Data Wrangling (Data Preprocessing)

Practical assessment 2

Mohan Vaishnav Phani Sistla (s3853664)

Required Packages for Code Execution

```
# This is the R chunk for the required packages
library(readr)
```

```
## Warning: package 'readr' was built under R version 3.6.3
```

```
library(dplyr)
```

```
## Warning: package 'dplyr' was built under R version 3.6.3
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(tidyr)
```

```
## Warning: package 'tidyr' was built under R version 3.6.3
```

```
library(mice)
```

```
## Warning: package 'mice' was built under R version 3.6.3
```

```
##
## Attaching package: 'mice'
```

```
## The following object is masked from 'package:stats':
##
##     filter
```

```
## The following objects are masked from 'package:base':
##
##     cbind, rbind
```

```
library(editrules)
```

```
## Warning: package 'editrules' was built under R version 3.6.3
```

```
## Loading required package: igraph
```

```
## Warning: package 'igraph' was built under R version 3.6.3
```

```
##
## Attaching package: 'igraph'
```

```
## The following object is masked from 'package:tidyr':
##
##     crossing
```

```
## The following objects are masked from 'package:dplyr':
##
##     as_data_frame, groups, union
```

```
## The following objects are masked from 'package:stats':
##
##     decompose, spectrum
```

```
## The following object is masked from 'package:base':
##
##     union
```

```
##
## Attaching package: 'editrules'
```

```
## The following objects are masked from 'package:igraph':
##
##     blocks, normalize
```

```
## The following objects are masked from 'package:tidyr':
##
##     contains, separate
```

```
## The following object is masked from 'package:dplyr':
##
##     contains
```

```
library(forecast)
```

```
## Warning: package 'forecast' was built under R version 3.6.3
```

```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

Executive Summary of Data Preprocessing

The dataset taken to do the pre-processing task has been taken from Kaggle website which consists information regarding Rio de Janerio Olympics. The information related to the Olympics details has been split into 3 datasets namely Athletes, which contains basic athlete details, Country information of the associated athlete and Event, which consists information regarding each sport involved in the Olympics. The first step taken for data pre-processing is to download the dataset from the website and select the appropriate dataset to work with. Next step is to create a project for the task which involves storing a dataset in a folder followed by creating R Markdown for the report. Once this process is completed, the athletes and country datasets are loaded into R Markdown and combined using the common attribute of nationality. Once, datasets have been combined, the next step is to analyse each and every variable associated with it and making sure that every variable data type and their summary are in order. Necessary data type transformations have been taken place wherever there is any anomaly in the dataset variables. The dataset is next checked for tidiness. The untidy dataset has been tidied and a new column has also been added to the dataset through mutation process. Tidy dataset is now checked for any missing values, special values and data inconsistencies. Missing values has been updated with appropriate technique as per the variable type and conditions for checking the special values and data inconsistencies have been applied to verify the variable adherence to respective constraints. Imputation of the variables is followed by checking for the outliers in the numeric variables and their removal using relevant techniques. Lastly, numeric data transformation of variable has taken place to accommodate variable into normal distribution for better statistical analysis in future.

Data Vetting Process

The dataset used for the analysis is taken from the website https://www.kaggle.com/rio2016/olympic-games (https://www.kaggle.com/rio2016/olympic-games) which consists of information related to Rio De Janeiro Olympics athletes. This information is contained in 3 datasets namely athletes,countries and event. Athletes dataset contains information regarding personal information about the athletes. Countries data provides us with the names of countries along with their population and respective Per capita GDP. Events dataset provides all the essential information regarding the olympic events associated with each sport.

For the purpose of current data preprocessing, we shall consider only Athletes and Countries dataset.

The athletes dataset consists of personal information of athletes in the form of variables illustrated as below:

- ID : Id of the athlete as alloted by the olympics committee.
- Name : Name of the athlte.
- Nationality : Nationality code of the athlete.
- Sex : Gender of the athelete
- DOB : Date_of_birth of the athlete
- Height : Height of the athlete.

- Weight : Weight of the athlete
- Sport : Sport played by the athlete in the olympics.
- Gold : Number of Gold medals won by the athlete.
- Silver : Number of silver medals won by athlete.
- Bronze: Number of bronze medal won by athlete.

Countries dataset consists of below variables:

- Country: Name of the country.
- Code : Country Code.
- Population : Country Population
- GDP_per_Capita : Per capita GDP of a country

Prerequiste steps need to be done prior to loading of datasets and further data analysis:

1. Locating the URL of the mentioned datasets.
2. Downloading the datasets and storing them in a folder.
3. Creating an R project in the directory where the datasets have been stored.
4. Once the project directory has been established, we can go ahead with the loading of datasets and carry on with the analysis part.

Once the prerequisite steps has been completed, we can combine the athletes and countries dataset.In order to combine the athlete and country dataset, we can see that we have the nationality and Country code common in athlete and country dataset respectively. So,we can combine the two dataset based on this common criteria.

