Submission Details

Omer Imran (1008712190)

INF 1340 – Midterm Write Up

Google Colab

Open it here.

Introduction

Before diving into the content itself, I will break down this document. The Colab link is above for you to have a look at the jupyter file.

We will first take a short look at the Tidy Data principles that we have discussed in class (Chapter 1). Then we dive into the common violations of the principles (Chapter 2) followed up by the procedure of how I cleaned the dataset (Chapter 3). Lastly, we'll dive into the tables that we did not change as well as the logic of dropping some rows (Chapter 4).

Table of Contents

Chapter 1: Tidy Data Principles

There are three fundamentals principles of Tidy data (Wickham). They are the following,

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

If these principles are upheld, then we have data that is machine-readable and exactly what we need for the rest of the data science cycle. Common problems that emerge from messy data are the following according to Wichkham,

1. Column headers are values, not variable names.
2. Multiple variables are stored in one column.
3. Variables are stored in both rows and columns.
4. Multiple types of observational units are stored in the same table.
5. A single observational unit is stored in multiple tables.

With the common problems identified, we will now look at these violations across all the tables.

Chapter 2: Violations

1. *Column headers are values, not variable names*

We will only be looking at the violations for table 1 to 6. The rest will not be investigated. For this, refer to chapter 4. We see frequent violations of this kind in our dataset – a sample of this is present below,

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)					
					1990	1995	2000	2005	2010	2015
1	WORLD		900		#########	#########	#########	#########	#########	#########
2	Developed regions	(b)	901		82 378 628	92 306 854	#########	#########	#########	#########
3	Developing regions	(c)	902		70 184 584	68 494 898	69 327 946	74 087 991	89 153 918	#########
4	Least developed countries	(d)	941		11 075 966	11 711 703	10 077 824	9 809 634	10 018 128	11 951 316
5	Less developed regions excluding least developed countries		934		59 105 261	56 778 501	59 244 124	64 272 611	79 130 668	91 262 036
6	Sub-Saharan Africa	(e)	947		14 690 319	15 324 570	13 716 539	13 951 086	15 496 764	18 993 986
7	Africa		903		15 690 623	16 352 814	14 800 306	15 191 146	16 840 014	20 649 557
8	Eastern Africa		910		5 964 031	5 022 742	4 844 795	4 745 792	4 657 063	6 129 113
9	Burundi		108	B R	333 110	254 853	125 628	172 874	235 259	286 810

Fig 1: A snippet of Table 1

Just in table 1 we see a handful of these infractions: the migrant stock is broken down into multiple columns with years representing each column. These violations are present across the Tables. The solution for this is to melt all these values into a single column. This will result in compliance to principle #1.

2. *Multiple variables are stored in one column*

The first instance of this is the "*Type of data (a)*" column present in all tables except Table 2.

Fig 2: A snippet of "*Type of data (a)*" column

This is a single column storing whether the migrant/refugee stock measurement from each table was of foreign born population or foreign citizens and if refugee stock was imputed as well as if international migrants were added to the stock or not. This has to be split into multiple binary columns in accordance to principle #2 of Tidy Data. More on this in the next chapter.

3. *Variables are stored in both rows and columns.*

We see a violation of this in every table.

7	**Africa**		**903**		**5 687 352**
8	**Eastern Africa**		**910**		**3 168 001**
9	Burundi		108	B R	267 929
10	Comoros		174	B	0
11	Djibouti		262	B R	54 508
12	Eritrea		232	I	0
13	Ethiopia		231	B R	741 965

Fig 3: Africa and Eastern Africa are variables in rows

In every table, we have major regions and regions in our "*Major area, region, country, or area of destination*" column. These regions act as aggregations of countries in said region. This is incorrect since it indicates these regions as variables stored in rows and violates principle #2 since Africa is not a unique observation – it's only a sum of African countries.

4. *Multiple types of observational units are stored in the same table.*

This issue is present solely in Table 6. It has three different units of measurement: refugee stock in years, refugees as a percentage of migrant stock in years, and annual rate of change of refugee stock in time periods.

