UNIVERSITY OF TORONTO
FACULTY OF INFORMATION
The iSchool

INF 1340H F LEC0101

Midterm Project
Data Wrangling + Cleaning

Clean UN Dataset Using Tidy Data Principles

Student Name: Miaomiao Yang

Nov 8th 2022

Table of Contents

Table of Figures

Introduction

The midterm project focuses on wrangling messy datasets into a tidy format, sometimes known as data tidying, which is structuring datasets to facilitate analysis. The tidy data principles provide a uniform method for organizing data values within a dataset. The tidy data standard is intended to facilitate initial data exploration as well as simplify the creation of interoperable data analysis tools (Wickham, 2014).

The objective of the midterm project is to clean six UN datasets using **Google Colab**. They are *Table 1 - International migrant Stock at mid-year by sex and by major area, region, country or area, 1990-2015*, *Table 2 - Total population at mid-year by sex and by major area, region, country or area, 1990-2015 (thousands), Table 3 - International migrant stock as a percentage of the total population by sex and by major area, region, country or area, 1990-2015, Table 4 - Female migrants as a percentage of the international migrant stock by major area, region, country or area, 1990-2015, Table 5 - Annual rate of change of the migrant stock by sex and by major area, region, country or area, 1990-2015 (percentage), and Table 6 - Estimated refugee stock at mid-year by major area, region, country or area, 1990-2015.*

The data cleaning process follows the tidy data principles based on the data characteristics of each table, including dropping of missing values, outlier detection, rewriting of column names, type conversion, and removal of superfluous columns.

Methods and Results

Tidy data is a standard way of mapping the meaning of a dataset to its structure. A dataset is messy or clean depending on how rows, columns, and tables are matched up with observations, variables, and types. The following three key ideas were kept in mind while working with messy data:

- Each column is a variable.
- Each row is an observation.
- Each cell has a single value.

In addition to the three main points, each data cleansing process follows the five tidy data principles based on how many issues were identified in each table.

1. Column names need to be informative, variable names are not values
2. Each column needs to consist of one and only one variable
3. Variables need to be in cells, not rows and columns

4. Each table column needs to have a singular data type
5. A single observational unit must be in one table

In this project, the six UN datasets are cleaned using **Google Colab**. A product from Google Research, **Google Colab** enables users to create and run arbitrary Python code through a web browser. It is particularly useful for machine learning, data analysis, and education.

<u>Applied Functions</u>

Before beginning to process data analysis, the initial stage in the data wrangling and cleaning project is to load in the *PANADAS.* Pandas is a Python package that provides data structures meant to perform real-world data analysis. Its features include handling missing data in floating-point and non-floating-point data and removing any columns and rows from a DataFrame.

At the beginning of the data analysis, we need to define each column name and import the Excel data table into **Google Colab** using the `pd.read_excel`. A brief summary of the DataFrame can be obtained using the `dataframe.info()` function. This provides a quick overview of the dataset, including the number of columns, column labels, column data types, memory usage, range index, and the number of cells in each column (non-null values).

Reshaping the data into a more computer-friendly form is the essential step in the tidy data process. Using `data.melt()` makes the table data analysis much easier. The `melt()` function can be used to transform a DataFrame into a format with one or more columns designated as identifier variables, while all other columns, considered measured variables, are unpivoted to the row axis, leaving just two non-identifier columns, variable ,and value. It works by taking observations spread across columns and melting them down into one column with multiple rows. However, we want to keep the shared metadata between the observations. By including those columns as `id_vars`, `var_name`, and `value_name`, values will be repeated as many times as needed to stay with their observations.

Due to some rows and columns missing abundant data, tidy data requires to drop of specified labels from rows or columns by `drop()` method. Remove rows or columns by specifying label names and corresponding axis or by directly specifying index or column names. In `data.drop(columns=[ ‘A’ , ’B’ ])` function, `axis = 1` specifies the labels will be dropped along the columns, `inplace=True` indicates operation inplace and return None. For dropping multiple columns, using the columns argument which is to pass a list of column names which are to be dropped. While `dropna ()` function is used to check for missing elements in columns and then drop rows by returning DataFrame with labels on the given axis omitted where (all or any) data are missing. Using `data.dropna(subset=[ 'A' , 'B' ], inplace=True)`, the *subset parameter* enables you to specify the subset of columns where `dropna` will look for missing values. `Inplace = True` is set while doing the tidy data, the `dropna` method will modify DataFrame directly, which means all missing values will be dropped from the original dataset and data will be overwritten. In `Drop ()` methods, it also included dropping values from a dataframe in an iterative way. This is a `for` loop that will pass over every value in a provided list.