```
# R chunk to load and combine datasets

athletes <- read.csv("athletes.csv")

countries <- read.csv("countries.csv")

athletes_combined_info <- athletes %>% left_join(countries, c("nationality" = "code"))

# Splitting the head function in order to accomodate all the rows in the document

head(athletes_combined_info[1:6])
```

	id <int>	name <fct>	nationality <fct>	sex <fct>	dob <fct>	height <dbl>
1	736041664	A Jesus Garcia	ESP	male	10/17/69	1.72
2	532037425	A Lam Shin	KOR	female	9/23/86	1.68
3	435962603	Aaron Brown	CAN	male	5/27/92	1.98
4	521041435	Aaron Cook	MDA	male	1/2/91	1.83
5	33922579	Aaron Gate	NZL	male	11/26/90	1.81
6	173071782	Aaron Royle	AUS	male	1/26/90	1.80

6 rows

```
head(athletes_combined_info[7:14])
```

	weight <int>	sport <fct>	g… <int>	silver <int>	bronze <int>	country <fct>	population <int>	gdp_per_capita <dbl>
1	64	athletics	0	0	0	Spain	46418269	25831.582
2	56	fencing	0	0	0	Korea, South	50617045	27221.524
3	79	athletics	0	0	1	Canada	35851774	43248.530
4	80	taekwondo	0	0	0	Moldova	3554150	1848.062
5	71	cycling	0	0	0	New Zealand	4595700	37807.967
6	67	triathlon	0	0	0	Australia	23781169	56310.963

6 rows

As we can notice from the website and the loaded csv files, the number of observations amounts to 11538. For the sake of ease of data manipulation and preprocessing for the analysis, we shall consider only the first 100 observations of the combined dataset.Also, we shall discard the ID variable of the athlete as it plays no pivotal role in the current analysis.

```
# R chunk to consider only the first 100 observations of the data

athletes_info <- athletes_combined_info[1:100,-1]
dim(athletes_info)
```

```
## [1] 100  13
```

athletes_info dataframe now consists of 100 rows and 13 columns

So, the combined dataset of obtained provides details of all the athletes along with their country details as well. The variables associated with obtained combined dataset shall be useful to perform any statistical tests as well.

Understanding of The Dataset

```
# R chunk for data inspection of the data frame

str(athletes_info)
```

```
## 'data.frame':    100 obs. of  13 variables:
##  $ name          : Factor w/ 11517 levels "A Jesus Garcia",..: 1 2 3 4 5 6 7 8 9 10 ...
##  $ nationality   : Factor w/ 212 levels "AFG","ALB","ALG",..: 60 103 34 120 142 11 199 11
60 62 ...
##  $ sex           : Factor w/ 2 levels "female","male": 2 1 2 2 2 2 2 2 1 1 ...
##  $ dob           : Factor w/ 5596 levels "","1/1/69","1/1/73",..: 624 5380 3536 196 1231 3
14 4093 5415 1473 4412 ...
##  $ height        : num  1.72 1.68 1.98 1.83 1.81 1.8 2.05 1.93 1.8 1.65 ...
##  $ weight        : int  64 56 79 80 71 67 98 100 62 54 ...
##  $ sport         : Factor w/ 28 levels "aquatics","archery",..: 3 10 3 23 8 25 26 1 3 3
...
##  $ gold          : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ silver        : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ bronze        : int  0 0 1 0 0 0 1 0 0 0 ...
##  $ country       : Factor w/ 201 levels "Afghanistan",..: 170 99 36 122 132 11 192 11 170
65 ...
##  $ population    : int  46418269 50617045 35851774 3554150 4595700 23781169 321418820 2378
1169 46418269 99390750 ...
##  $ gdp_per_capita: num  25832 27222 43249 1848 37808 ...
```

```
summary(athletes_info)
```

```
##              name     nationality     sex           dob         height
##  A Jesus Garcia: 1   ALG    :10   female:11   1/1/97  : 2   Min.   :1.600
##  A Lam Shin    : 1   EGY    : 5   male  :89   10/11/92: 2   1st Qu.:1.728
##  Aaron Brown   : 1   GBR    : 5               5/27/92 : 2   Median :1.780
##  Aaron Cook    : 1   HUN    : 5               1/1/89  : 1   Mean   :1.796
##  Aaron Gate    : 1   MAR    : 5               1/1/94  : 1   3rd Qu.:1.853
##  Aaron Royle   : 1   BRN    : 4               1/10/97 : 1   Max.   :2.100
##  (Other)       :94   (Other):66               (Other) :91   NA's   :8
##      weight             sport         gold          silver         bronze
##  Min.   : 49.00   athletics:30   Min.   :0.00   Min.   :0.00   Min.   :0.00
##  1st Qu.: 64.00   aquatics :11   1st Qu.:0.00   1st Qu.:0.00   1st Qu.:0.00
##  Median : 71.00   football :10   Median :0.00   Median :0.00   Median :0.00
##  Mean   : 75.01   boxing   : 7   Mean   :0.03   Mean   :0.03   Mean   :0.04
##  3rd Qu.: 84.00   shooting : 7   3rd Qu.:0.00   3rd Qu.:0.00   3rd Qu.:0.00
##  Max.   :132.00   wrestling: 5   Max.   :1.00   Max.   :1.00   Max.   :1.00
##  NA's   :15       (Other)  :30
##            country     population        gdp_per_capita
##  Algeria       :10   Min.   :9.290e+04   Min.   :  277.1
##  Egypt         : 5   1st Qu.:1.020e+07   1st Qu.: 2878.2
##  Hungary       : 5   Median :3.585e+07   Median :12363.5
##  Morocco       : 5   Mean   :6.491e+07   Mean   :19536.0
##  United Kingdom: 5   3rd Qu.:5.788e+07   3rd Qu.:37808.0
##  (Other)       :66   Max.   :1.311e+09   Max.   :73653.4
##  NA's          : 4   NA's   :5           NA's   :7
```

As we can see from the above output, we have 13 variables in the dataset namely Name, Nationality, Sex, dob, height, weight, sport, gold, silver, bronze, country, population, gdp_per_capita with 100 observations.

Height and gdp_per_capita are of numeric data type and Weight,gold,silver,bronze and population belong to integer data type.

From the output, we can discern that name,nationality, sport and country have been represented as factor variable when in reality, they should have been characters. Also, the Date_of_birth of the athlete should be in date format and not in factor.

We will analyse each variable individually in order to rectify the above mentioned anamolies in data types and convert the variable wherever necessary.

- Name : Name of the athlete should be a character but not a factor.