Estimated refugee stock at mid-year (both sexes)						Refugees as a percentage of the international migrant stock						Annual rate of change of the refugee stock				
1990	1995	2000	2005	2010	2015	1990	1995	2000	2005	2010	2015	1990-1995	1995-2000	2000-2005	2005-2010	2010-2015
##########	##########	##########	##########	##########	##########	12.3	11.1	9.2	6.9	6.9	8.0	-2.12	-3.84	-5.56	-0.03	2.95
2 014 564	3 609 670	2 997 256	2 361 229	2 046 917	1 954 224	2.4	3.9	2.9	2.0	1.5	1.4	9.39	-5.98	-7.28	-5.32	-2.09
##########	##########	##########	##########	##########	##########	24.0	20.8	18.5	14.7	14.9	17.1	-2.84	-2.33	-4.56	0.29	2.66
5 048 391	5 160 131	3 047 488	2 363 782	1 957 884	3 443 582	45.6	44.0	30.2	24.1	19.5	28.8	-0.68	-7.53	-4.54	-4.19	7.77
##########	9 084 039	9 783 059	8 551 722	##########	##########	19.9	16.0	16.5	13.3	14.4	15.5	-4.38	0.63	-4.32	1.53	1.57
5 516 042	5 747 830	3 421 165	2 555 099	2 215 890	3 638 433	37.5	37.5	24.9	18.3	14.3	19.2	-0.02	-8.16	-6.18	-4.95	5.85
5 687 352	5 949 953	3 609 138	2 750 644	2 413 917	4 022 363	36.2	36.4	24.4	18.1	14.3	19.5	0.08	-8.00	-5.95	-4.67	6.13
3 168 001	2 046 088	1 641 559	1 419 685	1 008 358	2 087 514	53.1	40.7	33.9	29.9	21.7	34.1	-5.31	-3.68	-2.49	-6.46	9.06

Fig 4: Columns in table 6 having different units

In accordance to principle #3 of Tidy data, we will split this into three separate tables.

Lastly, the common problem of "*A single observational unit is stored in multiple tables*" was not found.

Chapter 3: Data Cleaning Steps

As mentioned in the Jupyter notebook, the process of cleaning the tables is very similar for all of them. There are only slight changes in every one of them. I will describe the general process here with any exceptions. For a more detailed look, feel free to jump into the notebook.

1. Imports

Pandas was the only necessary import.

2. Functions

I created a bunch of functions that were repeatedly used throughout the notebook. This helped reduce coding clutter and made the whole process much easier to understand.

2. Functions

Here I write code for any functions that I will be utilizing in later sections of the notebook. Feel free to jump back here to understand how I'm using functions that I create.

In [6]:
```
#The add_Major_region functions below are used to add Major region into a separate column.

def add_Major_region (df):
    cells = range(8,70)
    for i in cells:
        df.iloc[i, 2] = "Africa"
    cells = range(71,126)
    for i in cells:
        df.iloc[i, 2] = "Asia"
    cells = range(127,179)
    for i in cells:
        df.iloc[i, 2] = "Europe"
    cells = range(180,231)
    for i in cells:
        df.iloc[i, 2] = "Latin America and the Caribbean"
    cells = range(232,237)
    for i in cells:
        df.iloc[i, 2] = "Northern America"
    cells = range(238,265)
    for i in cells:
        df.iloc[i, 2] = "Oceania"
```

Fig 5: Functions

3. Loading the dataset

We load the tables only. The rest do not serve any use – check the next section for this for more details.

3.1 Loading the Dataset

```
In [7]: #Loading all the datasets into dataframes.

        #You will need to change this according to your directory
        filename = "/Users/omerimran/Downloads/UN_MigrantStockTotal_2015.xlsx"

        #And this is the format for the colab notebook
        #filename = "/content/UN_MigrantStockTotal_2015.xlsx"

        #The number in-front of df indicates which table it imported. For example df_1 is the dataframe created from table 1
        df_1 = pd.read_excel(filename, "Table 1")
        df_2 = pd.read_excel(filename, "Table 2")
        df_3 = pd.read_excel(filename, "Table 3")
        df_4 = pd.read_excel(filename, "Table 4")
        df_5 = pd.read_excel(filename, "Table 5")
        df_6 = pd.read_excel(filename, "Table 6")
```

Fig 6: Loading Tables 1-6

4. Dropping Text

The first 15 rows only contain text that is not useful for the Tidy Data we are creating. We will drop them altogether.

```
In [9]: #Let's drop the first 15 rows. We've got a lot of cluter there. We'll rename the columns using a fucntion.

        df_1 = df_1.drop(df_1.index[0:15], axis = 0)
        df_1.head(20)
```

Out[9]:

	Unnamed: 0	Unnamed: 1	Unnamed: 2	Unnamed: 3	Unnamed: 4	Unnamed: 5	Unnamed: 6	Unnamed: 7	Unnamed: 8	Unnamed: 9	...	Unnamed: 13	Unnamed: 14	Unnam
15	1	WORLD	NaN	900	NaN	152563212	160801752	172703309	191269100	221714243.0	...	87884839	97866674	1146137
16	2	Developed regions	(b)	901	NaN	82378628	92306854	103375363	117181109	132560325.0	...	50536796	57217777	640810
17	3	Developing regions	(c)	902	NaN	70184584	68494898	69327946	74087991	89153918.0	...	37348043	40648897	505326
18	4	Least developed countries	(d)	941	NaN	11075966	11711703	10077824	9809634	10018128.0	...	5361902	5383009	54627

Fig 7: Dropping the initial text

5. Renaming the columns

Step 4 results in unnamed columns. We give them their requisite names.

```
In [10]: #We will now fix the columns. We need to rename them.

df_1 = df_1.rename({"Unnamed: 0": "Sort order",
                    "Unnamed: 1": "Major area, region, country or area of destination",
                    "Unnamed: 2": "Notes",
                    "Unnamed: 3": "Country code",
                    "Unnamed: 4": "Type of data (a)",
                    "Unnamed: 5": "1990Both",
                    "Unnamed: 6": "1995Both",
                    "Unnamed: 7": "2000Both",
                    "Unnamed: 8": "2005Both",
                    "Unnamed: 9": "2010Both",
                    "Unnamed: 10": "2015Both",
                    "Unnamed: 11": "1990Male",
                    "Unnamed: 12": "1995Male",
                    "Unnamed: 13": "2000Male",
                    "Unnamed: 14": "2005Male",
                    "Unnamed: 15": "2010Male",
                    "Unnamed: 16": "2015Male",
                    "Unnamed: 17": "1990Female",
                    "Unnamed: 18": "1995Female",
                    "Unnamed: 19": "2000Female",
                    "Unnamed: 20": "2005Female",
                    "Unnamed: 21": "2010Female",
                    "Unnamed: 22": "2015Female",
                   }, axis=1)

df_1.head(20)
```

Fig 8: Renaming for Table 1

6. Splitting the Major region, region, area column

Since the column contains so many regions, we will create two new columns that will each contain the major region (continent) and region (sub-region) – and will be named the same too. Two functions copy into respective columns, the major region and region a country is in.

	Sort order	Major area, region, country or area of destination	Major region	Region	Notes	Country code	Type of data (a)	1990Both	1995Both	2000Both	...	2000Male	2005Male	2010Male	2015Male	199(
15	1	WORLD	None	None	NaN	900	NaN	152563212	160801752	172703309	...	87884839	97866674	114613714.0	126115435.0	7
16	2	Developed regions	None	None	(b)	901	NaN	82378628	92306854	103375363	...	50536796	57217777	64081077.0	67618619.0	4
17	3	Developing regions	None	None	(c)	902	NaN	70184584	68494898	69327946	...	37348043	40648897	50532637.0	58496816.0	3
18	4	Least developed countries	None	None	(d)	941	NaN	11075966	11711703	10077824	...	5361902	5383009	5462714.0	6463217.0	
19	5	Less developed regions excluding least develop...	None	None	NaN	934	NaN	59105261	56778501	59244124	...	31986141	35265888	45069923.0	52033599.0	2
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
275	261	Samoa	Oceania	Polynesia	NaN	882	B	3357	4694	5998	...	3101	2940	2594.0	2469.0	
276	262	Tokelau	Oceania	Polynesia	NaN	772	B	270	266	262	...	144	133	206.0	233.0	
277	263	Tonga	Oceania	Polynesia	NaN	776	B	2911	3274	3684	...	1981	2328	2727.0	3127.0	
278	264	Tuvalu	Oceania	Polynesia	NaN	798	C	318	263	217	...	121	101	85.0	78.0	
279	265	Wallis and Futuna Islands	Oceania	Polynesia	NaN	876	B	1402	1680	2015	...	1018	1194	1401.0	1438.0	

Fig 8: Addition of two new variables of Major region and Region

7. Addition of Developing/Developed state

To essentially make sheet "*ANNEX*" redundant, we add a column that tells whether a country is developing or developed according to the "*NOTES*" sheet. Here again we used a function created in the step 2.

```
In [13]: #We are creating a new column "developed/developing"
         df_1.insert(4,'developed/developing', None )

         #This function populates whether a country is developed or developing based on the NOTES section of the dataset
         developing(df_1)
         df_1.head(60)
```

