Data Wrangling (Data Preprocessing)

Data cleaning - Multiple datasets

Vasitha Tilakumara

Setup

```
# Load the necessary packages required to reproduce the report. For example:
library(kableExtra)
library(magrittr)

# Load the Library packages to run the below r codes
library(readxl)
library(dplyr)
library(tidyr)
library(Hmisc)
library(forecast)
library(editrules)
```

Executive Summary

- **Importing:** Imported data from Excel files into R data frames, “health_data” and “development_data,” utilizing the “read_xlsx” function.
- **Reshaping/ tidying the datasets:** Employed “pivot_longer” and “pivot_wider” to transform and tidy the datasets, for the purpose of joining two data sets and simplify the analysis.
- **Merging:** Applied a left_join operation to combine “health_data_tidy” and”development_data_tidy” data frames based on common columns - “Country Name,” “Country Code,” and “year,” ensuring the integration of all pertinent observations.
- **Data type conversions:** Segregated the merged data into three distinct data frames for controlled character-to-factor, character-to-numeric, and character-to-integer data type conversions.
- **Mutate new variable:** Introduced a new variable, “Population” by summing values from pre-existing variables, notably “Population.ages.0.14..total,” “Population.ages.15.64..total,” and “Population.ages.65.and.above..total.”
- **Scanning missing values:** Utilized the “apply” function in conjunction with “is.na()” to assess the number of missing values in each row of the “tidy_data” data frame.
- **Handling missing values:** “impute()” function from the “Hmisc” package is used to strategically impute missing values, adopting median imputation for “Birth_rate,” “CHE,” “Death_rate,” and “ANNI_PC,” and mean imputation for “Total_population.”
- **Scanning Special values:** scan the dataset for the presence of special values, particularly “Inf” and “NaN.”
- **Scanning obvious errors:** Obvious errors are scanned using user defined rules with the help of `editrules` package
- **Scanning Outliers:** scanned for outliers by plotting boxplot for numerical variables.
- **Handling Outliers:** Mitigated outliers through a capping method, replacing values below the 5th percentile with the 5th percentile value and values exceeding the 95th percentile with the 95th percentile value, thus enhancing data quality for analysis.
- **Transform:** BoxCox transformation used to obtain a normal distribution.

Data

The World Bank Group is a major sources of funding and knowledge for developing countries. They provide a data bank website which is hosted by world bank group. This data bank contain variety of information collected and stored for analysis purpose. Using **Health Nutrition and Population Statistics**[1] and **World Development Indicator**[2] databases from the data bank website we have obtain our two data sets **Dataset-1: health_data** and **Dataset-2: development_data** respectively.

Dataset-1: Health Nutrition and Population Statistics The dataset comprises a range of health indicators for various countries. These indicators offer valuable insights into different facets of public health, facilitating comprehensive analyses of health-related trends and outcomes in diverse nations. From this database we have obtain the database by selecting below explained Series names, countries and years.

URL: https://databank.worldbank.org/source/health-nutrition-and-population-statistics (https://databank.worldbank.org/source/health-nutrition-and-population-statistics)

The below is the description about the variable names of the healt data set and data types of the variables,

- `Country name :[character variable type]` The country variable consist of specific countries : Australia, India, China,Srilanka ,united kindom , United States, Russian Federation ,United Arab Emirates .
- `Country codes : [character variable type]` The county codes consists of country codes related to the countries specified in country name variable
- `Series Names :[character variable type]` series names collectively cover a wide spectrum of health-related aspects, making them valuable for comprehensive analyses of public health trends and outcomes in various countries. Following series names have to be selected from the **Health Nutrition and Population Statistics** database under series tab to obtain the data set. The series names consists of the following :
    - **Adolescent fertility rate (births per 1,000 women ages 15-19)**: The adolescent fertility rate represents the number of births per 1,000 women aged 15 to 19.
    - **Birth rate, crude (per 1,000 people)**: The crude birth rate signifies the number of live births in a given year per 1,000 people, based on midyear population estimates.

- **Cause of death, by communicable diseases and maternal, prenatal and nutrition conditions (% of total)**: This statistic indicates the percentage of all deaths attributed to communicable diseases, maternal and prenatal conditions, as well as nutrition-related issues. Communicable diseases encompass infectious and parasitic diseases, respiratory infections, and nutritional deficiencies like underweight and stunting.
- **Cause of death, by injury (% of total)**: This data reflects the percentage of all deaths caused by various injuries, including unintentional and intentional ones.
- **Cause of death, by non-communicable diseases (% of total)**: This statistic represents the percentage of all deaths attributed to non-communicable diseases, such as cancer, diabetes mellitus, cardiovascular diseases, digestive diseases, skin diseases, musculoskeletal diseases, and congenital anomalies.
- **Current health expenditure (% of GDP)**: Current health expenditure is the proportion of a country's GDP spent on healthcare goods and services in a given year. It does not include capital health expenditures, like investments in buildings, machinery, IT, or vaccine stocks for emergencies or outbreaks.
- **Death rate, crude (per 1,000 people)**: The crude death rate indicates the number of deaths in a population, per 1,000 people, without specifying the causes. This includes both unintentional and intentional injuries.
- **Current health expenditure per capita (current USS)**: This figure represents the amount spent on healthcare per person in a given year, measured in current US dollars. It includes the costs of healthcare goods and services consumed during the year.

- `Series Codes :[character variable type]` The "Series Code" in the dataset is a unique identifier associated with each "Series Name." These codes are used to distinguish and reference specific health indicators
- `2013[YR2013]: [character variable type]`, `2018[YR2018]: [character variable type]`, `2019[YR2019]:[numeric variable type]`, `2020[YR2020]:[character variable type]`, `2021[YR2021]: [character variable type]`, `2022[YR2022]:[character variable type]` These variables consist of values of "Adolescent fertility rate (births per 1,000 women ages 15-19) ,Birth rate, crude (per 1,000 people),Cause of death, by communicable diseases and maternal, prenatal and nutrition conditions (% of total),Cause of death, by injury (% of total),Cause of death, by non-communicable diseases (% of total), Current health expenditure (% of GDP), Death rate, crude (per 1,000 people), Current health expenditure per capita (current USS) ", in a country in respective variable name year.

Dataset-2: World Development Indicator The World Development Indicators (WDI) is the primary compilation of development metrics by the World Bank. These indicators are sourced from internationally recognized authorities, providing the most up-to-date and accurate global development data, including national, regional, and worldwide estimates. It's worth noting that while the WDI no longer includes the title 'Global Development Finance (GDF),' it still encompasses all external debt and financial flows data. The GDF publication has been renamed as 'International Debt Statistics (IDS)' and has a separate, dedicated database. From this database we have obtain the database by selecting below explained Series names, countries and years.

URL: https://databank.worldbank.org/source/world-development-indicators (https://databank.worldbank.org/source/world-development-indicators)

The below is the description about the variable names of the healt data set and data types of the variables,

- `Country name :[character variable type]`
- `Country codes : [character variable type]`
- `Series Codes :[character variable type]`
- `2013[YR2013]: [character variable type]`, `2018[YR2018]: [character variable type]`, `2019[YR2019]:[numeric variable type]`, `2020[YR2020]:[character variable type]`, `2021[YR2021]: [character variable type]`, `2022[YR2022]:[character variable type]`

The above mentioned variables are same as the health data set and only difference is the **Series Name** variable compared to health data set. Below explanation of the values and data type of the variable. Following series names have to be selected from the **World Development Indicator** database under series tab to obtain the data set.

- `Series Names :[character variable type]` series names collectively cover a wide spectrum of health-related aspects, making them valuable for comprehensive analyses of public health trends and outcomes in various countries. The series names consists of the following :
  - **Access to clean fuels and technologies for cooking, rural (% of rural population)**: This statistic represents the percentage of the rural population primarily using clean cooking fuels and technologies. Notably, kerosene is not considered a clean cooking fuel according to WHO guidelines.
  - **Access to clean fuels and technologies for cooking, urban (% of urban population)**: This indicator reflects the percentage of the urban population primarily using clean cooking fuels and technologies. It's important to note that, according to WHO guidelines, kerosene is not classified as a clean cooking fuel.
  - **Access to electricity, rural (% of rural population)**: This figure signifies the percentage of the rural population with access to electricity.
  - **Access to electricity, urban (% of urban population)**: This metric denotes the percentage of the urban population with access to electricity.
  - **Population ages 0-14, total**: This represents the total population within the age range of 0 to 14. The population count follows the de facto definition, encompassing all residents regardless of their legal status or citizenship.
  - **Population ages 15-64, total**: This indicates the total population within the age range of 15 to 64. The population count is based on the de facto definition, encompassing all residents regardless of their legal status or citizenship.
  - **Population ages 65 and above (% of total population)**: This statistic expresses the population aged 65 and above as a percentage of the total population. The population count follows the de facto definition, including all residents regardless of their legal status or citizenship.
  - **Adjusted net national income per capita (annual % growth)**: Adjusted net national income refers to Gross National Income (GNI) after subtracting consumption of fixed capital and accounting for natural resource depletion. This indicator measures the annual percentage growth of this adjusted net national income per person.

```
# Import the data, provide your R codes here.
health_data <- read_xlsx("Health_Nutrition_and_Population_Statistics.xlsx")
development_data <- read_xlsx("World_Development_Indicators.xlsx")
```

- Two data frames named **health_data** and **development_data** is created `read_xlsx` function.

```
# Number of 'NA' values in each row in the data frames
na_row_h<-apply(X = is.na(health_data), MARGIN = 1, FUN = sum)
na_row_d<-apply(X = is.na(development_data), MARGIN = 1, FUN = sum)

# get rid of the rows which contains all the variables 'NA'
health_data <- health_data[which(na_row_h < 10),]
development_data <- development_data[which(na_row_d < 9),]
```