Pandas provide a method to split string around a passed separator/delimiter. `str.split (', ', expand = )` method can be applied to a whole series. When using `expand=True`, the split elements will expand out into separate columns.

`sort_values()` method sorts the DataFrame by the specified label which sorts a data frame in Ascending or Descending order of passed Column.

The project was done by using above mentioned methods and functions. In the Tidy Data Process, the detailed steps of cleansing of Table 1 will be shown. The similar steps will be applied for the other 5 tables. The results will be listed in **Appendix,** and the differences and comparison will be discussed in the **Discussion** section.

Tidy Data Process

Table 1 - International Migrant Stock at mid-year by sex and by major area, region, country, or area, 1990-2015

Step 1. Import panadas package, define column name, and load in excel table

Load in Python packages in **Google Colab** is the starting point of table analysis, `import pandas as pd`. Insert in Excel table *(Figure1. Load in Excel Table)* and defining column name (*Figure 2. Define Column Names)* is a necessary and first step and will be processed in each table analysis.

Figure 1. Load in Excel Table

```
# load in excel data table
data1 = pd.read_excel('UN_MigrantStockTotal_2015.xlsx','Table 1',header=15)
data1.columns = li1
data1
```

Figure 2. Define Column Name

```
# define column names
li1 = ['Sort_order','Major_area','Notes','Country_code','Type_of_data_(a)',
                 'sexes_1990','sexes_1995','sexes_2000','sexes_2005','sexes_2010','sexes_2015',
                 'male_1990','male_1995','male_2000','male_2005','male_2010','male_2015',
                 'female_1990','female_1995','female_2000','female_2005','female_2010','female_
```

Table 1. Original Data of Table 1

	Sort_order	Major_area	Notes	Country_code	Type_of_data_(a)	sexes_1990	sexes_1995	sexes
0	1	WORLD	NaN	900	NaN	152563212	160801752	1727
1	2	Developed regions	(b)	901	NaN	82378628	92306854	1033
2	3	Developing regions	(c)	902	NaN	70184584	68494898	693
3	4	Least developed countries	(d)	941	NaN	11075966	11711703	100
4	5	Less developed regions excluding least develop...	NaN	934	NaN	59105261	56778501	592
...	...	...	...	...	...	...	...	
260	261	Samoa	NaN	882	B	3357	4694	
261	262	Tokelau	NaN	772	B	270	266	
262	263	Tonga	NaN	776	B	2911	3274	
263	264	Tuvalu	NaN	798	C	318	263	
264	265	Wallis and Futuna Islands	NaN	876	B	1402	1680	

265 rows × 23 columns

Step 2. Obtain brief summary of the data

Table 2 Summary of Data1

```
# check the dtypes of the data frame column
data1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 265 entries, 0 to 264
Data columns (total 23 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   Sort_order        265 non-null    int64
 1   Major_area        265 non-null    object
 2   Notes             26 non-null     object
 3   Country_code      265 non-null    int64
 4   Type_of_data_(a)  232 non-null    object
 5   sexes_1990        265 non-null    object
 6   sexes_1995        265 non-null    object
 7   sexes_2000        265 non-null    object
 8   sexes_2005        265 non-null    object
 9   sexes_2010        265 non-null    int64
 10  sexes_2015        265 non-null    int64
 11  male_1990         265 non-null    object
 12  male_1995         265 non-null    object
 13  male_2000         265 non-null    object
 14  male_2005         265 non-null    object
 15  male_2010         265 non-null    int64
 16  male_2015         265 non-null    int64
 17  female_1990       265 non-null    object
 18  female_1995       265 non-null    object
 19  female_2000       265 non-null    object
 20  female_2005       265 non-null    object
 21  female_2010       265 non-null    int64
 22  female_2015       265 non-null    int64
dtypes: int64(8), object(15)
memory usage: 47.7+ KB
```

`data1.info ()` prints out the brief summary (*Table 2. Summary of Data1*) of Table 1. This can provide a quick overview of the dataset, including 265 entries, 23 columns, 23 column labels, column data types, and the number of non-null values.

Step 3. Split dataset

Table 1 has data for both sexes, male-only and female-only, in different years. According to *Tidy Data Principle 5, a single observational unit* must be in one table; we should split the data into three tables and tidy up each dataset. Split these different observational units for each to have its own table. Each table should have columns of the major area, country code, data type, sex and year.

Take both sexes as an instant. The following code indicates how to split out the data of both sexes *(Table 3. Date for Both Sexes after Splitting)* from the original table.

