# STAT570_Final_Project

Zehra Cebeci, Ömer Faruk Yahşi

## Data Cleaning and Tidying

In order to start coding for our project we need to load the necessary libraries for the project, including data manipulation (**`tidyverse`**, **`dplyr`**, **`purrr`**), data visualization (**`ggplot2`**), and data cleaning (**`janitor`**).

```r
library(tidyverse) # Comprehensive data manipulation and visualization tools
```

```
-- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
v dplyr     1.1.3     v readr     2.1.4
v forcats   1.0.0     v stringr   1.5.0
v ggplot2   3.4.3     v tibble    3.2.1
v lubridate 1.9.3     v tidyr     1.3.0
v purrr     1.0.2
-- Conflicts ------------------------------------------ tidyverse_conflicts() --
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to becon
```

```r
library(readxl)    # Reading Excel files into R
library(curl)      # Downloading files from the internet
```

```
Warning: package 'curl' was built under R version 4.3.2

Using libcurl 8.3.0 with Schannel

Attaching package: 'curl'

The following object is masked from 'package:readr':

    parse_date
```

```r
library(dplyr)     # Data manipulation toolkit
library(purrr)     # Functional programming tools
library(stringr)   # String manipulation functions
library(janitor)   # Tools for data cleaning
```

```

Attaching package: 'janitor'

The following objects are masked from 'package:stats':

    chisq.test, fisher.test
```

```r
library(ggplot2)   # Data visualization using Grammar of Graphics
```

The demographic data is obtained from an Excel file accessible via a specified URL. The file is downloaded utilizing the **curl_download** function, and the sheets of interest are identified.

```r
# Specify the web address of the Excel file
url <- "https://population.un.org/wpp/Download/Files/1_Indicators%20(Standard)/EXCEL_FILES

data <- "WPP2022_GEN_F01_DEMOGRAPHIC_INDICATORS_COMPACT_REV1.xlsx"

curl_download(url, destfile = data)
```

**Read_clean**, a custom function, is utilized to extract and clear data from individual sheets. Using **clean_names** and **map_df**, the final data is combined into a single data frame and standardized column names are established.

```r
# Downloaded sheets
sheets <- excel_sheets(data)

# Select the first two sheets
selected_sheets <- sheets[1:2]

# Function to read and clean data from a sheet
read_clean <- function(file_path, sheet) {
  read_excel(file_path, sheet = sheet, skip = 16, col_types = "text")
}

# Read and clean selected sheets, then bind them into a single data frame
raw_data <- map_df(selected_sheets, ~read_clean(data, .)) |>
```

```
    clean_names()
```

In this stage, we filter out rows with a specific type ('Label/Separator').

```
# Creating a new DataFrame without rows where 'type' is 'Label/Separator'
raw_data_wo_ls <- subset(raw_data, type != "Label/Separator")
```

We also convert population columns to numeric format and calculate the rate of change between January 1 and July 1 populations. The resulting data frame, **`sorted_data`**, is organized by year for further analysis.