```
# R chunk to convert name variable to character

athletes_info$name <- as.character(athletes_info$name)
class(athletes_info$name)
```

```
## [1] "character"
```

“Name” variable is changed to character

- Nationality : Just like variable “Name”, “Nationality” has been represented as factor.

```
# R chunk to convert nationality variable to character

athletes_info$nationality <- as.character(athletes_info$nationality)
class(athletes_info$name)
```

```
## [1] "character"
```

Nationality data type has been changed to character.

- Sex: Gender of the athletes has been aptly represented as factors.

```
# R chunk to view the levels of the variable Sex

athletes_info$sex %>% levels()
```

```
## [1] "female" "male"
```

Gender of the athletes participating in the olympics are male and female.

- dob : dob refers to the Date of birth of the athletes. Since, dob is the short hand representation of Date_of_birth, we need to rename the variable to its full form and change the data type of date from factor to date.

```
# R chunk to rename the dob variable

names(athletes_info)[names(athletes_info) == "dob"] <- "Date_of_birth"

# R chunk to convert date variable from factor to date format

athletes_info$`Date_of_birth` <- as.Date(athletes_info$`Date_of_birth`, format = "%m/%d/%Y")

class(athletes_info$`Date_of_birth`)
```

```
## [1] "Date"
```

The Date_of_birth variable has been renamed and converted to date date type.

- Country : Coutry variable needs to be converted to charater type instead of factor.

```
# R chunk to convert country variable to character

athletes_info$country <- as.character(athletes_info$country)
class(athletes_info$country)
```

```
## [1] "character"
```

Country variable has been converted to character.

- Sport : Similarly, sport variable needs to be converted to character data type;

```
# R chunk to convert sport variable to character

athletes_info$sport <- as.character(athletes_info$sport)
class(athletes_info$sport)
```

```
## [1] "character"
```

Sport variable has been converted to character.

```
# R chunk to view transformed data type conversions

str(athletes_info)
```

```
## 'data.frame':    100 obs. of  13 variables:
##  $ name          : chr  "A Jesus Garcia" "A Lam Shin" "Aaron Brown" "Aaron Cook" ...
##  $ nationality   : chr  "ESP" "KOR" "CAN" "MDA" ...
##  $ sex           : Factor w/ 2 levels "female","male": 2 1 2 2 2 2 2 2 1 1 ...
##  $ Date_of_birth : Date, format: "0069-10-17" "0086-09-23" ...
##  $ height        : num  1.72 1.68 1.98 1.83 1.81 1.8 2.05 1.93 1.8 1.65 ...
##  $ weight        : int  64 56 79 80 71 67 98 100 62 54 ...
##  $ sport         : chr  "athletics" "fencing" "athletics" "taekwondo" ...
##  $ gold          : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ silver        : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ bronze        : int  0 0 1 0 0 0 1 0 0 0 ...
##  $ country       : chr  "Spain" "Korea, South" "Canada" "Moldova" ...
##  $ population    : int  46418269 50617045 35851774 3554150 4595700 23781169 321418820 2378
1169 46418269 99390750 ...
##  $ gdp_per_capita: num  25832 27222 43249 1848 37808 ...
```

The above ouput reflects the correct data type of individual variable after respective conversion.

Now, as we can deduce from the above dataset, the above dataset is a bit untidy because the variables gold, silver and bronze are the medal types which should be included as observation for the Medal type variable but not as a variable name. Hence, after tidying the data, we can then order the medals as per the rank of the medal.

Tidy & Manipulate Data I (Untidy to Tidy Data Conversion)

As mentioned in the above step, the data is untidy because the Column names gold, silver and bronze should be included as observation values for the variable namely Medal type but should not be represented as Column header name by itself.

So, in order to tidy the data, we can create a separate column namely Medal_type which include gold, silver and bronze medal types and anothe column namely Medal_count which would give us the count of each medal.

```
# R chunk to create Medal_type and Medal_count columns

athlete_tidy <- athletes_info %>% pivot_longer(names_to = "Medal_type", values_to = "Medal_co
unt", cols = 8:10)
head(athlete_tidy[1:6])
```

name <chr>	nationality <chr>	sex <fct>	Date_of_birth <date>	height <dbl>	weight <int>
A Jesus Garcia	ESP	male	0069-10-17	1.72	64
A Jesus Garcia	ESP	male	0069-10-17	1.72	64
A Jesus Garcia	ESP	male	0069-10-17	1.72	64
A Lam Shin	KOR	female	0086-09-23	1.68	56
A Lam Shin	KOR	female	0086-09-23	1.68	56
A Lam Shin	KOR	female	0086-09-23	1.68	56

6 rows

```
head(athlete_tidy[7:12])
```

sport <chr>	country <chr>	population <int>	gdp_per_capita <dbl>	Medal_type <chr>	Medal_count <int>
athletics	Spain	46418269	25831.58	gold	0
athletics	Spain	46418269	25831.58	silver	0
athletics	Spain	46418269	25831.58	bronze	0
fencing	Korea, South	50617045	27221.52	gold	0
fencing	Korea, South	50617045	27221.52	silver	0
fencing	Korea, South	50617045	27221.52	bronze	0

6 rows

```
# R chunk to view data type of Medal_type

class(athlete_tidy$Medal_type)
```

```
## [1] "character"
```

As we can see, Medal_type variable is of character data type. In order to order the medal types according to their respective ranks, we need to convert the Medal_type to factor data type.