Table 3. Data for Both Sexes after Splitting

```
# principle 5: a single observational units must be in one table
# split the dataset to have separate tables for "sexes","male", and "female".
# the 3 data sets will have "Major_area","Country_code","Type_of_data"
sexes = data1[['Major_area','Country_code','Type_of_data_(a)','sexes_1990','sexes_1995','sexes_2000','sexes_2005','sexes_2010','
sexes
```

	Major_area	Country_code	Type_of_data_(a)	sexes_1990	sexes_1995	sexes_2000	sexes_2005	sexes_2010	sexes_2015
0	WORLD	900	NaN	152563212	160801752	172703309	191269100	221714243	243700236
1	Developed regions	901	NaN	82378628	92306854	103375363	117181109	132560325	140481955
2	Developing regions	902	NaN	70184584	68494898	69327946	74087991	89153918	103218281
3	Least developed countries	941	NaN	11075966	11711703	10077824	9809634	10018128	11951316
4	Less developed regions excluding least develop...	934	NaN	59105261	56778501	59244124	64272611	79130668	91262036
...	...	...	...	...	...	...	...	...	...
260	Samoa	882	B	3357	4694	5998	5746	5122	4929
261	Tokelau	772	B	270	266	262	258	429	487
262	Tonga	776	B	2911	3274	3684	4301	5022	5731
263	Tuvalu	798	C	318	263	217	183	154	141
264	Wallis and Futuna Islands	876	B	1402	1680	2015	2365	2776	2849

265 rows × 9 columns

Splitting out male-only and female-only followed the same procedures. This step simplified the dataframe from 265 entities, 23 columns, and 23 column labels to three dataframes with 265 entries, 9 columns. Dataframes of male-only and female-only now looks as shown below.

Table 2. Data for Male-only

	Major_area	Country_code	Type_of_data_(a)	male_1990	male_1995	male_2000
0	WORLD	900	NaN	77747510	81737477	87884839
1	Developed regions	901	NaN	40263397	45092799	50536796
2	Developing regions	902	NaN	37484113	36644678	37348043
3	Least developed countries	941	NaN	5843107	6142712	5361902
4	Less developed regions excluding least develop...	934	NaN	31641006	30501966	31986141
...	...	...	...	...	...	...
260	Samoa	882	B	1771	2451	3101
261	Tokelau	772	B	150	147	144
262	Tonga	776	B	1488	1718	1981
263	Tuvalu	798	C	180	148	121
264	Wallis and Futuna Islands	876	B	726	859	1018

265 rows × 9 columns

Table 3. Data for Female-only

```
#drop rows with missing values
tidy_male.dropna(subset=['Type_of_data_(a)'],inplace=True)
tidy_male
```

	Major_area	Country_code	Type_of_data_(a)	count	sex	year
8	Burundi	108	B R	163267	male	1990
9	Comoros	174	B	6717	male	1990
10	Djibouti	262	B R	64242	male	1990
11	Eritrea	232	I	6228	male	1990
12	Ethiopia	231	B R	607284	male	1990
...	...	...	...	...	...	...
1585	Samoa	882	B	2469	male	2015
1586	Tokelau	772	B	233	male	2015
1587	Tonga	776	B	3127	male	2015
1588	Tuvalu	798	C	78	male	2015
1589	Wallis and Futuna Islands	876	B	1438	male	2015

1377 rows × 6 columns

Step 4. Reshaping table format

Based on the one *Tidy Data Principle 1*. column names need to be informative, variable names are not values, reshaping the table from wide format to long format using `melt ()`. Both sexes in the years of 1990, 1995, 2000, 2005, 2010, and 2015 are in the `"sexes_year"` column, and their previous values are stored in `"count"` column. This step further simplifies the table for both sexes, male-only, and female-only to 1590 entries, 5 columns, and 5column labels.

Table 4. Data of Both Sexes after Melt

```
# principle 1: Some column headers are values, not variable names
tidy_sexes = sexes.melt(id_vars = ['Major_area','Country_code','Type_of_data_(a)'],
                        var_name = "sexes_year",value_name= "count")
tidy_sexes
```

	Major_area	Country_code	Type_of_data_(a)	sexes_year	count
0	WORLD	900	NaN	sexes_1990	152563212
1	Developed regions	901	NaN	sexes_1990	82378628
2	Developing regions	902	NaN	sexes_1990	70184584
3	Least developed countries	941	NaN	sexes_1990	11075966
4	Less developed regions excluding least develop...	934	NaN	sexes_1990	59105261
...	...	...	...	...	...
1585	Samoa	882	B	sexes_2015	4929
1586	Tokelau	772	B	sexes_2015	487
1587	Tonga	776	B	sexes_2015	5731
1588	Tuvalu	798	C	sexes_2015	141
1589	Wallis and Futuna Islands	876	B	sexes_2015	2849

1590 rows × 5 columns

Table 5.Data of Male-only after Melt

```
tidy_male = male.melt(id_vars = ['Major_area','Country_code','Type_of_data_(a)'],
                      var_name = "male_year",value_name= "count")
tidy_male
```