This section get ride of observation having all the 'NA' in all the cells,

- `apply` function is used to apply `sum` function to rows (MARGIN = 1, apply `sum` function on all the rows) of the data frames to count the number of 'NA' values in each row.
- `is.na(health_data)` checks for 'NA' values in each cell of the **health_data** data frame, and similarly for the development_data data frame.
- `FUN = sum` applies the sum function to count the 'NA' values in each row. This results in two vectors, **na_row_h** and **na_row_d**, each containing the count of 'NA' values for the respective data frames.
- `Which()` : `which()` function filter the rows in `health_data` and `development_data` respectively, it filter out rows with NA count equal to total number of variables in each data sets (this remove all the rows having empty values in all the cells).

```
health_data$`2013 [YR2013]` <- as.numeric(health_data$`2013 [YR2013]`)
health_data$`2018 [YR2018]` <- as.numeric(health_data$`2018 [YR2018]`)
health_data$`2020 [YR2020]` <- as.numeric(health_data$`2020 [YR2020]`)
health_data$`2021 [YR2021]` <- as.numeric(health_data$`2021 [YR2021]`)
health_data$`2022 [YR2022]` <- as.numeric(health_data$`2022 [YR2022]`)

# Following data transformation obtain a partial tidy version of the data sets imported
health_data_tidy <-pivot_longer(health_data[,c(-4)],names_to = "year", values_to = "value", cols = 4:9)
health_data_tidy <- health_data_tidy %>% pivot_wider(names_from = "Series Name", values_from = "value")

# appply kable styling for head() function to display the tidied data frame
kable(head(health_data_tidy)[,1:4], caption = "health_data_tidy data frame columns[1:4]") %>%
  kable_styling(bootstrap_options = c("hover", "condensed"),full_width = TRUE)
```

health_data_tidy data frame columns[1:4]

Country Name	Country Code	year	Adolescent fertility rate (births per 1,000 women ages 15-19)
Australia	AUS	2013 [YR2013]	15.059
Australia	AUS	2018 [YR2018]	9.983
Australia	AUS	2019 [YR2019]	8.938
Australia	AUS	2020 [YR2020]	8.081
Australia	AUS	2021 [YR2021]	8.096
Australia	AUS	2022 [YR2022]	NA

```
kable(head(health_data_tidy)[,5:8], caption = "health_data_tidy data frame columns[5:8]") %>%
  kable_styling(bootstrap_options = c("hover", "condensed"),full_width = TRUE)
```

health_data_tidy data frame columns[5:8]

Birth rate, crude (per 1,000 people)	Cause of death, by communicable diseases and maternal, prenatal and nutrition conditions (% of total)	Cause of death, by injury (% of total)	Cause of death, by non-communicable diseases (% of total)
13.3	NA	NA	NA
12.6	NA	NA	NA
12.1	4.941054	5.944592	89.11435
11.5	NA	NA	NA
12.1	NA	NA	NA
NA	NA	NA	NA

```
kable(head(health_data_tidy)[,9:12], caption = "health_data_tidy data frame columns[9:12]") %>%
  kable_styling(bootstrap_options = c("hover", "condensed"),full_width = TRUE)
```

health_data_tidy data frame columns[9:12]

Current health expenditure (% of GDP)	Death rate, crude (per 1,000 people)	Current health expenditure per capita (current US$)	NA
8.750828	6.4	5843.174	NA

Current health expenditure (% of GDP)	Death rate, crude (per 1,000 people)	Current health expenditure per capita (current US$)	NA
10.073916	6.3	5864.395	NA
10.229891	6.7	5555.374	NA
10.648995	6.3	5901.106	NA
NA	6.7	NA	NA
NA	NA	NA	NA

This section the following **health_data** data frame will be transform in order to do the merging. To achieve this steps first all the variable data types that are used for transformation should be in the same data type. Since there is a numeric variable ("2019 [YR2019]") in the **health_data** data frame the rest of the character variables are converted to numeric variables. This conversion is required in order to do the data sets transformation in the below section using pivot_longer function.

All the character type variables are converted to numeric variables using `as.numeric()` function[3].

Due to the untidiness in the data sets it is required to transform the data sets in order to merge. To make it partially tidy for the purose of joining the two data sets data, they are reshape using `pivot_longer` and `pivot_wider` functions from the `tidyr` package[4]. Therefore, following steps are taken in each data sets,

- **Step 1: Using pivot_longer to Convert Data from Wide to Partially Tidy Format:**
  - `pivot_longer` is used to transform the health_data data frame from a wide format to a long format.
  - `health_data[, c(-4)]` selects all columns in the health_data data frame except the 4th column. This is done to exclude the "Series Code" column from the reshaping process since it is irrelevant for the analysis and it can create unexpected many-to-many relationship variables.
  - `names_to = "year"` specifies that the names of the columns being transformed (in this case, the years) will be stored in a new column named "year" in the partially tidy format.
  - `values_to = "value"` specifies that the values from the selected columns **cols = 4:9** (columns: "2013 [YR2013]","2018 [YR2018]","2019 [YR2019]","2020 [YR2020]","2021 [YR2021]","2022 [YR2022]") will be stored in a new column named "value".
- **Step 2: Using pivot_wider to Further Organize Data into a Wide Format:**
  - In this step, the partially tidy health_data_tidy data frame is further organized into a wide format.
  - `pivot_wider` is used to create separate columns for different values in the "Series Name" column.
  - `values_from = "value"` the values in the "value" column are used to populate the cells in the wide format.

As a result of these two steps, the data has been reshaped from the original wide format, where each year had its own column, to a partially tidy format, and then to a wide format where each "Series Name" becomes a separate column. This reshaping is done to create two data sets suitable for merging.

```
development_data_tidy <-pivot_longer(development_data[,c(-4)],names_to = "year", values_to = "value", cols = 4:8)
development_data_tidy <-development_data_tidy %>% pivot_wider(names_from = "Series Name", values_from = "value")

# appply kable styling for head() function to display the tidied data frame
kable(head(development_data_tidy)[,1:4], caption = "development_data_tidy data frame columns[1:4]") %>%
  kable_styling(bootstrap_options = c("hover", "condensed"),full_width = TRUE)
```

development_data_tidy data frame columns[1:4]

Country Name	Country Code	year	Access to clean fuels and technologies for cooking, rural (% of rural population)
Australia	AUS	2018 [YR2018]	100
Australia	AUS	2019 [YR2019]	100
Australia	AUS	2020 [YR2020]	100
Australia	AUS	2021 [YR2021]	100
Australia	AUS	2022 [YR2022]	..
India	IND	2018 [YR2018]	40.9

```
kable(head(development_data_tidy)[,5:8], caption = "development_data_tidy data frame columns[5:8]") %>%
  kable_styling(bootstrap_options = c("hover", "condensed"),full_width = TRUE)
```

development_data_tidy data frame columns[5:8]

Access to clean fuels and technologies for cooking, urban (% of urban population)	Access to electricity, rural (% of rural population)	Access to electricity, urban (% of urban population)	Adjusted net national income per capita (annual % growth)
100	100	100	..
100	100	100	..
100	100	100	..

Access to clean fuels and technologies for cooking, urban (% of urban population)	Access to electricity, rural (% of rural population)	Access to electricity, urban (% of urban population)	Adjusted net national income per capita (annual % growth)
100	100	100	..
..	..	..	..
89.1	93.9461421860235	99.1	4.6952974448871885

```
kable(head(development_data_tidy)[,9:12], caption = "development_data_tidy data frame columns[9:12]") %>%
  kable_styling(bootstrap_options = c("hover", "condensed"),full_width = TRUE)
```

development_data_tidy data frame columns[9:12]

Population ages 0-14, total	Population ages 15-64, total	Population ages 65 and above, total	NA
4688774	16365890	3911979	NA
4733160	16569868	4037189	NA
4756934	16733580	4164775	NA
4719755	16712026	4256298	NA
4722983	16864971	4390981	NA
370831170	912553718	85618417	NA

Same steps `pivot_longer()` & `Pivot_wider()` followed in the untidy **health_data** data frame is applied to untidy **development_data** data frame to prepare the data set for the merged.

- **Step 1: Reshaping the Data from Wide to Long Format Using pivot_longer:**
  - `pivot_longer` is used to transform the development_data data frame from a wide format to a long format.
  - `development_data[, c(-4)]` selects all columns in the development_data data frame except the 4th column. This is done to exclude the “Series Code” column from the reshaping process since it is irrelevant for the analysis and it can create unexpected many-to-many relationship variables.
  - `names_to = "year"` specifies that the column containing the variable names (e.g., “2018 [YR2018]”) in the wide format will be stored in a new column named “year” in the long format.
  - `values_to = "value"` specifies that the values of the variables will be stored in a new column named “value” in the long format.
  - `cols = 4:8` specifies the columns to be included in the reshaping process. In this case, it selects columns 4 through 8 (columns: “2018 [YR2018]”,“2019 [YR2019]”,“2020 [YR2020]”,“2021 [YR2021]”, “2022 [YR2022]”) in the wide format.

After this step, the **development_data_tidy** data frame will be in a long format, with a **“year”** column representing the variable names (e.g., “2018 [YR2018]”) and a **“value”** column representing the values of the variables.

- **Step 2: Reshaping the Data from Long to Wide Format Using pivot_wider:**
  - `pivot_wider` is used to transform the development_data_tidy data frame from a long format back to a wide format.
  - `names_from = "Series Name"` specifies that the values in the “Series Name” column in the long format will be used to generate new column names in the wide format.
  - `values_from = "value"` specifies that the values in the “value” column in the long format will populate the cells in the wide format.