```
# R chunk to convert Medal_type to factor data type and Ordering the medals as per Olympics r
anking

athlete_tidy$Medal_type <- athlete_tidy$Medal_type %>% factor(levels = c("gold","silver","bro
nze"),
                                                              ordered = TRUE)

athlete_tidy$Medal_type %>% levels()
```

```
## [1] "gold"   "silver" "bronze"
```

From the above output, Medal_type has been converted to factor data type and gold, silver and bronze medals have been arranged as per the ranks obtained in the Olympics.

Tidy & Manipulate Data II (Data Mutation)

From the obtained athlete_tidy dataset, we can find the Body Mass Index (BMI) of the athlete participating in the Olympics using their repective body weights and heights.

Formula to Calculate BMI : BMI = $weight$ / $height^2$

```
# This is the R chunk to create a column BMI calculated from weight and height

athlete_tidy <- athlete_tidy %>% mutate(BMI = weight/(height^2))
head(athlete_tidy[1:6])
```

name <chr>	**nationality** <chr>	**sex** <fct>	**Date_of_birth** <date>	**height** <dbl>	**weight** <int>
A Jesus Garcia	ESP	male	0069-10-17	1.72	64
A Jesus Garcia	ESP	male	0069-10-17	1.72	64
A Jesus Garcia	ESP	male	0069-10-17	1.72	64
A Lam Shin	KOR	female	0086-09-23	1.68	56
A Lam Shin	KOR	female	0086-09-23	1.68	56
A Lam Shin	KOR	female	0086-09-23	1.68	56

6 rows

```
head(athlete_tidy[7:13])
```

sport <chr>	**country** <chr>	**population** <int>	**gdp_per_capita** <dbl>	**Medal_type** <ord>	**Medal_count** <int>	**BMI** <dbl>
athletics	Spain	46418269	25831.58	gold	0	21.63332
athletics	Spain	46418269	25831.58	silver	0	21.63332
athletics	Spain	46418269	25831.58	bronze	0	21.63332
fencing	Korea, South	50617045	27221.52	gold	0	19.84127

sport <chr>	country <chr>	population <int>	gdp_per_capita <dbl>	Medal_type <ord>	Medal_count <int>	BMI <dbl>
fencing	Korea, South	50617045	27221.52	silver	0	19.84127
fencing	Korea, South	50617045	27221.52	bronze	0	19.84127

6 rows

As we can find from above output, BMI of an athlete has been added to the dataset.

Scan I (Checking for Missing values, Special values and Data Inconsistencies)

Extending our dataset analysis, in this step, we shall explore the dataset for possible missing values, special values and obvious errors.

Checking Missing values:

```
# R code to check for NA values in each columns in dataset

colSums(is.na(athlete_tidy))
```

```
##           name    nationality            sex  Date_of_birth         height
##              0              0              0              0             24
##         weight          sport        country     population gdp_per_capita
##             45              0             12             15             21
##     Medal_type    Medal_count            BMI
##              0              0             45
```

As purported from above, height, weight, country, population, gdp_percapita and BMI are the variables that contain 24,45,12,15,21,45 missing values respectively.

Out of the mentioned variables, country is the only variable which is of character data type. Here, Country is the kind of variable which is unique to a particular athlete participating in the Olympics. So, in order to impute the country which is of character data type, we'll use the Mode() function which is, in this case best suitable to impute the missing country value of an athlete in the dataset. Mode() function will take the maximum number of appearing country in dataset and try to impute the missing value to it.

```
# R code to impute the missing value for country variable

athlete_tidy$country[which(is.na(athlete_tidy$country))] <- mode(athlete_tidy$country)
sum(is.na(athlete_tidy$country))
```

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

Country values have been imputed as the number of missing values for country is shown zero.

Now, if we check for other numeric and integer variable, height, weight and BMI of an athlete participating in Olympics has to be related to some extent because of the Category levels conditions set by Olympics committee. Because, height and weight of an athlete can determine his/her BMI, we shall use the technique of "Predictive Mean Matching" (pmm) to impute the missing values in height,weight and BMI variables. PMM method is a semi parametric imputation technique similar to that of regression.

```
# R code to impute Height,weight and BMI variables

athlete_measurment_impute <- athlete_tidy %>% select(height,weight,BMI)
head(athlete_measurment_impute)
```

height <dbl>	weight <int>	BMI <dbl>
1.72	64	21.63332
1.72	64	21.63332
1.72	64	21.63332
1.68	56	19.84127
1.68	56	19.84127
1.68	56	19.84127

6 rows

```
athlete_impute1 <- mice(athlete_measurment_impute,m=5,
                        maxit = 50,method = 'pmm',seed = 500)
# pmm method for data imputation
```

```
athlete_measurment_impute <- complete(athlete_impute1) # reverting back the imputed list into
data frame


colSums(is.na(athlete_measurment_impute))
```

```
## height weight    BMI
##      0      0      0
```

```
# Mapping back the imputed data frame to original tidy dataframe

athlete_tidy$height <- athlete_measurment_impute$height

athlete_tidy$weight <- athlete_measurment_impute$weight

athlete_tidy$BMI <- athlete_measurment_impute$BMI
```

From the above output, we can see that height, weight and BMI values of the athlete have been imputed.

As for the population and gdp_per_capita of a country is concerned, both of the variables has its own individually values and no specific relation as such. So we can impute the missing values with mean of the respective variable.