	Major_area	Country_code	Type_of_data_(a)	male_year	count
0	WORLD	900	NaN	male_1990	77747510
1	Developed regions	901	NaN	male_1990	40263397
2	Developing regions	902	NaN	male_1990	37484113
3	Least developed countries	941	NaN	male_1990	5843107
4	Less developed regions excluding least develop...	934	NaN	male_1990	31641006
...	...	...	...	...	...
1585	Samoa	882	B	male_2015	2469
1586	Tokelau	772	B	male_2015	233
1587	Tonga	776	B	male_2015	3127
1588	Tuvalu	798	C	male_2015	78
1589	Wallis and Futuna Islands	876	B	male_2015	1438

1590 rows × 5 columns

Table 6.Data of Female-only after Melt

```
tidy_female = female.melt(id_vars = ['Major_area','Country_code','Type_of_data_(a)'],
                          var_name = "female_year",value_name= "count")
tidy_female
```

	Major_area	Country_code	Type_of_data_(a)	female_year	count
0	WORLD	900	NaN	female_1990	74815702
1	Developed regions	901	NaN	female_1990	42115231
2	Developing regions	902	NaN	female_1990	32700471
3	Least developed countries	941	NaN	female_1990	5236216
4	Less developed regions excluding least develop...	934	NaN	female_1990	27464255
...	...	...	...	...	...
1585	Samoa	882	B	female_2015	2460
1586	Tokelau	772	B	female_2015	254
1587	Tonga	776	B	female_2015	2604
1588	Tuvalu	798	C	female_2015	63
1589	Wallis and Futuna Islands	876	B	female_2015	1411

1590 rows × 5 columns

Step 5. Splitting 'sex' and 'year'

Based on Tidy Data Principle 2. each column needs to consist of one and only one variable, `'sex'` and `'year'` should be separated into two columns by `str.split ()` function and drop '`sex_year`' column by `data.drop()` after.

Table 8. Data of Both Sexes after Splitting 'sex' and 'year'

```
#principle 2. each column needs to consist of one and only one varibale
#split sexes and year
tidy_sexes[['sex','year']]=tidy_sexes.sexes_year.str.split("_",expand=True)
tidy_sexes.drop(columns=['sexes_year'],axis=1,inplace=True)
tidy_sexes
```

	Major_area	Country_code	Type_of_data_(a)	count	sex	year
0	WORLD	900	NaN	152563212	sexes	1990
1	Developed regions	901	NaN	82378628	sexes	1990
2	Developing regions	902	NaN	70184584	sexes	1990
3	Least developed countries	941	NaN	11075966	sexes	1990
4	Less developed regions excluding least develop...	934	NaN	59105261	sexes	1990
...	...	...	...	...	...	...
1585	Samoa	882	B	4929	sexes	2015
1586	Tokelau	772	B	487	sexes	2015
1587	Tonga	776	B	5731	sexes	2015
1588	Tuvalu	798	C	141	sexes	2015
1589	Wallis and Futuna Islands	876	B	2849	sexes	2015

1590 rows × 6 columns

Table 7.Data of Male-only after Splitting 'sex' and 'year'

```
#split sexes and year
tidy_male[['sex','year']]=tidy_male.male_year.str.split("_",expand=True)
tidy_male.drop(columns=['male_year'],axis=1,inplace=True)
tidy_male
```

	Major_area	Country_code	Type_of_data_(a)	count	sex	year
0	WORLD	900	NaN	77747510	male	1990
1	Developed regions	901	NaN	40263397	male	1990
2	Developing regions	902	NaN	37484113	male	1990
3	Least developed countries	941	NaN	5843107	male	1990
4	Less developed regions excluding least develop...	934	NaN	31641006	male	1990
...	...	...	...	...	...	...
1585	Samoa	882	B	2469	male	2015
1586	Tokelau	772	B	233	male	2015
1587	Tonga	776	B	3127	male	2015
1588	Tuvalu	798	C	78	male	2015
1589	Wallis and Futuna Islands	876	B	1438	male	2015

1590 rows × 6 columns

Table 9Data of female-only after Splitting 'sex' and 'year'

	Major_area	Country_code	Type_of_data_(a)	count	sex	year
0	WORLD	900	NaN	74815702	female	1990
1	Developed regions	901	NaN	42115231	female	1990
2	Developing regions	902	NaN	32700471	female	1990
3	Least developed countries	941	NaN	5236216	female	1990
4	Less developed regions excluding least develop...	934	NaN	27464255	female	1990
...	...	...	...	...	...	...
1585	Samoa	882	B	2460	female	2015
1586	Tokelau	772	B	254	female	2015
1587	Tonga	776	B	2604	female	2015
1588	Tuvalu	798	C	63	female	2015
1589	Wallis and Futuna Islands	876	B	1411	female	2015

1590 rows × 6 columns

Step 6. Removing rows with `'..'` in `'Count'` Column.