In this step, the development_data_tidy data frame is reshaped from a long format, with “year” and “value” columns, back to a wide format where each unique value in the “Series Name” column becomes a separate column in the wide format.

```
# After partially tidying the data two data sets are joined based on the Country Name and the Country Code
merged_data <- health_data_tidy %>% left_join(development_data_tidy,by=c("Country Name","Country Code","year"))
```

The two untidy data frames **health_data_tidy** and **development_data_tidy** are merged using a `left_join` based on the columns **Country Name**, **Country Code** and **year**, which result in having all the observation in health_data data set plus all the matching observations from the development_data data set. The resulting merged data frame is stored in `merged_data`. In this scenario the left_join is used to preserve the values related to the **2013** in the **year** variable of the **health_data** data frame, since **2013** year value is not common in both the data sets **health_data** and **development_data**.

Understand

```
# data structure of the data frame
class(merged_data)
```

```
## [1] "tbl_df"     "tbl"        "data.frame"
```

```
# Dimensions of the datasets
dim(merged_data)
```

```
## [1] 96 21
```

```
# Display structure of the datasets
str(merged_data)
```

```
## tibble [96 × 21] (S3: tbl_df/tbl/data.frame)
##  $ Country Name                                                                                  : chr [1:96] "Aus
tralia" "Australia" "Australia" "Australia" ...
##  $ Country Code                                                                                  : chr [1:96] "AU
S" "AUS" "AUS" "AUS" ...
##  $ year                                                                                          : chr [1:96] "201
3 [YR2013]" "2018 [YR2018]" "2019 [YR2019]" "2020 [YR2020]" ...
##  $ Adolescent fertility rate (births per 1,000 women ages 15-19)                                 : num [1:96] 15.0
6 9.98 8.94 8.08 8.1 ...
##  $ Birth rate, crude (per 1,000 people)                                                          : num [1:96] 13.3
12.6 12.1 11.5 12.1 ...
##  $ Cause of death, by communicable diseases and maternal, prenatal and nutrition conditions (% of total): num [1:96] NA N
A 4.94 NA NA ...
##  $ Cause of death, by injury (% of total)                                                        : num [1:96] NA N
A 5.94 NA NA ...
##  $ Cause of death, by non-communicable diseases (% of total)                                     : num [1:96] NA N
A 89.1 NA NA ...
##  $ Current health expenditure (% of GDP)                                                         : num [1:96] 8.75
10.07 10.23 10.65 NA ...
##  $ Death rate, crude (per 1,000 people)                                                          : num [1:96] 6.4
6.3 6.7 6.3 6.7 ...
##  $ Current health expenditure per capita (current US$)                                           : num [1:96] 5843
5864 5555 5901 NA ...
##  $ NA.x                                                                                          : num [1:96] NA N
A NA NA NA NA NA NA NA NA ...
##  $ Access to clean fuels and technologies for cooking, rural (% of rural population)             : chr [1:96] NA
"100" "100" "100" ...
##  $ Access to clean fuels and technologies for cooking, urban (% of urban population)             : chr [1:96] NA
"100" "100" "100" ...
##  $ Access to electricity, rural (% of rural population)                                          : chr [1:96] NA
"100" "100" "100" ...
##  $ Access to electricity, urban (% of urban population)                                          : chr [1:96] NA
"100" "100" "100" ...
##  $ Adjusted net national income per capita (annual % growth)                                     : chr [1:96] NA
".." ".." ".." ...
##  $ Population ages 0-14, total                                                                   : chr [1:96] NA
"4688774" "4733160" "4756934" ...
##  $ Population ages 15-64, total                                                                  : chr [1:96] NA
"16365890" "16569868" "16733580" ...
##  $ Population ages 65 and above, total                                                           : chr [1:96] NA
"3911979" "4037189" "4164775" ...
##  $ NA.y                                                                                          : chr [1:96] NA N
A NA NA ...
```

```
# Print the summary table using kable and kable_styling
kable(summary(merged_data)[,1:12], caption = "Summary of the merged data frame PART I") %>%
  kable_styling(bootstrap_options = c("hover", "condensed"),full_width = TRUE)
```

Summary of the merged data frame PART I

Country Name	Country Code	year	Adolescent fertility rate (births per 1,000 women ages 15-19)	Birth rate, crude (per 1,000 people)	Cause of death, by communicable diseases and maternal, prenatal and nutrition conditions (% of total)	Cause of death, by injury (% of total)	Cause of death, by non-communicable diseases (% of total)	Current health expenditure (% of GDP)	Death rate, crude (per 1,000 people)	Current health expenditure per capita (current US$)	NA.x
Length:96	Length:96	Length:96	Min. : 2.348	Min. : 6.600	Min. : 3.550	Min. : 3.686	Min. :65.93	Min. : 2.858	Min. : 0.896	Min. : 55.67	Min. : NA
Class :character	Class :character	Class :character	1st Qu.: 9.282	1st Qu.: 9.762	1st Qu.: 5.031	1st Qu.: 5.969	1st Qu.:76.71	1st Qu.: 3.912	1st Qu.: 6.307	1st Qu.: 294.46	1st Qu.: NA

Country Name	Country Code	year	Adolescent fertility rate (births per 1,000 women ages 15-19)	Birth rate, crude (per 1,000 people)	Cause of death, by communicable diseases and maternal, prenatal and nutrition conditions (% of total)	Cause of death, by injury (% of total)	Cause of death, by non-communicable diseases (% of total)	Current health expenditure (% of GDP)	Death rate, crude (per 1,000 people)	Current health expenditure per capita (current US$)	NA.x
Mode :character	Mode :character	Mode :character	Median :13.720	Median :11.440	Median : 8.731	Median : 6.804	Median :83.65	Median : 5.168	Median : 7.046	Median : 1601.55	Mediai : NA
NA	NA	NA	Mean :15.556	Mean :11.988	Mean :10.235	Mean : 7.572	Mean :82.19	Mean : 7.038	Mean : 7.435	Mean : 2613.80	Mean :NaN
NA	NA	NA	3rd Qu.:17.595	3rd Qu.:14.459	3rd Qu.:13.002	3rd Qu.: 9.101	3rd Qu.:88.88	3rd Qu.: 9.875	3rd Qu.: 8.925	3rd Qu.: 4288.00	3rd Qu.: NA
NA	NA	NA	Max. :49.803	Max. :19.935	Max. :24.166	Max. :15.809	Max. :89.63	Max. :18.816	Max. :16.700	Max. :11702.41	Max. : NA
NA	NA	NA	NA's :26	NA's :26	NA's :82	NA's :82	NA's :82	NA's :39	NA's :26	NA's :39	NA's :96

```
kable(summary(merged_data)[,13:21], caption = "Summary of the merged data frame PART II") %>%
  kable_styling(bootstrap_options = c("hover", "condensed"),full_width = TRUE)
```

Summary of the merged data frame PART II

Access to clean fuels and technologies for cooking, rural (% of rural population)	Access to clean fuels and technologies for cooking, urban (% of urban population)	Access to electricity, rural (% of rural population)	Access to electricity, urban (% of urban population)	Adjusted net national income per capita (annual % growth)	Population ages 0-14, total	Population ages 15-64, total	Population ages 65 and above, total	NA.y
Length:96	Length:96	Length:96	Length:96	Length:96	Length:96	Length:96	Length:96	Length:96
Class :character	Class :character	Class :character	Class :character	Class :character	Class :character	Class :character	Class :character	Class :character
Mode :character	Mode :character	Mode :character	Mode :character	Mode :character	Mode :character	Mode :character	Mode :character	Mode :character
NA	NA	NA	NA	NA	NA	NA	NA	NA
NA	NA	NA	NA	NA	NA	NA	NA	NA
NA	NA	NA	NA	NA	NA	NA	NA	NA
NA	NA	NA	NA	NA	NA	NA	NA	NA

`class(merged_data)` is used to determine the class or data type of the object merged_data. The result of this code indicates that the object merged_data belongs to multiple classes: “tbl_df”, “tbl” and “data.frame”

Dimension of the merged data set is inspected using `dim()` function. The output of this function produced,

- No. of observations = 96
- No. of variables = 10

The merged data set is structured as a tibble. Summary of the types of variables and data structures in this data set:

- Character Variables (chr):
  - Country Name
  - Country Code
  - year
  - Adolescent fertility rate (births per 1,000 women ages 15-19)
  - Birth rate, crude (per 1,000 people)
  - Cause of death, by communicable diseases and maternal, prenatal and nutrition conditions (% of total)
  - Cause of death, by injury (% of total)
  - Cause of death, by non-communicable diseases (% of total)
  - Current health expenditure (% of GDP)
  - Death rate, crude (per 1,000 people)
  - Current health expenditure per capita (current US$) These columns contain character data, such as textual descriptions, numerical values represented as text, or missing data (denoted as “NA”).