23	9	Burundi	Africa	Eastern Africa	Developing	NaN	108	B R	333110	254853	...	61094	84805	115823.0	14131
24	10	Comoros	Africa	Eastern Africa	Developing	NaN	174	B	14079	13939	...	6511	6286	6060.0	607
25	11	Djibouti	Africa	Eastern Africa	Developing	NaN	262	B R	122221	99774	...	52920	51315	53295.0	5908
26	12	Eritrea	Africa	Eastern Africa	Developing	NaN	232	I	11848	12400	...	6856	7729	8603.0	883
27	13	Ethiopia	Africa	Eastern Africa	Developing	NaN	231	B R	1155390	806904	...	322219	269725	297534.0	54706
28	14	Kenya	Africa	Eastern Africa	Developing	NaN	404	B R	297292	618745	...	348594	391310	459343.0	54164
29	15	Madagascar	Africa	Eastern Africa	Developing	NaN	450	C	23917	21177	...	13276	14744	16410.0	1827
30	16	Malawi	Africa	Eastern Africa	Developing	NaN	454	B R	1127724	241624	...	111530	105931	103869.0	10254

Fig 9: Addition of a new variable of developing/developed

8. Addition of four columns to replace Type of data

We created four more columns as binary variables. They are set to 1 or 0 depending upon the contents of the "*Type of data (a)*" column. For example, if a country's migrant stock is measured by foreign-born population as indicated by a *B*, then the column "*B*" is set to 1.

```
In [14]: #Here we are adding four columns: B, C, R, I according to the "Type of data (a)" column.

df_1.insert(8,'I', 0 )
df_1.insert(8,'R', 0 )
df_1.insert(8,'C', 0 )
df_1.insert(8,'B', 0 )

#This function populates values to B, C, R, I according to the "Type of data (a)" column
type_of_data(df_1)

df_1.head(40)
```

Out[14]:

	Sort order	Major area, region, country or area of destination	Major region	Region	developed/developing	Notes	Country code	Type of data (a)	B	C	...	2000Male	2005Male	2010Male	2015Male	1990Female
15	1	WORLD	None	None	Developing	NaN	900	NaN	0	0	...	87884839	97866674	114613714.0	126115435.0	7481570
16	2	Developed regions	None	None	Developing	(b)	901	NaN	0	0	...	50536796	57217777	64081077.0	67618619.0	4211523
17	3	Developing regions	None	None	Developing	(c)	902	NaN	0	0	...	37348043	40648897	50532637.0	58496816.0	3270047
18	4	Least developed countries	None	None	Developing	(d)	941	NaN	0	0	...	5361902	5383009	5462714.0	6463217.0	523621
19	5	Less developed regions excluding	None	None	Developing	NaN	934	NaN	0	0	...	31986141	35265888	45069923.0	52033599.0	2746425

Fig 10: Addition of columns for replacing Type of data (a)

9. Cleaning up

Before moving onto the next step, I drop the first few rows like World and Developed, the aggregated rows like Africa and Latin America, as well as reindex the table. I also drop the now redundant columns of "*Notes*" and "*Type of data (a)*".

```
In [15]: #Removing reduntant rows of regions and major regions
df_1 = df_1.drop([15+6,15+7,15+28,15+38,15+46,15+52,15+70,15+71,15+77,15+85,15+97,15+107,15+126,15+127,15+138,15+152

#Removing first 6 rows of data as we don't need them
df_1 = df_1.drop([15,16,17,18,19,20], axis = 0)

#Reseting Index and dropping the rest of the index columns
df_1 = df_1.reset_index() #Do we need brackets here
df_1 = df_1.drop(["index", "Sort order"], axis = 1)

#Dropping Notes, Type of data(a)
df_1 = df_1.drop("Notes", axis = 1)
df_1 = df_1.drop("Type of data (a)", axis = 1)

df_1.tail(50)
```

Out[15]:

	Major area, region, country or area of destination	Major region	Region	developed/developing	Country code	B	C	R	I	1990Both	...	2000Male	2005Male	2010Male	2015Male	1990Female
182	Belize	Latin America and the Caribbean	Central America	Developing	84	1	0	1	0	30404	...	18631	21045	23459.0	27092.0	14011
183	Costa Rica	Latin America and the Caribbean	Central America	Developing	188	1	0	1	0	417628	...	156798	176335	195873.0	202613.0	205455

Fig 11: Cleaning up before the next step

10. Melt the unit

Now since the columns are not stored as a single variable, I make the table longer by using the melt function – the same we discussed in class. Now time represents the year and gender (we will fix this in the next step) and migrant stock is our value.

```
In [16]: #Here we melt all the year and migrant stock columns

         df_1 = df_1.melt(id_vars = ["Major area, region, country or area of destination", "Major region", "Region", "develop
         df_1.tail(50)
```

Out[16]:

	Major area, region, country or area of destination	Major region	Region	developed/developing	Country code	B	C	R	I	time	migrant stock
4126	Belize	Latin America and the Caribbean	Central America	Developing	84	1	0	1	0	2015Female	26768.0
4127	Costa Rica	Latin America and the Caribbean	Central America	Developing	188	1	0	1	0	2015Female	219084.0
4128	El Salvador	Latin America and the Caribbean	Central America	Developing	222	1	0	1	0	2015Female	22049.0
4129	Guatemala	Latin America and the Caribbean	Central America	Developing	320	1	0	1	0	2015Female	40216.0

Fig 12: Melt result of Table 1

11. Separating Gender and Year

Using the code taught in class, I separate the gender and year from the time column we created in the last step. This was pre-planned since step 5 of renaming the columns. We're almost done!

```
#Took this code from Shion's Class
#We are separating year and gender from the time column created during melting
df_1=(df_1.assign(gender = lambda x: x.time.str[4:].astype(str), year = lambda x: x.time.str[0:4].astype(str)).drop(
df_1.head(5)
```

	Major area, region, country or area of destination	Major region	Region	developed/developing	Country code	B	C	R	I	migrant stock	gender	year
0	Burundi	Africa	Eastern Africa	Developing	108	1	0	1	0	333110	Both	1990
1	Comoros	Africa	Eastern Africa	Developing	174	1	0	0	0	14079	Both	1990
2	Djibouti	Africa	Eastern Africa	Developing	262	1	0	1	0	122221	Both	1990
3	Eritrea	Africa	Eastern Africa	Developing	232	0	0	0	1	11848	Both	1990
4	Ethiopia	Africa	Eastern Africa	Developing	231	1	0	1	0	1155390	Both	1990

Fig 13: Separating gender and year from the time column

12. The Result

After some more renaming of the columns and rearrangement here is our final result. We have a table of 4176 rows and 12 columns for table 1.

	country	country code	continent	region	development	b	c	r	i	migrant stock	year	gender
0	Burundi	108	Africa	Eastern Africa	Developing	1	0	1	0	333110	1990	Both
1	Comoros	174	Africa	Eastern Africa	Developing	1	0	0	0	14079	1990	Both
2	Djibouti	262	Africa	Eastern Africa	Developing	1	0	1	0	122221	1990	Both
3	Eritrea	232	Africa	Eastern Africa	Developing	0	0	0	1	11848	1990	Both
4	Ethiopia	231	Africa	Eastern Africa	Developing	1	0	1	0	1155390	1990	Both

Fig 14: Final result of Table 1

That's it! The only difference for the rest of the tables are the subtle changes in name. Moreover, table 6 is slightly different since it includes three different observational units and as such we split it into three tables.

13. Download the Result

Once all the tables have been cleaned, the last cell downloads the data into an excel file with each sheet containing the cleaned tables.

Chapter 4: Notes and Assumptions

First of all, we are only going to be looking at the data in tables 1 through 6 in the dataset file. The other sheets provide no useful data for the next steps of our procedure:

- "*CONTENTS*" sheet only describes what information the rest of the sheets hold. It's handy only if you want a bird's eye view of the data.
- "*ANNEX*" sheet functions as a description for each country. The data it describes is already present in tables 1-6 and hence it serves no purpose.
- "*NOTES*" sheet describes which countries are considered developed, developing, and least developed. Again, this information is partly stored. Moreover, the rest of the details are only there to clarify any confusion

regarding contested/disputed territories like Nagorno-Karabakh region in Azerbaijan.

These sheets have not been imported. In case they are needed, you may open the uncleaned dataset for a quick look at what information you need. That is why we dropped the "*Notes*" column in all tables.

Additionally, the "*Major area, region, country, or area of destination*" column, "*North America*" is considered a major area, and it does not have a region. For consistency purposes, I added North America as both the major area as well as region for the countries and areas in this category. Moreover, I dropped the "*Sub-Saharan Africa*" region. This is an aggregation that can be easily recreated with code if such level of granularity is needed. "*World*" row was dropped with the same logic.

For similar reasons, the "*Developed*" and "*Developing*" rows were dropped too. For "*Least Developed*" I decided not to keep it out of choice. A categorization of "*Least Developed*" seems too harsh for a country to be in – this segmentation of countries is controversial as well. And, such a level of granularity is unnecessary. The least developed countries were categorized as developing.

Work Cited

Wickham, Hadley. *Tidy Data* | *Journal of Statistical Software*. 12 Sept. 2014, www.jstatsoft.org/article/view/v059i10.