When first checking table 1; it is easily found there are some values `'..'` in the 'Count' column. Values `'..'` can be removed by `drop()` function and in an iterative way. Now, the cleaned table for both sexes has 1575 entries and 6 columns.

Table 10 - Removing '..' Value for both sexes

```
for col in ['count']:
    tidy_sexes.drop(tidy_sexes[tidy_sexes[col] == '..'].index, inplace=True)
tidy_sexes
```

	Major_area	Country_code	Type_of_data_(a)	count	sex	year
0	WORLD	900	NaN	152563212	sexes	1990
1	Developed regions	901	NaN	82378628	sexes	1990
2	Developing regions	902	NaN	70184584	sexes	1990
3	Least developed countries	941	NaN	11075966	sexes	1990
4	Less developed regions excluding least develop...	934	NaN	59105261	sexes	1990
...	...	...	...	...	...	...
1585	Samoa	882	B	4929	sexes	2015
1586	Tokelau	772	B	487	sexes	2015
1587	Tonga	776	B	5731	sexes	2015
1588	Tuvalu	798	C	141	sexes	2015
1589	Wallis and Futuna Islands	876	B	2849	sexes	2015

1575 rows × 6 columns

Table 11 Removing '..' Value for male-only

```
for col in ['count']:
    tidy_male.drop(tidy_male[tidy_male[col] == '..'].index, inplace=True)
tidy_male
```

	Major_area	Country_code	Type_of_data_(a)	count	sex	year
0	WORLD	900	NaN	77747510	male	1990
1	Developed regions	901	NaN	40263397	male	1990
2	Developing regions	902	NaN	37484113	male	1990
3	Least developed countries	941	NaN	5843107	male	1990
4	Less developed regions excluding least develop...	934	NaN	31641006	male	1990
...	...	...	...	...	...	...
1585	Samoa	882	B	2469	male	2015
1586	Tokelau	772	B	233	male	2015
1587	Tonga	776	B	3127	male	2015
1588	Tuvalu	798	C	78	male	2015
1589	Wallis and Futuna Islands	876	B	1438	male	2015

1575 rows × 6 columns

Table 12 Removing '..' Value for female-only

```
for col in ['count']:
    tidy_female.drop(tidy_female[tidy_female[col] == '..'].index, inplace=True)
tidy_female
```

	Major_area	Country_code	Type_of_data_(a)	count	sex	year
0	WORLD	900	NaN	74815702	female	1990
1	Developed regions	901	NaN	42115231	female	1990
2	Developing regions	902	NaN	32700471	female	1990
3	Least developed countries	941	NaN	5236216	female	1990
4	Less developed regions excluding least develop...	934	NaN	27464255	female	1990
...	...	...	...	...	...	...
1585	Samoa	882	B	2460	female	2015
1586	Tokelau	772	B	254	female	2015
1587	Tonga	776	B	2604	female	2015
1588	Tuvalu	798	C	63	female	2015
1589	Wallis and Futuna Islands	876	B	1411	female	2015

1575 rows × 6 columns

Step 7. Drop rows which has missing values

By checking the brief summary, it is worth noticing that the 'Type _ of _ data _ (a)' column is partially missing data; thus, using `tidy_sexes.dropna()` function to remove the missing values in rows of 'Type _ of _ data _ (a)'. By doing so, both sexes entries were brought down to 1377.

Table 13. Removing rows with missing value for both sexes

```
#drop rows with missing values
tidy_sexes.dropna(subset=['Type_of_data_(a)'],inplace=True)
tidy_sexes
```

	Major_area	Country_code	Type_of_data_(a)	count	sex	year
8	Burundi	108	B R	333110	sexes	1990
9	Comoros	174	B	14079	sexes	1990
10	Djibouti	262	B R	122221	sexes	1990
11	Eritrea	232	I	11848	sexes	1990
12	Ethiopia	231	B R	1155390	sexes	1990
...	...	...	...	...	...	...
1585	Samoa	882	B	4929	sexes	2015
1586	Tokelau	772	B	487	sexes	2015
1587	Tonga	776	B	5731	sexes	2015
1588	Tuvalu	798	C	141	sexes	2015
1589	Wallis and Futuna Islands	876	B	2849	sexes	2015

1377 rows × 6 columns

Table 14 Removing rows with missing value for male-only

```
#drop rows with missing values
tidy_male.dropna(subset=['Type_of_data_(a)'],inplace=True)
tidy_male
```