- Numeric Variables:
    - Access to clean fuels and technologies for cooking, rural (% of rural population)
    - Access to clean fuels and technologies for cooking, urban (% of urban population)
    - Access to electricity, rural (% of rural population)
    - Access to electricity, urban (% of urban population)
    - Adjusted net national income per capita (annual % growth)
    - Population ages 0-14, total
    - Population ages 15-64, total
    - Population ages 65 and above, total
- Missing Data:
    - NA.x: The following variable contain missing or unspecified data, indicated by “NA.”
    - NA.y: Similar to “NA.x,” this variable contains missing or unspecified data, indicated by “NA.”

```
# Convert all the character type variables into factors
merged_data <- data.frame(lapply(merged_data[,names(merged_data)
                                        %in% c("Country Name","year")],
                                                  function(x) if (is.character(x)) {as.factor(x)} else {x}),
                          lapply(merged_data[,!names(merged_data)
                                          %in% c("Country Name","Country Code","year", "NA.x", "NA.y",
                                                  "Population ages 0-14, total",
                                                  "Population ages 15-64, total",
                                                  "Population ages 65 and above, total")],
                                                  function(x) if (is.character(x)) {as.numeric(x)} else {x}),
                         lapply(merged_data[,names(merged_data)
                                         %in% c("Population ages 0-14, total",
                                                 "Population ages 15-64, total",
                                                 "Population ages 65 and above, total")],
                                               function(x) if (is.character(x)) {as.integer(x)} else {x}))

# check the levels of the main factor variables used in the analysis
levels(merged_data$Country.Name)
```

```
##  [1] "Australia"
##  [2] "China"
##  [3] "Data from database: Health Nutrition and Population Statistics"
##  [4] "India"
##  [5] "Japan"
##  [6] "Last Updated: 09/21/2023"
##  [7] "Malaysia"
##  [8] "New Zealand"
##  [9] "Russian Federation"
## [10] "Singapore"
## [11] "Sri Lanka"
## [12] "Thailand"
## [13] "United Arab Emirates"
## [14] "United Kingdom"
## [15] "United States"
## [16] "Vietnam"
```

```
levels(merged_data$year)
```

```
## [1] "2013 [YR2013]" "2018 [YR2018]" "2019 [YR2019]" "2020 [YR2020]"
## [5] "2021 [YR2021]" "2022 [YR2022]"
```

- Data type conversions The merged data frame is subset into three data frames to apply character->factor, character->numeric, character->integer data type conversions using `as.factor` , `as.numeric` , `as.integer` functions[5] respectively. `lapply` function is used to apply the necessary user define function to obtain the data type conversions. It systematically applies user defined different data type conversion functions to specific subsets of merged data frame. In the end, all three subset data frame are combined into one data frame.

Converting Character Variables to Factors: The first subset data frame,

- Country Name
- year

`lapply` function is used in combination with a user define function to convert the selected categorical variables from character to factor

Converting Character Variables to Numeric: The second subset data frame,

- Access to clean fuels and technologies for cooking, rural (% of rural population)
- Access to clean fuels and technologies for cooking, urban (% of urban population)
- Access to electricity, rural (% of rural population)
- Access to electricity, urban (% of urban population)
- Adjusted net national income per capita (annual % growth)

`lapply` function is used in combination with a user define function to convert the selected character variables from character to numeric, which are suitable for continuous variables that can be subjected to mathematical calculations.

Character Variables Converted to Integer: The second subset data frame,

- Population ages 0-14, total
- Population ages 15-64, total
- Population ages 65 and above, total

`lapply` function is used in combination with a user define function to convert the selected character variables from character to numeric. Integers are ideal for count-like data, as population figures are whole numbers and typically do not have decimal values.

```
# Next, levels arranges and ordered
merged_data$Country.Name <- factor(merged_data$Country.Name,
                                   levels = c("Australia","China","India","Japan",
                                              "Malaysia","New Zealand","Russian Federation",
                                              "Singapore","Sri Lanka","Thailand",
                                              "United Arab Emirates","United Kingdom",
                                              "United States","Vietnam"))

merged_data$year <- factor(merged_data$year,
                           levels = c("2013 [YR2013]","2018 [YR2018]","2019 [YR2019]","2020 [YR2020]","2021 [YR2021]","2022
[YR2022]"),
                           labels = c("2013","2018","2019","2020","2021","2022"))

# check the levels of the main factor variables after arranging and labelling
levels(merged_data$Country.Name)
```

```
##  [1] "Australia"            "China"                "India"
##  [4] "Japan"                "Malaysia"             "New Zealand"
##  [7] "Russian Federation"   "Singapore"            "Sri Lanka"
## [10] "Thailand"             "United Arab Emirates" "United Kingdom"
## [13] "United States"        "Vietnam"
```

```
levels(merged_data$year)
```

```
## [1] "2013" "2018" "2019" "2020" "2021" "2022"
```

Next, factor variables are arranged and order. * `Country.Name` : Variable is arranged according to the alphabetical order * `year` : Variable is arranged from 2013~2022 and the level are label for clear visualization and understanding

Tidy & Manipulate Data I

```
# Subset the merged data set to get the variables used for the analysis
merged_data <- merged_data[,!names(merged_data) %in% c("Adolescent.fertility.rate..births.per.1.000.women.ages.15.19.",
                                                        "Current.health.expenditure.per.capita..current.US..",
                                                        "Cause.of.death..by.injury....of.total.", "Cause.of.death..by.commun
icable.diseases.and.maternal..prenatal.and.nutrition.conditions....of.total.",
                                                        "Cause.of.death..by.non.communicable.diseases....of.total.",
                                                        "Access.to.clean.fuels.and.technologies.for.cooking..rural....of.rur
al.population.",
                                                        "Access.to.clean.fuels.and.technologies.for.cooking..urban....of.urb
an.population." ,
                                                        "Access.to.electricity..rural....of.rural.population.",
                                                        "Access.to.electricity..urban....of.urban.population.")]
# Display merged_data data frame after subseting
kable(head(merged_data)[,1:5], caption = "merged_data data frame columns[1:5]") %>%
  kable_styling(bootstrap_options = c("hover", "condensed"),full_width = TRUE)
```

merged_data data frame columns[1:5]

Country.Name	year	Birth.rate..crude..per.1.000.people.	Current.health.expenditure....of.GDP.	Death.rate..crude..per.1.000.people.
Australia	2013	13.3	8.750828	6.4
Australia	2018	12.6	10.073916	6.3
Australia	2019	12.1	10.229891	6.7
Australia	2020	11.5	10.648995	6.3
Australia	2021	12.1	NA	6.7
Australia	2022	NA	NA	NA

```
kable(head(merged_data)[,6:9], caption = "merged_data data frame columns[5:10]") %>%
  kable_styling(bootstrap_options = c("hover", "condensed"),full_width = TRUE)
```

merged_data data frame columns[5:10]

Adjusted.net.national.income.per.capita..annual...growth.	Population.ages.0.14..total	Population.ages.15.64..total	Population.ages.65.and.above..to
NA	NA	NA	N
NA	4688774	16365890	39119
NA	4733160	16569868	40371
NA	4756934	16733580	41647
NA	4719755	16712026	42562
NA	4722983	16864971	43909

The two data sets selected were untidy at the beginning. To obtain the merged data frame it was required to transform the both the data sets using `pivot_longer` and `pivot_wider` functions from the `dplyr` package. This transformation is done at the **Data** section of the report and with a clear explanation on how the `pivot_longer` and `pivot_wider` functions are used. This steps facilitates the analysis of the data by confirming to tidy data principles, where each variable corresponds to a separate column, each observation is in a single row and each value in a single cell.

Subsetting: The following variables are removed since it will not be utilized in analysis.

- Subset of the merged data frame excluded some variables such as
- Adolescent.fertility.rate..births.per.1.000.women.ages.15.19."
- Current.health.expenditure.per.capita..current.US.."
- Cause.of.death..by.injury….of.total."
- Cause.of.death..by.communicable.diseases.and.maternal..prenatal.and.nutrition.conditions….of.total."
- Cause.of.death..by.non.communicable.diseases….of.total."
- Access.to.clean.fuels.and.technologies.for.cooking..rural….of.rural.population."
- Access.to.clean.fuels.and.technologies.for.cooking..urban….of.urban.population."
- Access.to.electricity..rural….of.rural.population."
- Access.to.electricity..urban….of.urban.population.

Tidy & Manipulate Data II

```
merged_data$Population <- merged_data$Population.ages.0.14..total +
                                      merged_data$Population.ages.15.64..total +
                                      merged_data$Population.ages.65.and.above..total


tidy_data <- data.frame(Country_name = merged_data$Country.Name,
                        Year = merged_data$year,
                        Birth_rate = merged_data$Birth.rate..crude..per.1.000.people.,
                        CHE = merged_data$Current.health.expenditure....of.GDP.,
                        Death_rate = merged_data$Death.rate..crude..per.1.000.people.,
                        ANNI_PC = merged_data$Adjusted.net.national.income.per.capita..annual...growth.,
                        Total_population = merged_data$Population)

head(tidy_data)
```

	Country_name <fct>	Year <fct>	Birth_rate <dbl>	CHE <dbl>	Death_rate <dbl>	ANNI_PC <dbl>	Total_population <int>
1	Australia	2013	13.3	8.750828	6.4	NA	NA
2	Australia	2018	12.6	10.073915	6.3	NA	24966643
3	Australia	2019	12.1	10.229891	6.7	NA	25340217
4	Australia	2020	11.5	10.648995	6.3	NA	25655289
5	Australia	2021	12.1	NA	6.7	NA	25688079
6	Australia	2022	NA	NA	NA	NA	25978935

6 rows

A new variable, "Population," is mutated in to the merged_data data frame. The variable is obtained by calculating the sum of following variables.

- `Population.ages.0.14..total`
- `Population.ages.15.64..total`
- `Population.ages.65.and.above..total`

This calculation aggregates the population data into a single variable.

Next the variables names are renamed for better understanding.

- Country_name = Country.Name
- Year = year
- Birth_rate = Birth.rate..crude..per.1.000.people.
- CHE = Current.health.expenditure….of.GDP.
- Death_rate = Death.rate..crude..per.1.000.people.
- ANNI_PC = Adjusted.net.national.income.per.capita..annual…growth.
- Total_population = Population

The primary objective of this tidy and manipulate is to create a more organized and structured data frame, tidy_data, by adding a calculated population variable and selecting specific columns of interest.

