924.0
		2015	398866.0	227379.0	171487.0

1575 rows × 3 columns

11) If I cleaned more or equal to two tables, I merged them together. At the end, I merged all 6 cleaned tables altogether.

Combine Data from Table 1, Table2, Table3, Table4, Table 5, and Table 6

```
# Merge the first five tables
df_all = [df1_merge, df2_merge,df3_merge,df4_FemaleMigrants_long,df5_merge,df6_merge]
UN_MigrantStockTotal_2015 = pd.concat(df_all, axis =1)

UN_MigrantStockTotal_2015 = UN_MigrantStockTotal_2015.fillna("*")
UN_MigrantStockTotal_2015
```

Country/Location	Code	Mid-year	International Migrant Stock (Both Sexes)	International Migrant Stock (Male)	International Migrant Stock (Female)	Total Population in thousands(Both Sexes)	Total Population in thousands(Male)	Total Population in thousands(Female)	In Mi pe
Afghanistan	4	1990	57686.0	32558.0	25128.0	12067.57	6179.834	5887.736	
		1990-1995	*	*	*	*	*	*	
		1995	71522.0	39105.0	32417.0	16772.522	8682.442	8090.08	
		1995-2000	*	*	*	*	*	*	
		2000	75917.0	42848.0	33069.0	19701.94	10146.537	9555.403	

12) I saw that some values in the final table were null. For instance, when the row was 1990-1995 and the column was "International Migrant Stock (Both Sexes), the value was null because international migrant stock corresponds to specific mid-year (e.g. 1990,1995) instead of year interval. (e.g. 1990-1995). As a result, I set all null values in this table to "*" for better visualization.

In conclusion, I think this is the right solution because each variable has its own column, each observation has its own row, and each value has its own cell. I filtered the most important information from each table and combined them into one table. However, I am interested in how to deal with those missing values in the table. There is probably a better way to tidy the data.