	Major_area	Country_code	Type_of_data_(a)	count	sex	year
8	Burundi	108	B R	163267	male	1990
9	Comoros	174	B	6717	male	1990
10	Djibouti	262	B R	64242	male	1990
11	Eritrea	232	I	6228	male	1990
12	Ethiopia	231	B R	607284	male	1990
...	...	...	...	...	...	...
1585	Samoa	882	B	2469	male	2015
1586	Tokelau	772	B	233	male	2015
1587	Tonga	776	B	3127	male	2015
1588	Tuvalu	798	C	78	male	2015
1589	Wallis and Futuna Islands	876	B	1438	male	2015

1377 rows × 6 columns

Table 15 Removing rows with missing value for female-only

```
#drop rows with missing values
tidy_female.dropna(subset=['Type_of_data_(a)'],inplace=True)
tidy_female
```

	Major_area	Country_code	Type_of_data_(a)	count	sex	year
8	Burundi	108	B R	169843	female	1990
9	Comoros	174	B	7362	female	1990
10	Djibouti	262	B R	57979	female	1990
11	Eritrea	232	I	5620	female	1990
12	Ethiopia	231	B R	548106	female	1990
...	...	...	...	...	...	...
1585	Samoa	882	B	2460	female	2015
1586	Tokelau	772	B	254	female	2015
1587	Tonga	776	B	2604	female	2015
1588	Tuvalu	798	C	63	female	2015
1589	Wallis and Futuna Islands	876	B	1411	female	2015

1377 rows × 6 columns

By applying all the above methods, three cleaned tables were generated finally.

Discussion

In general, there are three issues were explored in these tables while cleaning:

- Multiple types of observational units are stored in the same table.
- Some column headers are values, not variable names.
- Multiple variables are stored in one column.

Multiple types of observational units are stored in the same table

To deal with the problem of multiple types of observational units being stored in the same table required splitting these different observational units for each to have its own table. This would be achieved by splitting the original table into three dataframes which are both sexes, male-only, and female-only.

Some column headers are value, not variable names

Sex in the different years is value but is used as column headers. This is fixed by the `melt()` function to create a '`sex_year`' column and an extra column for the '`count`'.

Multiple variables are stored in one column

The 'sex_year' column stores both sex and year. The sex and year for each row are extracted and added to the relevant column. The original sex and year column is dropped from the dataframe. This step is also achieved by str.split() function and drop() function.

The steps of cleaning Table 1 were indicated in **Methods and Results**. The way of dealing with the other five tables is slightly different. The method used in Table 1 is first to split the dataframe into three separate tables, both sexes, male-only, and female-only, then clean each one-by-one using the same procedure. In this way, a single observational unit is in one table. However, in dealing with others, this step was omitted, which caused to obtain a long table format.

Besides, in the beginning of the cleaning, step 7 was added which is to drop columns that are superfluous data and rows has missing data. Looking at the columns one-by-one, it was easily found that 'Notes' and 'Sort_order' are superfluous columns. Thus, removing those two columns by drop(columns=) function.

Table 16 Table 2 after dropping superfluous columns

```
data2.drop(columns=['Notes','Sort_order'],axis=1,inplace=True)
data2
```

	Major_area	Country_code	sexes_1990	sexes_1995	sexes_2000	sexes_2005	sexes_2010	sexes_2015	male_1990
0	WORLD	900	5309667.699	5735123.084	6126622.121	6519635.850	6929725.043	7349472.099	2670423.701
1	Developed regions	901	1144463.062	1169761.211	1188811.731	1208919.509	1233375.711	1251351.086	555255.626
2	Developing regions	902	4165204.637	4565361.873	4937810.390	5310716.341	5696349.332	6098121.013	2115168.075
3	Least developed countries	941	510057.629	585189.354	664386.087	752804.951	847254.847	954157.804	254042.556
4	Less developed regions excluding least develop...	934	3655147.008	3980172.519	4273424.303	4557911.390	4849094.485	5143963.209	1861125.519
...	...	...	...	...	...	...	...	...	...
260	Samoa	882	162.865	170.158	174.614	179.928	186.029	193.228	85.009
261	Tokelau	772	1.609	1.520	1.552	1.210	1.135	1.250	..
262	Tonga	776	95.152	95.889	97.898	100.858	103.947	106.170	48.247
263	Tuvalu	798	9.004	9.227	9.419	9.694	9.827	9.916	..
264	Wallis and Futuna Islands	876	13.880	14.143	14.497	14.246	13.565	13.151	..

When dealing with Table 3, drop rows of 'Type _ of _ data _ (a)' which has missing value followed by dropping conlumns. data.dropna() function to remove the missing values in rows of 'Type _ of _ data _ (a)'.