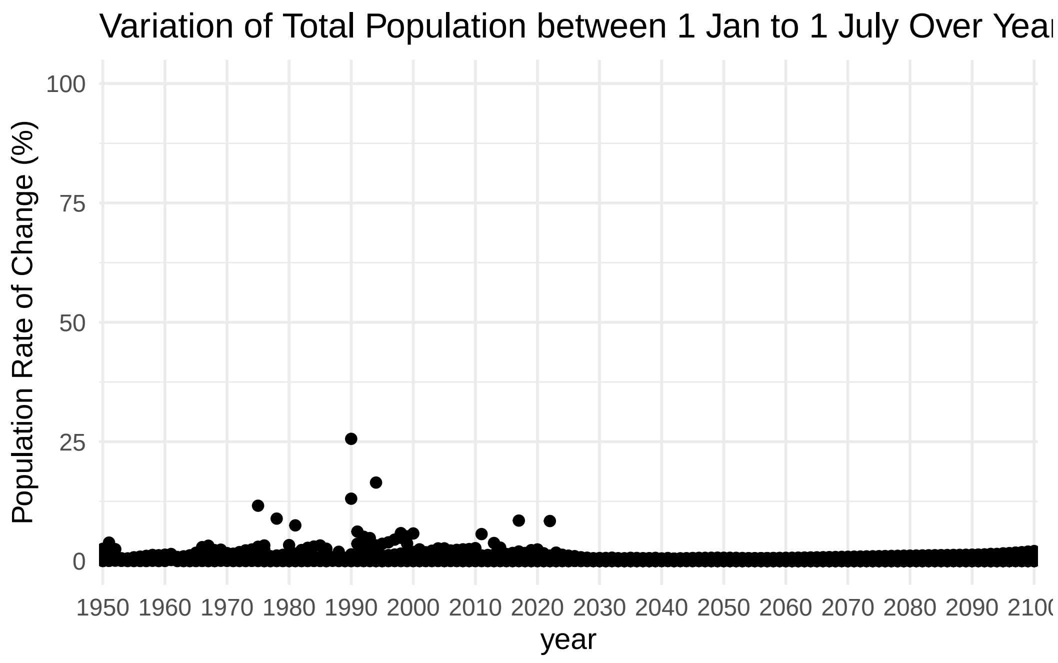
```
raw_data_wo_ls$total_population_as_of_1_january_thousands <- as.numeric(raw_data_wo_ls$tot
raw_data_wo_ls$total_population_as_of_1_july_thousands <- as.numeric(raw_data_wo_ls$total_

# Group by year and arrange by year
sorted_data <- raw_data_wo_ls |>
  mutate(rate_of_change = ((total_population_as_of_1_january_thousands - total_population_
  select(rate_of_change, everything()) |>
  group_by(year) |>
  arrange(year) |>
  ungroup()
```

To visually explore the variation in the total population rate of change over the years, we use the **`ggplot2`** package. A scatter plot is created according to years.

```
# Create a ggplot with the specified columns
ggplot(sorted_data, aes(x = year, y = rate_of_change)) +
  geom_point() +
  scale_y_continuous(name = "Population Rate of Change (%)",
                     limits = c(0, 100))+
  labs(title = "Variation of Total Population between 1 Jan to 1 July Over Years") +
  theme_minimal()+
  scale_x_discrete(breaks = seq(1950, 2100, by = 10))
```

```
Warning: Removed 31885 rows containing missing values (`geom_point()`).
```



After an in-depth examination of the percentage variation in populations between January 1 and July 1, it became apparent that, aside from a few outliers, the rate of change across the six time periods is not significantly pronounced. In light of this observation, we proceeded with a data reduction process to streamline our dataset for a more focused analysis. This involved excluding specific columns that were deemed unnecessary for our current analytical objectives.

This reduction aimed to simplify the dataset, making it more manageable for subsequent analyses while retaining the essential information for our research

```r
# Creating a new DataFrame without specific columns
raw_data_reduced <- raw_data_wo_ls |>
  select(-contains("_as_of_1_january"),
         -index,
         -iso3_alpha_code,
         -iso2_alpha_code,
         -sdmx_code,
         -parent_code,
         -notes,
         -location_code
         )
```

```r
total_na_count <- sum(is.na(raw_data_reduced))

print(total_na_count)
```

```
[1] 0
```

Upon initial analysis, the results of the **`total_na_count`** indicated the absence of any missing values. However, during the exploration of the scatter plot, a warning surfaced, indicating the presence of rows with missing values. This discrepancy prompted a closer examination of the data quality.

Further investigation revealed that the missing values were not explicitly defined as 'NA.' Instead, an unconventional placeholder, represented by three dots ("…"), was used to indicate missing data in the initial untidy dataset.