Scan I

```
# calculate the number of missing values in each row
num_missing_row<-apply(X = is.na(tidy_data), MARGIN = 1, FUN = sum)
```

In this section initially the number of missing values are obtain in each row and then number of missing values in each column is obtained. Below sections explain the step taken to check for missing values, then the result obtained from the calculation is used to get rid/ impute the rows and columns.

Calculating the Number of Missing Values per Row: `apply` function is employed in combination with `is.na()` to calculate the number of missing values in each row of the tidy_data data frame. Specifically, it applies the `sum` function across all the rows (MARGIN = 1, apply `sum` function on all the rows) to count the missing values in each row. This helps identify and filter out rows with a significant percentage of missing data, contributing to data quality improvement and facilitating further analysis.

```
# Dimensions of the data set
dimension<- dim(tidy_data)

# get rid of each row having 80% or more NA values
tidy_data <- tidy_data[which(num_missing_row < (dimension[2]*(80/100))),]
```

Filtering Rows with 80% or More Missing Values: Next, using the above calculated number of missing values in each row, then filtered the rows in tidy_data by retaining only those with less than 80% missing values. Rows with a higher percentage of missing values are removed. These rows with more 80% NA values in each is the result of the data transformation done at the beginning. It could have avoided by subsetting the data sets at the beginning but due to the assignment requirement it was left alone and data sets were partially tidied.

```
# check the number of missing value in each variable
tidy_data %>%  sapply(function(x) length(which(is.na(x))))
```

```
##     Country_name             Year       Birth_rate              CHE
##                0                0               14               27
##       Death_rate          ANNI_PC Total_population
##               14               53               14
```

Checking Missing Values in Each Column: sapply is employed in a professional and systematic manner to assess each column within the tidy_data data frame. The `sapply` function[6] iteratively processes each column denoted as x, and it applies the which(is.na(x)) function to identify the indices of missing values in each column. Consequently, the `length` function is employed to count the number of missing values for each variable providing a systematic way to detect and handle missing data.

```
# median imputation (for numerical variables)
tidy_data$Birth_rate <- impute(tidy_data$Birth_rate, fun = median)
tidy_data$CHE <- impute(tidy_data$CHE, fun = median)
tidy_data$Death_rate <- impute(tidy_data$Death_rate, fun = median)
tidy_data$ANNI_PC <- impute(tidy_data$ANNI_PC, fun = median)

# mean imputation (for numerical variables)
tidy_data$Total_population <- impute(tidy_data$Total_population, fun = mean)
```

Next, Explains how these missing values are handled,

- `impute()` function[7] used to perform imputation for missing values in numerical variables from an `Hmisc` package[8] in r..
- Median imputation (using average value of the all the values in the variable) is applied to **“Birth_rate”**, **“CHE”**, **“Death_rate”** and **“ANNI_PC”**
- Mean imputation (the value in the middle of a data set variable) is used for the **“Total_population”** variable. These strategies provide reasonable estimates for the missing data points.

```
# check the number of missing value in each variable
tidy_data %>%  sapply(function(x) length(which(is.na(x))))
```

```
##     Country_name             Year       Birth_rate              CHE
##                0                0                0                0
##       Death_rate          ANNI_PC Total_population
##                0                0                0
```

Now, It is evident that there are no missing values left in data frame with the help of the `sapply()` [9] in combination with `length()`, `which()`, `is.na()`.

```
# Checking for any infinite values in the data frame
sum(sapply(tidy_data,is.infinite))
```

```
## [1] 0
```

```
# checking for any Nan values (meaning "not a number")
sum(sapply(tidy_data,is.nan))
```

```
## [1] 0
```

- Check for number of `Inf` and `NaN` values in the merged data frame respectively. `is.infinite()` and `is.nan()` functions[10] in combination with `sum()` function are used to count the number of infinite and NaN (Not-a-Number) values in the tidy_data data frame. This helps identify special values that could affect analyses. In this scenario there are not infinte or NaN in th data frame.

```
# checking for obvious error based on the rules specfied
# defining all the rule for all the variables in the data set
Rules <- editset(c("Total_population>0",
                    "Birth_rate>0",
                    "CHE>0",
                    "Death_rate>0",
                    "Country_name %in% c(\"Australia\",\"China\",\"India\",\"Japan\",\"Malaysia\",
                                                                \"New Zealand\",\"Russian Federation\",\"Singapo
re\",
                                                                \"Sri Lanka\",\"Thailand\",\"United Arab Emirate
s\",
                                                                \"United Kingdom\",\"United States\",\"Vietnam
\")",
                   "Year %in% c(\"2013\",\"2018\",\"2019\",\"2020\",\"2021\",\"2022\")"))

# Set of all the rules defined
Rules
```

```
##
## Data model:
## dat1 : Country_name %in% c('Australia', 'China', 'India', 'Japan', 'Malaysia', 'New Zealand', 'Russian Federation', 'Sing
apore', 'Sri Lanka', 'Thailand', 'United Arab Emirates', 'United Kingdom', 'United States', 'Vietnam')
## dat2 : Year %in% c('2013', '2018', '2019', '2020', '2021', '2022')
##
## Edit set:
## num1 : 0 < Total_population
## num2 : 0 < Birth_rate
## num3 : 0 < CHE
## num4 : 0 < Death_rate
```

```
# Data set is checked against all the defined rules
Violated<-violatedEdits(Rules, tidy_data)

# sum of violation of all the rules defined above
sum(Violated)
```

```
## [1] 0
```

```
# summary of the all the rule violations
summary(Violated)
```

```
## No violations detected, 0 checks evaluated to NA
```

```
## NULL
```

This section will provide a detail description of handling obvious error using define rule with the use of `editset` function and `violatedEdits` function from `editrules` [11] package.

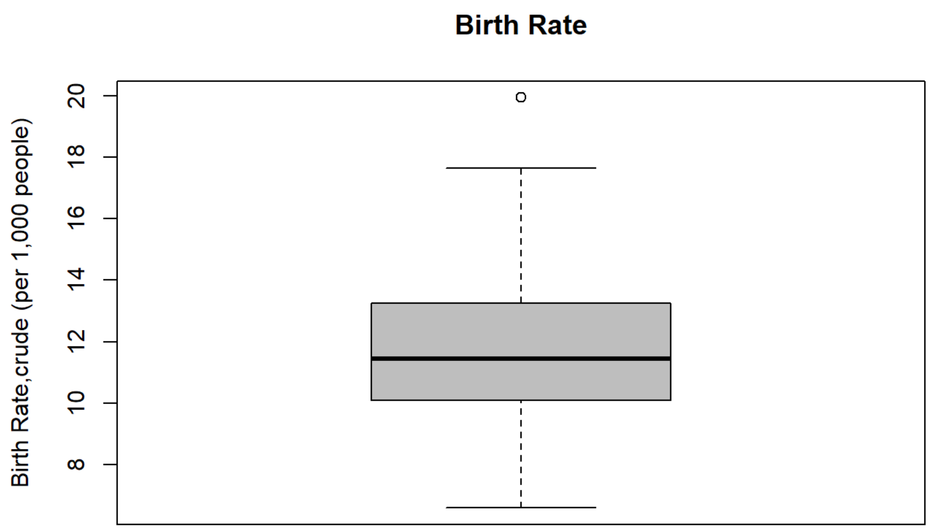
- Defining Rules: In this section, first, define the set of rules using the `editset` function from `editrules` package. These rules specify conditions for variables in the dataset. Here are the rules you defined:
- `Total_population > 0` : The total population should be greater than 0.
- `Birth_rate > 0` : The birth rate should be greater than 0.
- `CHE > 0` : A variable named CHE should be greater than 0.
- `Death_rate > 0` : The death rate should be greater than 0.
- `Country_name %in% c(...)` and `Year %in% c(...)` : These two rules specify that the Country_name should be one of the specified countries, and the Year should be one of the specified years.
- Applying Rules to Data: After defining these rules, they are applied to the data in each variables using the `violatedEdits` function. It checks each record in your dataset to see if it violates any of the rules. A TRUE value for a specific rule indicates a violation, while FALSE indicates no violation.

`summary()` function is used to check the summary of the data violations according to the define rules. In this scenario it is evident that there are no data violations

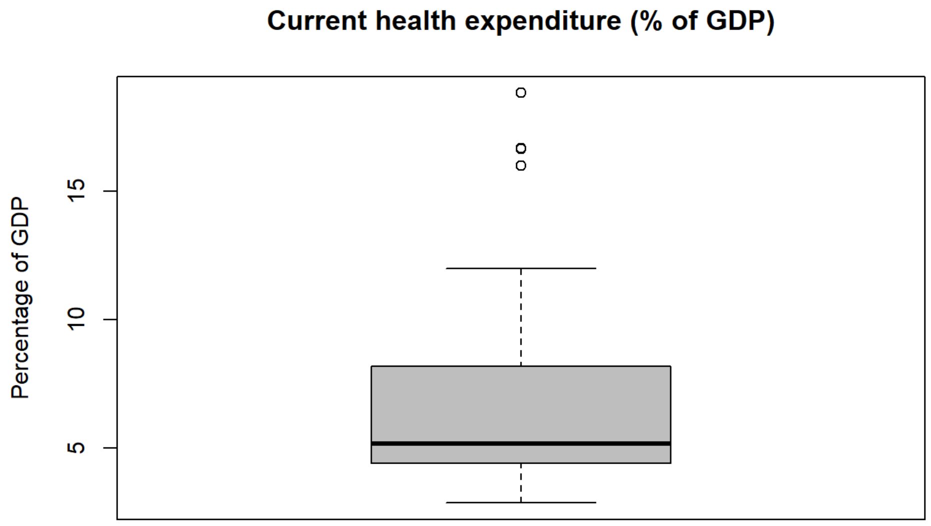
In summary, the **SCAN I** r code chunk were dedicated to data quality improvement and missing data handling. Rows with excessive missing values were removed, ensuring that the dataset retained high data integrity. Additionally, imputation techniques were applied to maintain the consistency of the dataset, enhancing its suitability for analysis. The checks for special values contributed to overall data quality assurance. Overall, these steps prepare the data for further analysis and interpretation, providing a more reliable foundation for deriving insights and making data-driven decisions.