Table 17 Removing rows in Table 3 which has missing value

```
data3.dropna(subset=['Type_of_data_(a)'],inplace=True)
data3
```

	Major_area	Country_code	Type_of_data_(a)	sexes_1990	sexes_1995	sexes_2000	sexes_2005
8	Burundi	108	B R	5.934467	4.084818	1.85646	2.178842
9	Comoros	174	B	3.391353	2.906538	2.519463	2.135195
10	Djibouti	262	B R	20.773307	15.092667	13.90981	11.830716
11	Eritrea	232	I	0.377435	0.391897	0.366377	0.341519
12	Ethiopia	231	B R	2.404203	1.409754	0.920155	0.67126
...	...	...	...	...	...	...	...
260	Samoa	882	B	2.061216	2.758613	3.435005	3.1935
261	Tokelau	772	B	16.780609	17.5	16.881443	21.322314
262	Tonga	776	B	3.059316	3.414365	3.7631	4.264411
263	Tuvalu	798	C	3.531764	2.850331	2.303854	1.887766
264	Wallis and Futuna Islands	876	B	10.100865	11.878668	13.899427	16.601151

232 rows × 21 columns

When cleaning Table 4, there is only one observation in this table, female only, which means it does not have the issue of multiple types of observational units stored in the same table. So, Step 3 in **Methods and Results** can be omitted in Table 4. In addition, `sort_value()` function was applied to sort the '`percentage`' in ascending which makes the table more clear.

Table 18. Table 4 Applied sort_value function

```
tidy_data4.sort_values(by=['percentage'], inplace=True)
tidy_data4
```

	Major_area	Country_code	Type_of_data_(a)	percentage	sex	year
1221	Bangladesh	50	B R	13.325719	female	2015
993	Bangladesh	50	B R	13.458551	female	2010
765	Bangladesh	50	B R	13.628353	female	2005
81	Bangladesh	50	B R	13.856924	female	1990
309	Bangladesh	50	B R	13.858366	female	1995
...	...	...	...	...	...	...
542	Nepal	524	B R	66.296281	female	2000
998	Nepal	524	B R	67.248646	female	2010
314	Nepal	524	B R	68.548227	female	1995
1226	Nepal	524	B R	68.96434	female	2015
86	Nepal	524	B R	70.70381	female	1990

Although successfully figuring out and cleaning three issues in this project; it is still not perfect. Here is one thing worth noticing, which is the sequence of the columns needs to be in the right order. The value column should be placed at the last instead of in the mid; this problem caused by `str.split()`. After spliting 'sex_year' column into 'sex' column and 'year' column, the value column moved into the middle and the way of making all the column in a right order haven't found.

Conclusion

This mid-term project is a good practice for students to fully understand that the tidy data has the following properties:

1. Each variable forms a column.
2. Each observation forms a row.
3. Each type of observational unit forms a table.

Extracting meaningful information may help to tidy data. When data is extracted directly from databases, the columns of tables hold special meaning to analysis. In such cases, they become actual values and not variables. Other times the columns contain more than one kind of information, which needs to be separated. Each analysis may involve more than one tidy data principle that requires carefully observed and treated differently.

Reference

Wickham, H. (2014). Tidy Data. *The Journal of Statistical Software*.

Appendix

Table 2 - Total population at mid-year by sex and by major area, region, country or area, 1990-2015 (thousands)

	Major_area	Country_code	population	sex	year
0	WORLD	900	5309667.699	sexes	1990
1	Developed regions	901	1144463.062	sexes	1990
2	Developing regions	902	4165204.637	sexes	1990
3	Least developed countries	941	510057.629	sexes	1990
4	Less developed regions excluding least develop...	934	3655147.008	sexes	1990
...	...	...	...	...	...
4756	Micronesia (Federated States of)	583	50.95	female	2015
4760	Polynesia	957	336.115	female	2015
4763	French Polynesia	258	138.468	female	2015
4765	Samoa	882	93.584	female	2015
4767	Tonga	776	52.931	female	2015

4386 rows × 5 columns

```
tidy_data2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 4386 entries, 0 to 4767
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype 
---  ------        --------------  ----- 
 0   Major_area    4386 non-null   object
 1   Country_code  4386 non-null   int64 
 2   population    4386 non-null   object
 3   sex           4386 non-null   object
 4   year          4386 non-null   object
dtypes: int64(1), object(4)
memory usage: 205.6+ KB
```

Table 3 - International migrant stock as a percentage of the total population by sex and by major area, region, country or area, 1990-2015