To rectify this issue, we took a two-step approach:

1. **Replacing "…" with NA:** All occurrences of the three dots ("…") in the dataset were replaced with the standard 'NA' representation. This step was crucial for ensuring uniformity and accuracy in identifying missing values across the dataset.
2. **Conversion to Numeric:** Subsequently, we converted all related columns with numerical values from character (string) to numeric format. This conversion ensured that the data, now with consistent missing value representation, could be accurately processed and analyzed.

```r
raw_data_rounded <- raw_data_reduced |>
  mutate(across(everything(), as.character)) |>
  mutate_all(~ifelse(. == "...", NA, .)) |>
  mutate_at(vars(names(raw_data_reduced)[4:ncol(raw_data_reduced)]), as.numeric, ~round(.,


total_na_count_after <- sum(is.na(raw_data_rounded))

print(total_na_count_after)
```

```
[1] 26128
```

We filter the data by level, including Country/Area, Sub-Region, Region, Development Group, and Income Group, in order to conduct a more comprehensive analysis. A distinct data frame is utilized to maintain each filtered dataset, which facilitates targeted analyses of particular demographic categories.

```r
# Filtering and Creating New Columns for Country/Area Level:
country_area_level_filtered <- raw_data_rounded |>
  filter(type == "Country/Area") |>
  mutate(country_and_area = region_subregion_country_or_area) |>
  select(-region_subregion_country_or_area) |>
  relocate("country_and_area", .after = "variant")

# The data is filtered to include only entries with the type "Country/Area."
# A new column 'country_and_area' is created, containing information from 'region_subregio
# Unnecessary columns are deselected, and 'country_and_area' is relocated after the 'varia

# Filtering and Creating New Columns for Sub-Region Level:
sub_region_level_filtered <- raw_data_rounded |>
  filter(str_detect(type, regex("region", ignore_case = FALSE))) |>
  mutate(sub_region = region_subregion_country_or_area) |>
  select(-region_subregion_country_or_area) |>
  relocate("sub_region", .after = "variant")

# Entries containing the term "region" (case-insensitive) are filtered, indicating the Sub
# A new column 'sub_region' is created, capturing information from 'region_subregion_count
# Unnecessary columns are deselected, and 'sub_region' is relocated after the 'variant' co

# Filtering and Creating New Columns for Region Level:
region_level_filtered <- raw_data_rounded |>
  filter(str_detect(type, regex("Region|World", ignore_case = FALSE))) |>
  mutate(region = region_subregion_country_or_area) |>
  select(-region_subregion_country_or_area) |>
  relocate("region", .after = "variant")

# Entries containing the terms "Region" or "World" (case-insensitive) are filtered, repres
# A new column 'region' is created, extracting information from 'region_subregion_country_
# Unnecessary columns are deselected, and 'region' is relocated after the 'variant' column

# Filtering and Creating New Columns for Development Group Level:
development_group_level_filtered <- raw_data_rounded |>
  filter(str_detect(type, regex("Development Group", ignore_case = FALSE))) |>
  mutate(development_group = region_subregion_country_or_area) |>
  select(-region_subregion_country_or_area) |>
  relocate("development_group", .after = "variant")

# Entries containing the term "Development Group" (case-insensitive) are filtered, indicat
# A new column 'development_group' is created, capturing information from 'region_subregio
# Unnecessary columns are deselected, and 'development_group' is relocated after the 'vari

# Filtering and Creating New Columns for Income Group Level:
income_group_level_filtered <- raw_data_rounded |>
  filter(str_detect(type, regex("Income Group", ignore_case = FALSE))) |>
  mutate(income_group = region_subregion_country_or_area) |>
  select(-region_subregion_country_or_area) |>
  relocate("income_group", .after = "variant")

# Entries containing the term "Income Group" (case-insensitive) are filtered, indicating t
# A new column 'income_group' is created, extracting information from 'region_subregion_co
# Unnecessary columns are deselected, and 'income_group' is relocated after the 'variant'
```