```
# R code to impute population and gdp_per_capita variable

athlete_tidy$population[is.na(athlete_tidy$population)] <- mean(athlete_tidy$population, na.r
m = TRUE)

athlete_tidy$gdp_per_capita[is.na(athlete_tidy$gdp_per_capita)] <- mean(athlete_tidy$gdp_per_
capita,
                                                                        na.rm = T
RUE)

# Verifying whether values has been imputed or not

sum(is.na(athlete_tidy$population))
```

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

```
sum(is.na(athlete_tidy$gdp_per_capita))
```

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

Missing values in population and gdp_per_capita has been imputed as NA values is shown zero in the above output.

```
# R code to verify that original tidy dataset variables have been imputed or not

colSums(is.na(athlete_tidy))
```

```
##           name    nationality            sex  Date_of_birth         height
##              0              0              0              0              0
##         weight          sport        country     population gdp_per_capita
##              0              0              0              0              0
##     Medal_type    Medal_count            BMI
##              0              0              0
```

As shown, all missing numeric and integer variables has been successfully imputed.

Checking for Special Values:

Now that we have imputed any missing values in the dataset, we shall now look into the dataset for amy special values and try to impute them as well. As for special values, we shall look for any infinite or NAN value in the data.

```
# R chunk to check for any special values in dataset

is.specialorNA <- function(x){
if (is.numeric(x)) (is.infinite(x) | is.nan(x))
}

sapply(athlete_tidy, function(x) which(is.specialorNA(x)==TRUE))
```

```
## $name
## integer(0)
##
## $nationality
## integer(0)
##
## $sex
## integer(0)
##
## $Date_of_birth
## integer(0)
##
## $height
## integer(0)
##
## $weight
## integer(0)
##
## $sport
## integer(0)
##
## $country
## integer(0)
##
## $population
## integer(0)
##
## $gdp_per_capita
## integer(0)
##
## $Medal_type
## integer(0)
##
## $Medal_count
## integer(0)
##
## $BMI
## integer(0)
```

In the above code snippet, we tried to find special values in dataset such as infinite and NAN and found that dataset does not contain any of the special values.

Checking for data inconsistencies in dataset :

In the datasets, we have certain variables such as height, weight where the values has to obey certain rules. For instances, height of a person cannot be 0 and BMI cannot be zero. In order to take care of such inconsistencies, we shall form the rules accordingly and check the dataset for any such inconsistencies.

```
# R chunk to validate certain rules in the dataset

(Rule1 <- editset(c("height >= 0", "height <= 3","weight >=0","BMI >=0")))  # Framing rules a
s per variable constraint
```

```
##
## Edit set:
## num1 : 0 <= height
## num2 : height <= 3
## num3 : 0 <= weight
## num4 : 0 <= BMI
```