Scan II

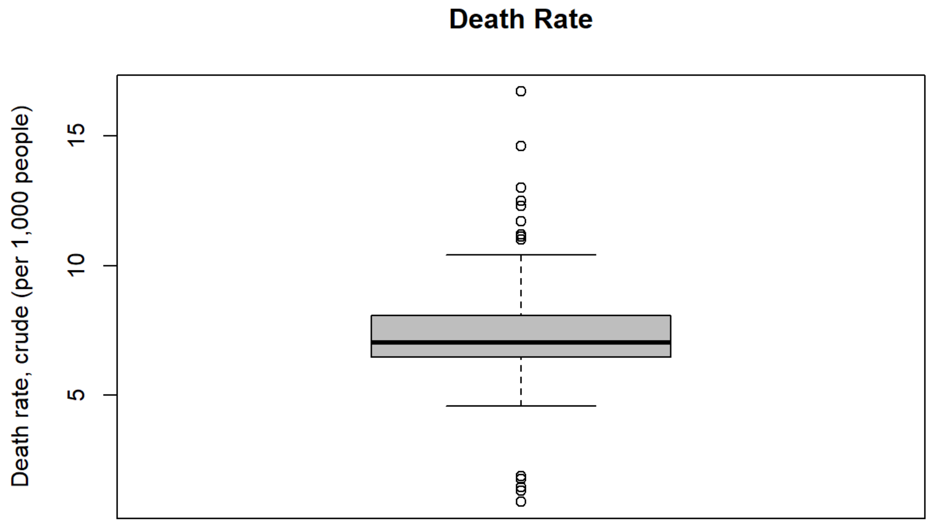
```
# checking the outliers in all the numericl variables using boxplot
tidy_data$Birth_rate %>%  boxplot(main="Birth Rate", ylab="Birth Rate,crude (per 1,000 people)", col = "grey")
```



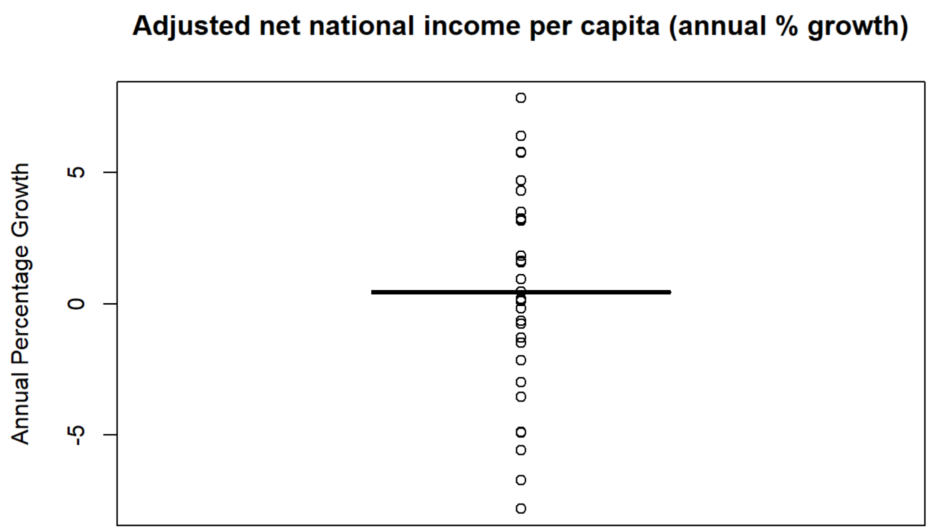

```
tidy_data$CHE %>%  boxplot(main="Current health expenditure (% of GDP)", ylab="Percentage of GDP", col = "grey")
```



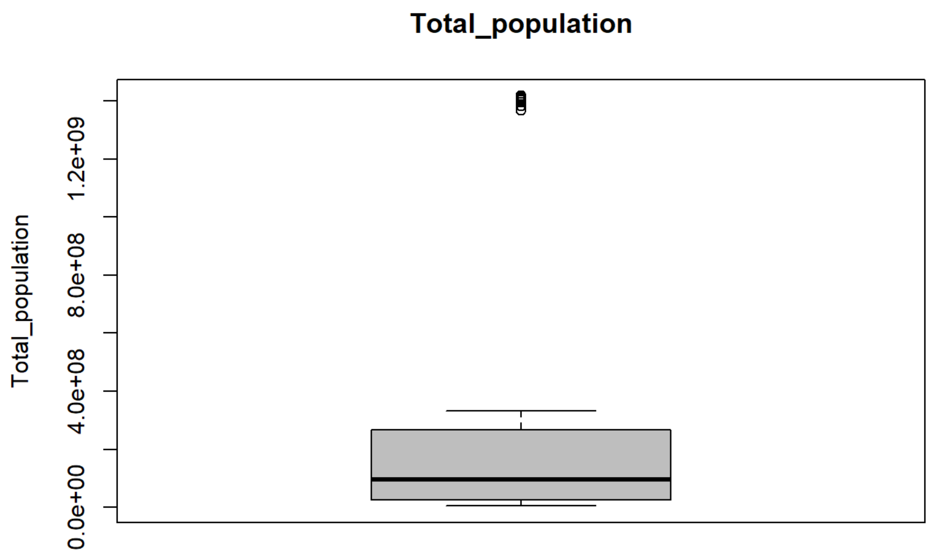

```
tidy_data$Death_rate %>%  boxplot(main="Death Rate", ylab="Death rate, crude (per 1,000 people)", col = "grey")
```




```
tidy_data$ANNI_PC %>%  boxplot(main="Adjusted net national income per capita (annual % growth)", ylab="Annual Percentage Gro
wth", col = "grey")
```




```
tidy_data$Total_population %>%  boxplot(main="Total_population", ylab="Total_population", col = "grey")
```



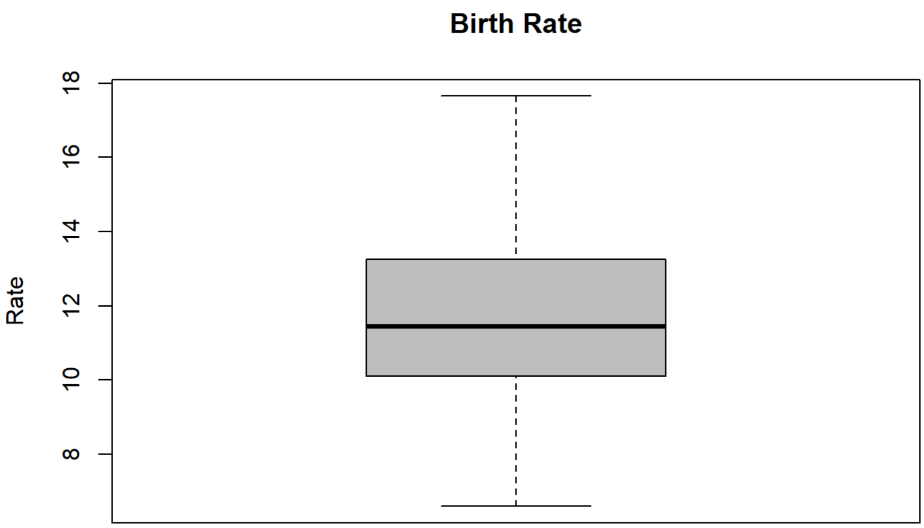
Following section detection of outliers and handling of outliers is achieved with the help of `boxplot()` and `Capping methods` [12]

- **Boxplot Visualization:** Created boxplots to identify the outliers in all the numeric variables in the merged dataset. Following are the numeric variables in the merged data frame,
  - Birth Rate
  - Current health expenditure (% of GDP)
  - Death Rate
  - Adjusted net national income per capita (annual % growth)
  - Population

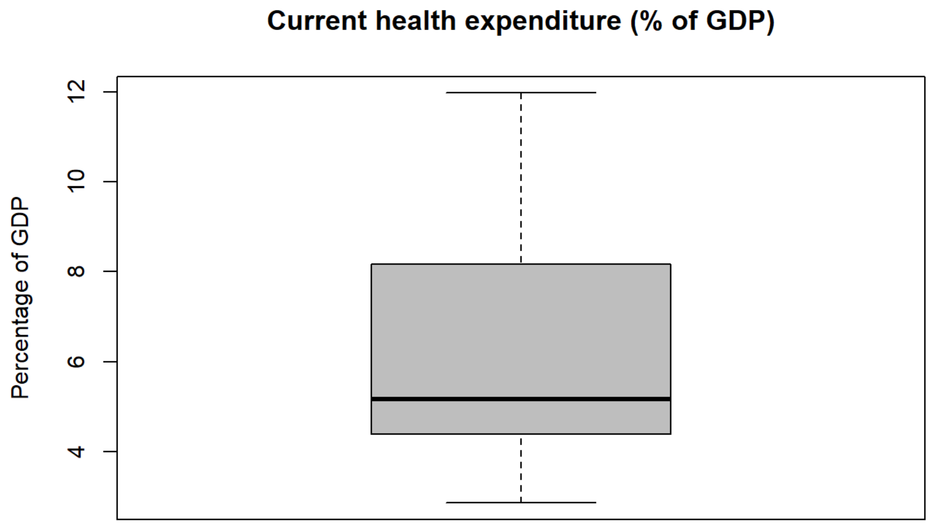
```
# function is define to handle outliers in the Salary variable
cap <- function(x){
    quantiles <- quantile( x, c(.05, 0.25, 0.75, .95 ))
    x[ x < quantiles[2] - 1.5*IQR(x) ] <- quantiles[1]
    x[ x > quantiles[3] + 1.5*IQR(x) ] <- quantiles[4]
    x}

# using the user define the cap function the capping method is used to handle the outliers
tidy_data$Birth_rate<-tidy_data$Birth_rate %>%  cap()
tidy_data$CHE<-tidy_data$CHE %>%  cap()
tidy_data$Death_rate<-tidy_data$Death_rate %>%  cap()
tidy_data$ANNI_PC<-tidy_data$ANNI_PC %>%  cap()
tidy_data$Total_population<-tidy_data$Total_population %>%  cap()
```

- **Handling Outliers**
  - Capping method is used to get rid of the outliers in the numeric variables.
  - User define function is created to handle outliers. It calculates the 5th, 25th, 75th and 95th percentiles of the input variable x using the quantile function. It replaces values in x that are less than the 5th percentile with the 5th percentile value and values greater than the 95th percentile with the 95th percentile value.The modified x is returned.This function essentially caps the extreme values at the 5th and 95th percentiles, which is a common way to handle outliers.