	Major_area	Country_code	Type_of_data_(a)	population	sex	year
0	Burundi	108	B R	5.934467	sexes	1990
1	Comoros	174	B	3.391353	sexes	1990
2	Djibouti	262	B R	20.773307	sexes	1990
3	Eritrea	232	I	0.377435	sexes	1990
4	Ethiopia	231	B R	2.404203	sexes	1990
...	...	...	...	...	...	...
4161	Kiribati	296	B	2.615835	female	2015
4163	Micronesia (Federated States of)	583	B	2.518155	female	2015
4169	French Polynesia	258	B	9.33212	female	2015
4171	Samoa	882	B	2.628654	female	2015
4173	Tonga	776	B	4.919612	female	2015

3749 rows × 6 columns

```
tidy_data3.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 3749 entries, 0 to 4173
Data columns (total 6 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   Major_area        3749 non-null   object
 1   Country_code      3749 non-null   int64
 2   Type_of_data_(a)  3749 non-null   object
 3   population        3749 non-null   object
 4   sex               3749 non-null   object
 5   year              3749 non-null   object
dtypes: int64(1), object(5)
memory usage: 205.0+ KB
```

Table 4 - Female migrants as a percentage of the international migrant stock by major area, region, country or area, 1990-2015

	Major_area	Country_code	Type_of_data_(a)	percentage	sex	year
1221	Bangladesh	50	B R	13.325719	female	2015
993	Bangladesh	50	B R	13.458551	female	2010
765	Bangladesh	50	B R	13.628353	female	2005
81	Bangladesh	50	B R	13.856924	female	1990
309	Bangladesh	50	B R	13.858366	female	1995
...	...	...	...	...	...	...
542	Nepal	524	B R	66.296281	female	2000
998	Nepal	524	B R	67.248646	female	2010
314	Nepal	524	B R	68.548227	female	1995
1226	Nepal	524	B R	68.96434	female	2015
86	Nepal	524	B R	70.70381	female	1990

1368 rows × 6 columns

```
tidy_data4.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1368 entries, 1221 to 86
Data columns (total 6 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   Major_area        1368 non-null   object
 1   Country_code      1368 non-null   int64
 2   Type_of_data_(a)  1368 non-null   object
 3   percentage        1368 non-null   object
 4   sex               1368 non-null   object
 5   year              1368 non-null   object
dtypes: int64(1), object(5)
memory usage: 74.8+ KB
```

Table 5 - Annual rate of change of the migrant stock by sex and by major area, region, country or area, 1990-2015 (percentage)

	Major_area	Country_code	Type_of_data_(a)	percentage	sex	year
2632	Somalia	706	I R	-64.632712	female	1990
22	Somalia	706	I R	-63.96855	sexes	1990
1327	Somalia	706	I R	-63.351937	male	1990
3079	Honduras	340	B R	-33.299961	female	1995
469	Honduras	340	B R	-33.167473	sexes	1995
...	...	...	...	...	...	...
3163	Chad	148	B R	27.611733	female	2000
3751	Afghanistan	4	B	28.900067	female	2010
1468	Serbia	688	B	36.486193	male	1990
163	Serbia	688	B	36.964744	sexes	1990
2773	Serbia	688	B	37.380672	female	1990

3420 rows × 6 columns

```
tidy_data5.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 3420 entries, 2632 to 2773
Data columns (total 6 columns):
 #   Column            Non-Null Count  Dtype 
---  ------            --------------  ----- 
 0   Major_area        3420 non-null   object
 1   Country_code      3420 non-null   int64 
 2   Type_of_data_(a)  3420 non-null   object
 3   percentage        3420 non-null   object
 4   sex               3420 non-null   object
 5   year              3420 non-null   object
dtypes: int64(1), object(5)
memory usage: 187.0+ KB
```

Table 6 - Estimated refugee stock at mid-year by major area, region, country or area, 1990-2015.

	Major_area	Country_code	Type_of_data_(a)	count	sex	year
1307	Albania	8	B	13.605596	male	2015
523	Albania	8	B	99	sexes	2010
1755	Albania	8	B	5.768321	female	2005
635	Albania	8	B	7839	sexes	2015
1419	Albania	8	B	7.507896	female	1990
...	...	...	...	...	...	...
1802	Zambia	894	B R	-10.118905	female	2010
906	Zambia	894	B R	71.197539	male	2000
122	Zambia	894	B R	129965	sexes	1995
346	Zambia	894	B R	155718	sexes	2005
234	Zambia	894	B R	228663	sexes	2000

1904 rows × 6 columns

```
tidy_data6.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1904 entries, 1307 to 234
Data columns (total 6 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   Major_area        1904 non-null   object
 1   Country_code      1904 non-null   int64
 2   Type_of_data_(a)  1904 non-null   object
 3   count             1904 non-null   object
 4   sex               1904 non-null   object
 5   year              1904 non-null   object
dtypes: int64(1), object(5)
memory usage: 104.1+ KB
```