```
Violated <- violatedEdits(Rule1, athlete_tidy)  # checking for any rule violation
summary(Violated)
```

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

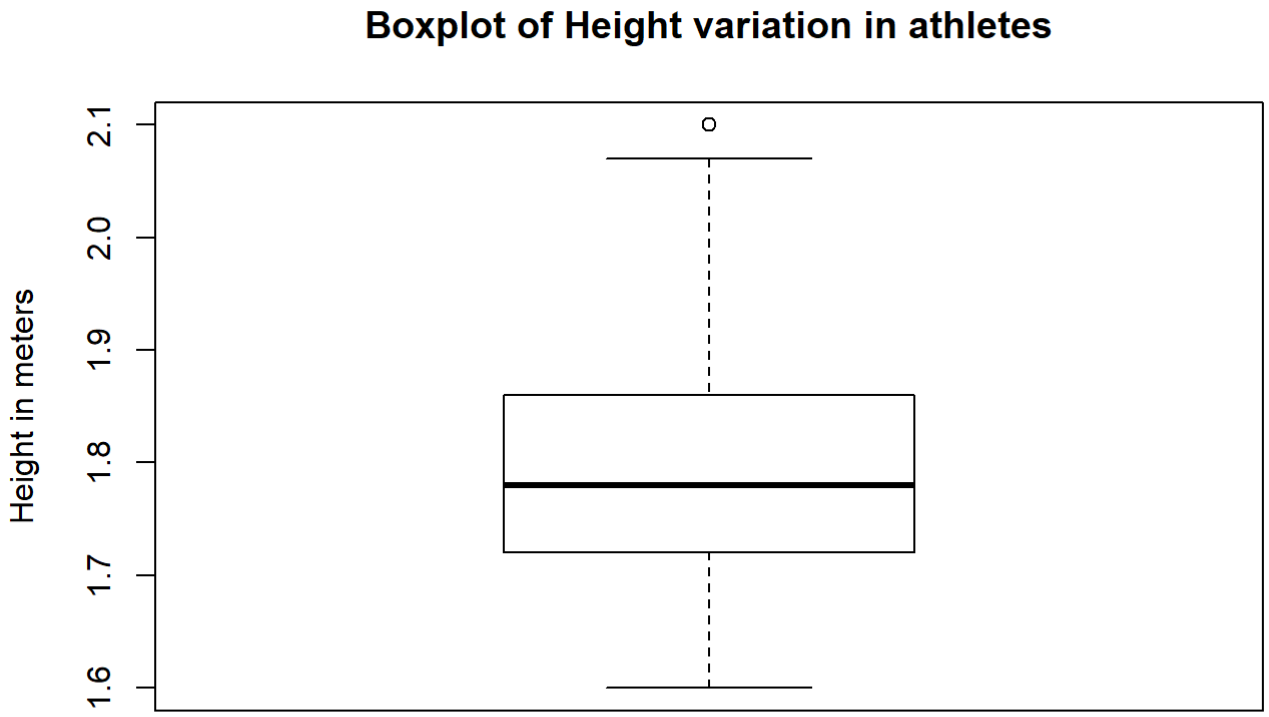
```
## NULL
```

So, from the above output, we can conclude that dataset dosent violate any of the mention rules and adheres to the restriction of the variables.

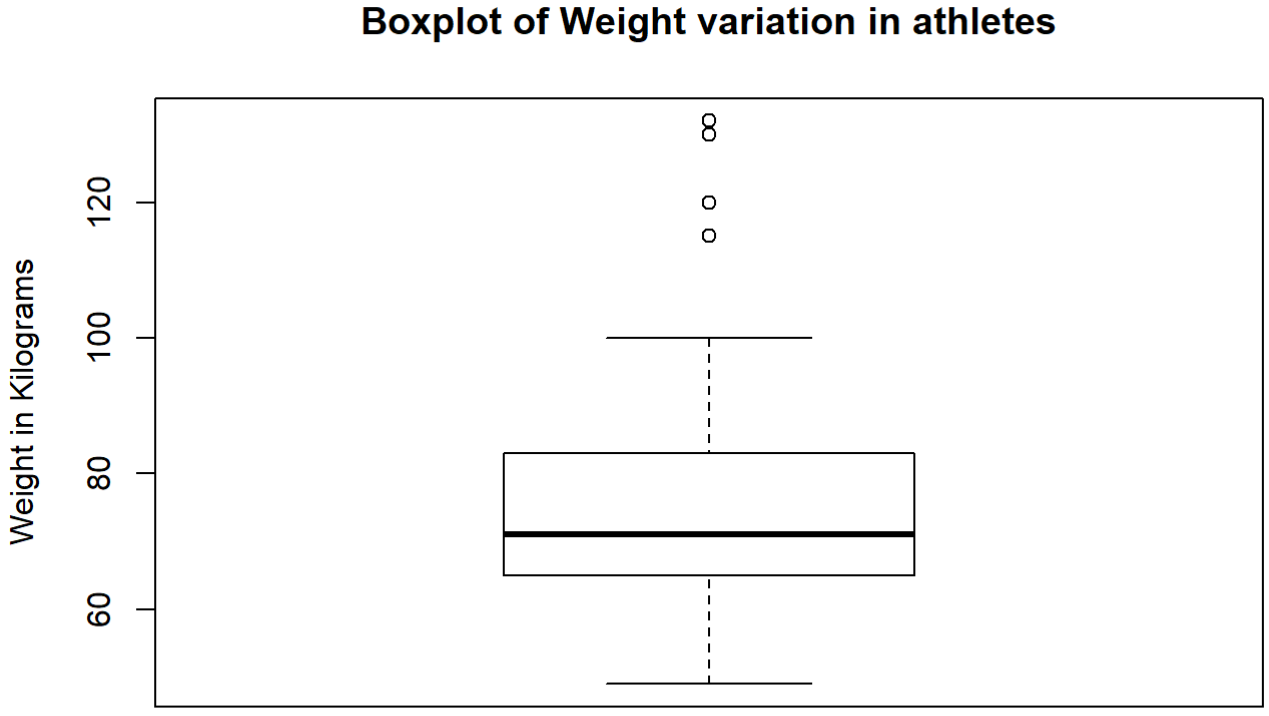
Scan II (Checking for Outliers in Data)

As depicted from the dataset, height, weight, BMI, population, and gdp_per_capita are the numeric variables associated with Olympics dataset. In order to detect outliers among variables, we use boxplot to visualise outliers.

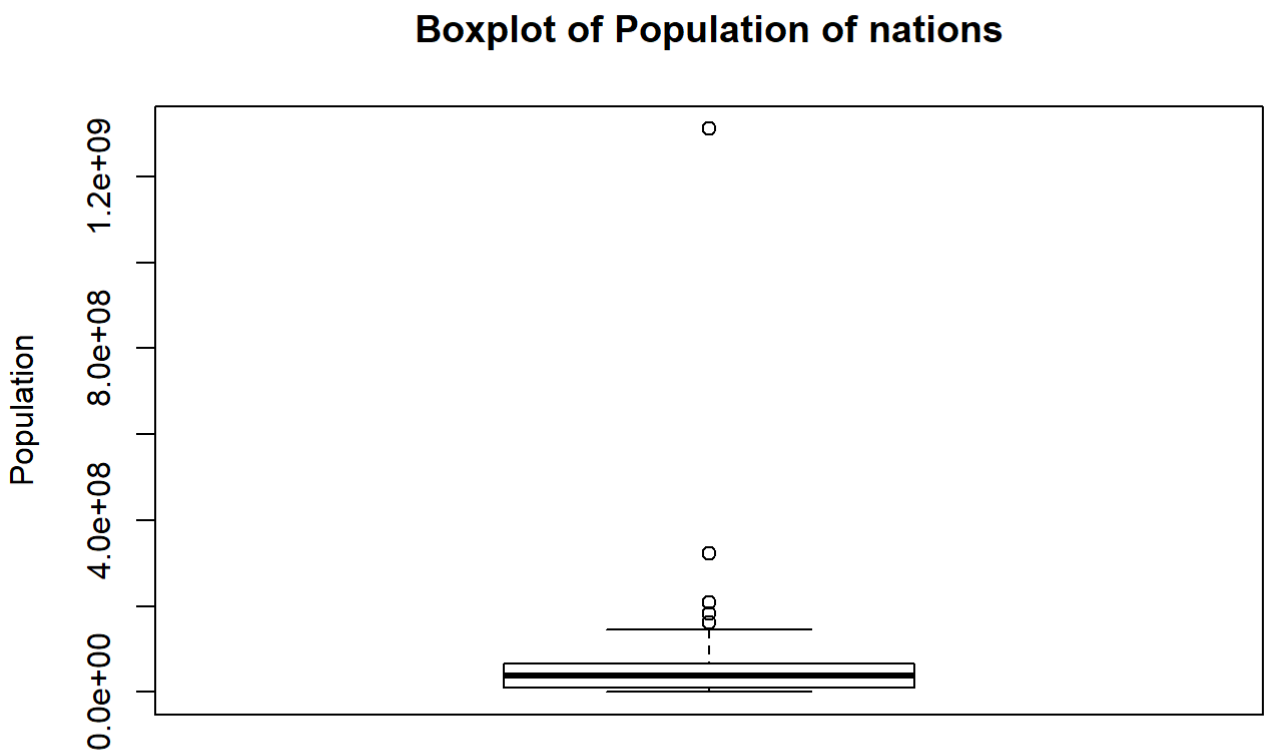
```
# R chunk to visualize Outliers for numeric variables