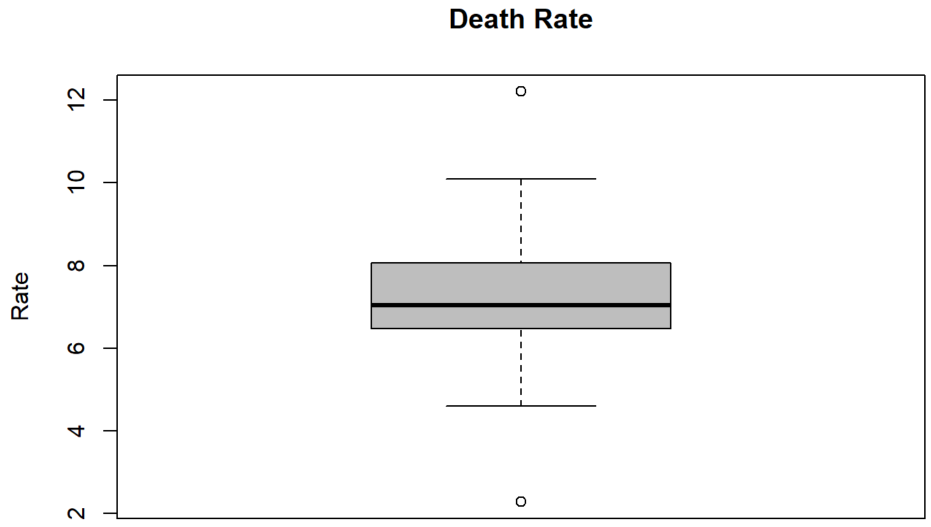
```
# Check for any more outliers after the capping method
tidy_data$Birth_rate %>%  boxplot(main="Birth Rate", ylab="Rate", col = "grey")
```



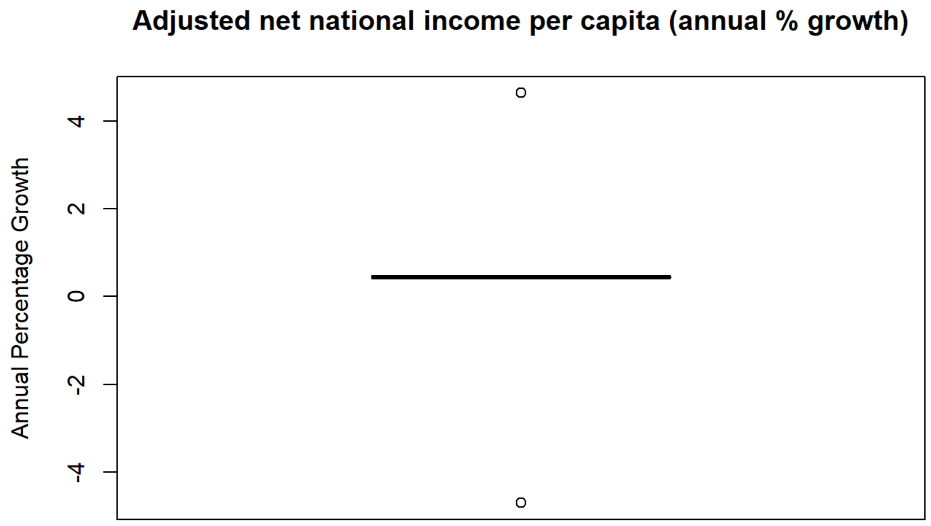

```
tidy_data$CHE %>%  boxplot(main="Current health expenditure (% of GDP)", ylab="Percentage of GDP", col = "grey")
```



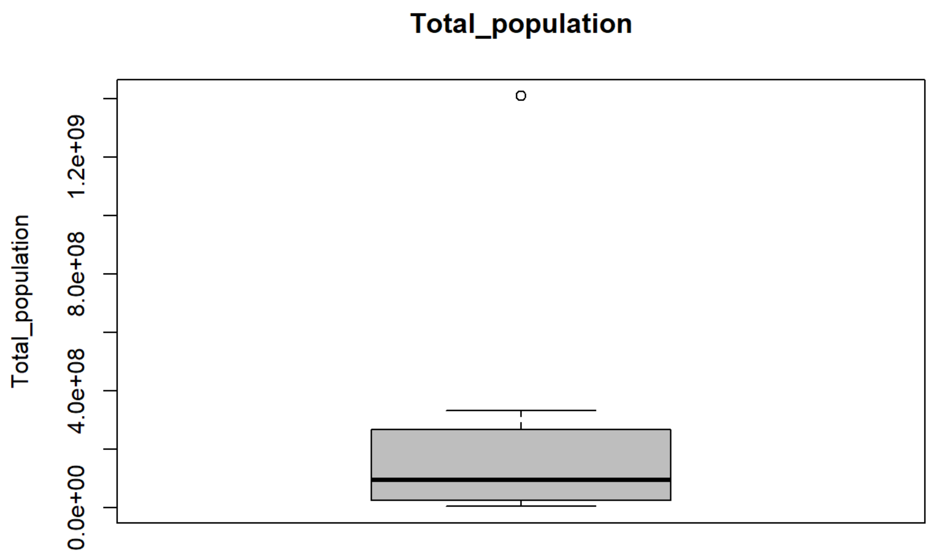

```
tidy_data$Death_rate %>%  boxplot(main="Death Rate", ylab="Rate", col = "grey")
```




```
tidy_data$ANNI_PC %>%  boxplot(main="Adjusted net national income per capita (annual % growth)", ylab="Annual Percentage Gro
wth", col = "grey")
```




```
tidy_data$Total_population %>%  boxplot(main="Total_population", ylab="Total_population", col = "grey")
```

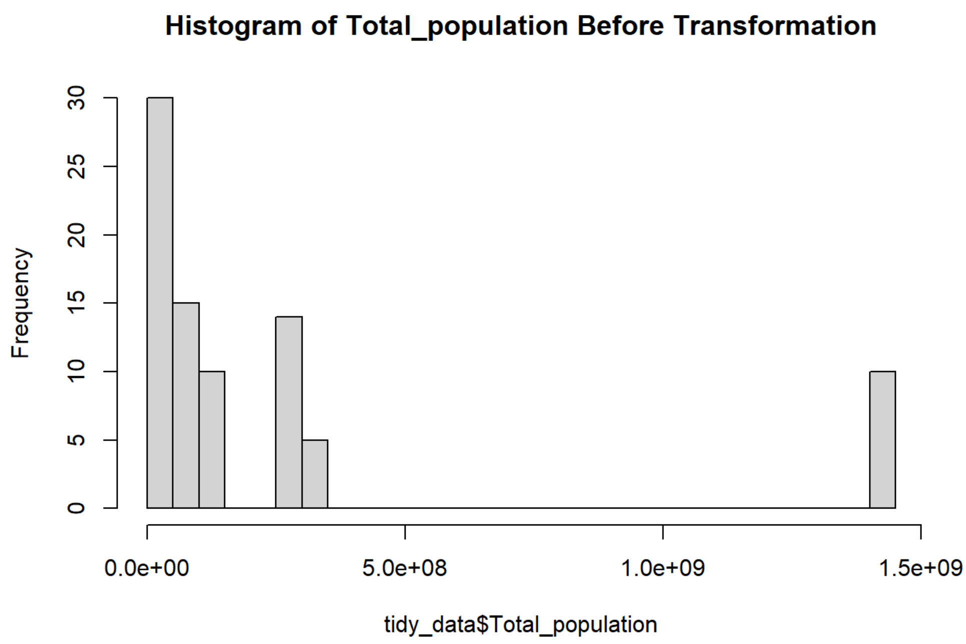


Finally, confirmed if the outliers are handled using boxplot for the same numerical variables after removing outliers using the cap function. The boxplots created in this step represent the modified data without outliers.