boxplot(athlete_tidy$height, main = "Boxplot of Height variation in athletes", ylab = "Height
in meters")
```



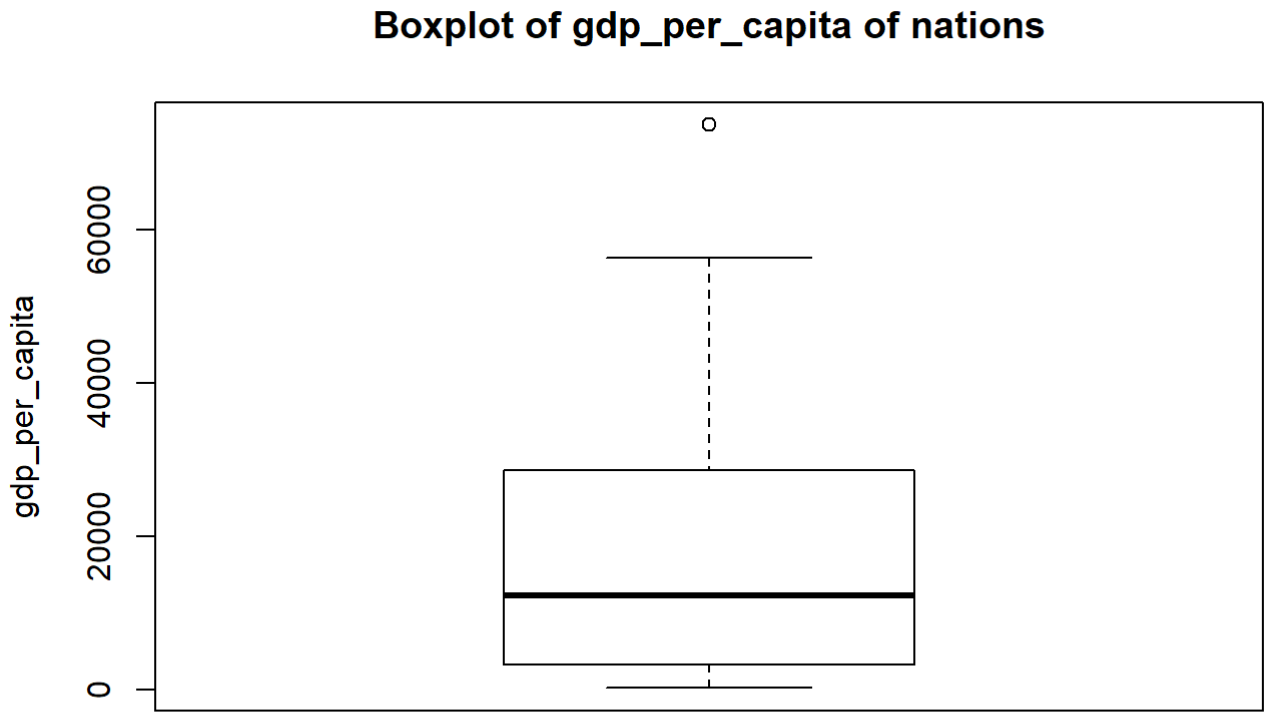