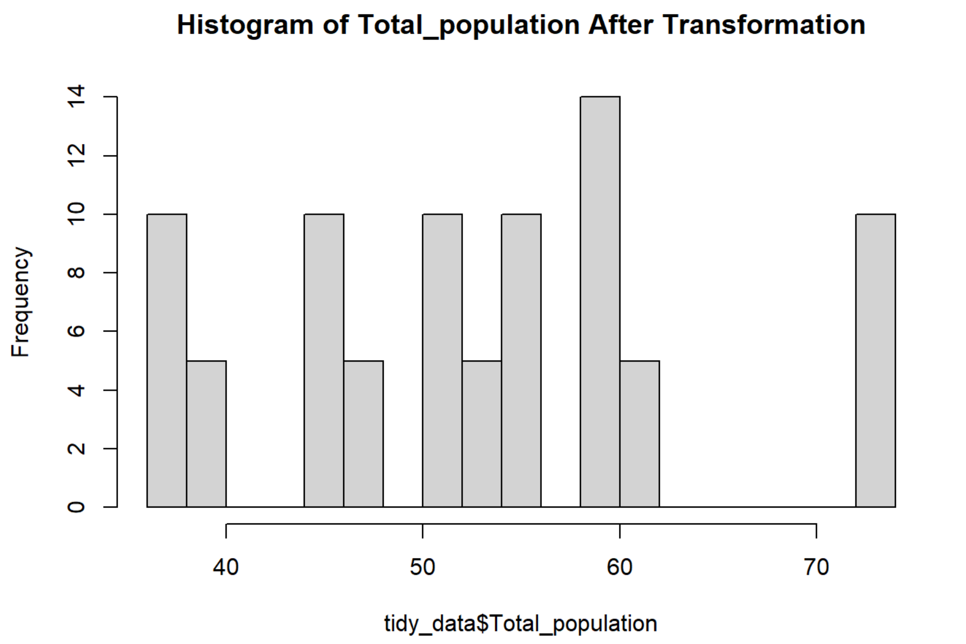
Overall, visually inspected the distribution and outliers of the numeric variables using boxplots, identified and handled outliers using the cap function, and then created new boxplots to visualize the cleaned data. This process is a common way to preprocess data and make it more suitable for further analysis.

Transform

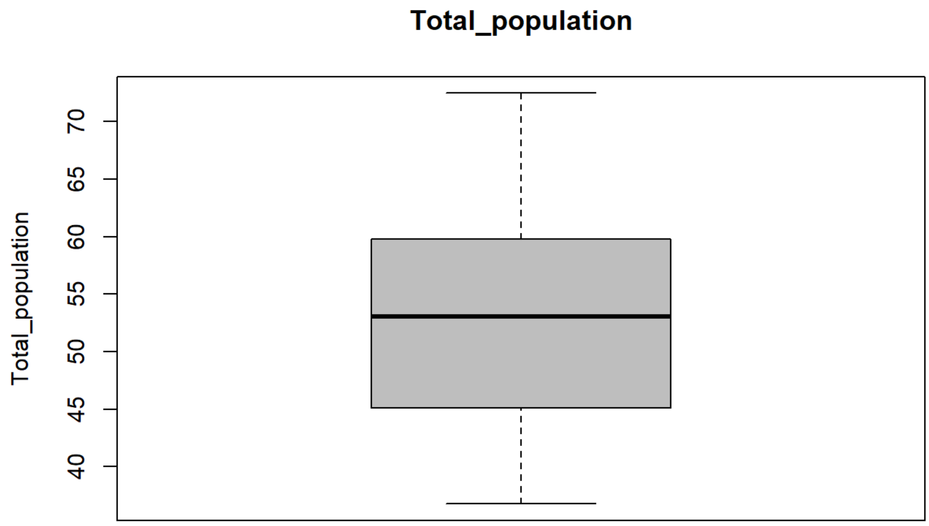
```
# distribution plot before transfromation
hist(tidy_data$Total_population,main="Histogram of Total_population Before Transformation", breaks = 25)
```




```
# applying BoxCox transformation to Total_population
tidy_data$Total_population<- BoxCox(tidy_data$Total_population,lambda = "auto")

# checking the normal distribution after data transformation
hist(tidy_data$Total_population,main="Histogram of Total_population After Transformation", breaks = 25)
```




```
# check for remaining outliers
tidy_data$Total_population %>%  boxplot(main="Total_population", ylab="Total_population", col = "grey")
```



- In this steps, data transformation technique known as **boxcox transformation** is applied to decrease the skewness and convert the distribution into a normal distribution. In order to achieve this function in the `forecast` [13] package is utilized.
- Check the distribution of the **Total_population** variable,
  - It is observe by plotting a histogram of the variable Total_population using the hist function the distribution is observed before the data transformation.
  - This code displays the distribution of the variable in a histogram with 25 bins.
- `Boxcox()` transformation:
  - Box-Cox transformation function from the package `forecast` is applied to the "Total_population" variable in a data frame named **tidy_data**.
  - `tidy_data$Total_population` : Selects the "Total_population" variable from the **tidy_data** data frame. This variable is transformed using the Box-Cox transformation.
  - `BoxCox(tidy_data$Total_population, lambda = "auto")` :
    - `Boxcox()` function is used to transform `tidy_data$Total_population` variable
    - `lambda = "auto"` : The lambda parameter determines the power to which the data is raised during the transformation. When lambda is set to "auto," the function will automatically estimate the optimal lambda value for the transformation.

In summary, this code takes the "Total_population" variable in the tidy_data data frame, applies the Box-Cox transformation to it, and automatically determines the best lambda value for the transformation. The Box-Cox transformation is often used to stabilize variance and make data more closely follow a normal distribution.

After the transformation from the boxplot it is evident the BoxCox transformation has handled the rest of the outliers in the **Total_population** variable.

Link to Presentation

https://rmit-arc.instructuremedia.com/embed/77c12da4-d0fb-4a05-9235-196660c32b20 (https://rmit-arc.instructuremedia.com/embed/77c12da4-d0fb-4a05-9235-196660c32b20)

Reference

1. Health Nutrition and Population Statistics (2023),DataBank website,Accessed 04 August 2023. https://databank.worldbank.org/source/health-nutrition-and-population-statistics (https://databank.worldbank.org/source/health-nutrition-and-population-statistics)↩

2. World Development Indicator (2023),DataBank website,Accessed 04 August 2023. https://databank.worldbank.org/source/world-development-indicators (https://databank.worldbank.org/source/world-development-indicators)↩

3. RMIT University (2023) Module 3 overview summary learning objectives to check attributes of R objects Learn how to convert between data types/structures, RMIT Canva website, Accessed 12 September 2023. https://rare-phoenix-161610.appspot.com/secured/Module_03.html#Summary (https://rare-phoenix-161610.appspot.com/secured/Module_03.html#Summary)↩

4. RMIT University (2023) Module 4 Overview Summary Learning Objectives Tidy Data Principles Common problems with messy data sets The tidyr, RMIT Canva website, Accessed 12 September 2023. http://rare-phoenix-161610.appspot.com/secured/Module_04.html (http://rare-phoenix-161610.appspot.com/secured/Module_04.html)↩

5. RMIT University (2023) Module 3 overview summary learning objectives to check attributes of R objects Learn how to convert between data types/structures, RMIT Canva website, Accessed 12 September 2023. https://rare-phoenix-161610.appspot.com/secured/Module_03.html#Summary (https://rare-phoenix-161610.appspot.com/secured/Module_03.html#Summary)↩

6. RMIT University (2023) Module 5 overview summary learning objectives missing data identifying missing data recode missing data excluding missing, RMIT Canva website, Accessed 12 September 2023. https://rare-phoenix-161610.appspot.com/secured/Module_05.html#Special_values (https://rare-phoenix-161610.appspot.com/secured/Module_05.html#Special_values)↩
7. RMIT University (2023) Module 5 overview summary learning objectives missing data identifying missing data recode missing data excluding missing, RMIT Canva website, Accessed 12 September 2023. https://rare-phoenix-161610.appspot.com/secured/Module_05.html#Special_values (https://rare-phoenix-161610.appspot.com/secured/Module_05.html#Special_values)↩
8. RMIT University (2023) Module 5 overview summary learning objectives missing data identifying missing data recode missing data excluding missing, RMIT Canva website, Accessed 12 September 2023. https://rare-phoenix-161610.appspot.com/secured/Module_05.html#Special_values (https://rare-phoenix-161610.appspot.com/secured/Module_05.html#Special_values)↩
9. RMIT University (2023) Module 5 overview summary learning objectives missing data identifying missing data recode missing data excluding missing, RMIT Canva website, Accessed 12 September 2023. https://rare-phoenix-161610.appspot.com/secured/Module_05.html#Special_values (https://rare-phoenix-161610.appspot.com/secured/Module_05.html#Special_values)↩
10. RMIT University (2023) Module 5 overview summary learning objectives missing data identifying missing data recode missing data excluding missing, RMIT Canva website, Accessed 12 September 2023. https://rare-phoenix-161610.appspot.com/secured/Module_05.html#Special_values (https://rare-phoenix-161610.appspot.com/secured/Module_05.html#Special_values)↩
11. RMIT University (2023) Module 5 overview summary learning objectives missing data identifying missing data recode missing data excluding missing, RMIT Canva website, Accessed 12 September 2023. https://rare-phoenix-161610.appspot.com/secured/Module_05.html#Special_values (https://rare-phoenix-161610.appspot.com/secured/Module_05.html#Special_values)↩
12. RMIT University (2023) Module 6 overview summary learning objectives outliers types of outliers most common causes of outliers detecting, RMIT Canva website, Accessed 12 September 2023. https://rare-phoenix-161610.appspot.com/secured/Module_06.html#Capping_(aka_Winsorising) (https://rare-phoenix-161610.appspot.com/secured/Module_06.html#Capping_(aka_Winsorising))↩
13. RMIT University (2023) Module 7 Transform: Data Transformation, Standardisation, and Reduction, RMIT Canva website, Accessed 12 September 2023. https://rare-phoenix-161610.appspot.com/secured/Module_06.html#Capping_(aka_Winsorising) (https://rare-phoenix-161610.appspot.com/secured/Module_06.html#Capping_(aka_Winsorising))↩