```
boxplot(athlete_tidy$weight, main = "Boxplot of Weight variation in athletes", ylab = "Weight
in Kilograms")
```



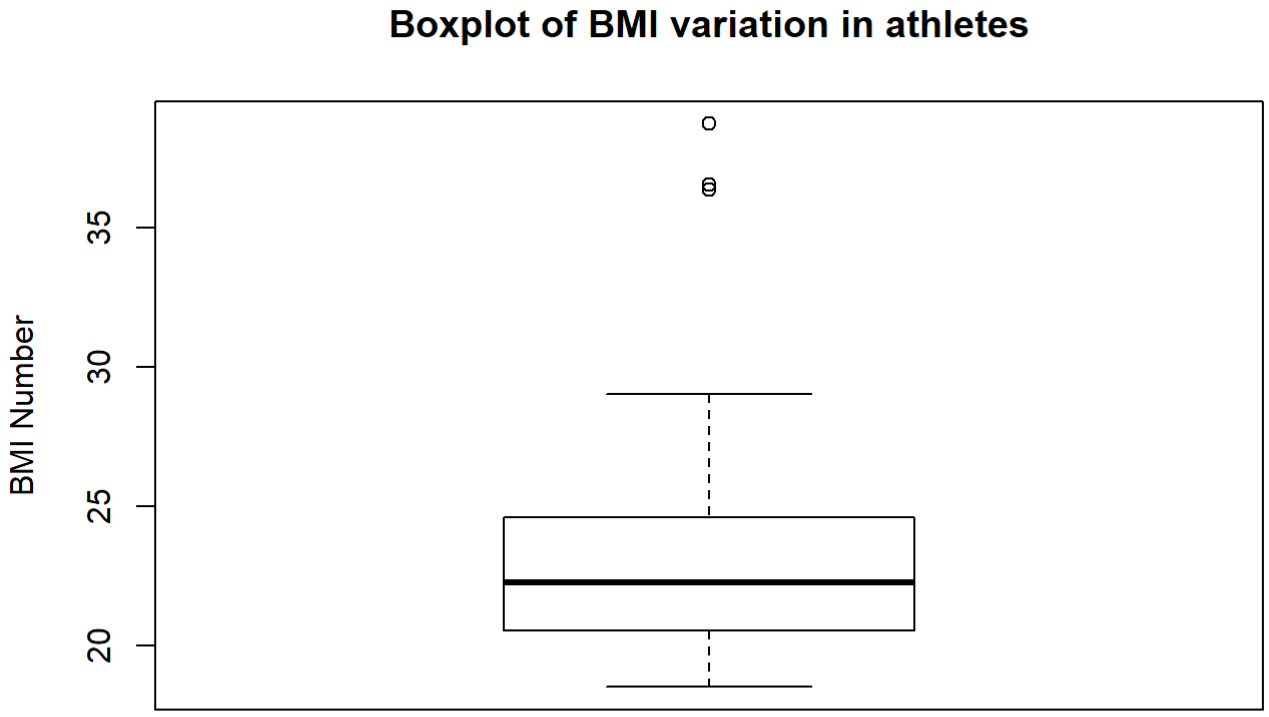
```
boxplot(athlete_tidy$population, main = "Boxplot of Population of nations", ylab = "Populatio
n")
```




```
boxplot(athlete_tidy$gdp_per_capita, main = "Boxplot of gdp_per_capita of nations", ylab = "g
dp_per_capita")
```




```
boxplot(athlete_tidy$BMI, main = "Boxplot of BMI variation in athletes", ylab = "BMI Number")
```



From the above boxplots, we can interpret the following regarding individual variable outliers:

1. Height : Height of the athletes have only 1 outlier which accounts to 0.33% of the heights in dataset.
2. Weight : Weight of the athletes have 4 outlier values which accounts for 1.33% of the weight values of athletes.
3. Population: Population of the nations has 5 outlier values which is 1.67% of total population of given nations.
4. gdp_per_capita : GDP per capita of the nations have only 1 outlier value which is 0.33% of total gdp_per_capita of the countries given.
5. BMI : BMI of the athletes have 4 outlier value which accounts for 1.33% of total BMI of athletes.

Total outliers in the dataset corresponds to 15 outliers

As there are minimum number of outliers present in the dataset and majority outliers are present at right extreme, instead of going for multi variate approach of removing variables, we can remove them using the simple method of capping.

```
# R chunk to define the capping function for outliers

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
}
```

```
# R chunk to remove outliers for height variable using capping method

summary(athlete_tidy$height)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.600   1.720   1.780   1.795   1.860   2.100
```

```
athlete_tidy$height <- athlete_tidy$height %>% cap()

summary(athlete_tidy$height)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.600   1.720   1.780   1.794   1.860   2.070
```

```
# R chunk to remove outliers for weight variable using capping method

summary(athlete_tidy$weight)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   49.00   65.00   71.00   75.01   82.50  132.00
```

```
athlete_tidy$weight <- athlete_tidy$weight %>% cap()

summary(athlete_tidy$weight)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   49.00   65.00   71.00   74.55   82.50  115.00
```

```
# R chunk to remove outliers for population  variable using capping method

summary(athlete_tidy$population)
```

```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## 9.290e+04 1.055e+07 3.693e+07 6.491e+07 6.491e+07 1.311e+09
```

```
athlete_tidy$population <- athlete_tidy$population %>% cap()

summary(athlete_tidy$population)
```

```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
##     92900  10551219  36925634  50514777  64910399 213526093
```

```
# R chunk to remove outliers for gdp_per_capita variable using capping method

summary(athlete_tidy$gdp_per_capita)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   277.1  3430.6 12363.5 19536.0 27905.6 73653.4
```

```
athlete_tidy$gdp_per_capita <- athlete_tidy$gdp_per_capita %>% cap()

summary(athlete_tidy$gdp_per_capita)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   277.1  3430.6 12363.5 19015.8 27905.6 56311.0
```

```
# R chunk to remove outliers for BMI variable using capping method

summary(athlete_tidy$BMI)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   18.52   20.55   22.26   23.10   24.55   38.74
```

```
athlete_tidy$BMI <- athlete_tidy$BMI %>% cap()

summary(athlete_tidy$BMI)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   18.52   20.55   22.26   22.74   24.55   29.04
```

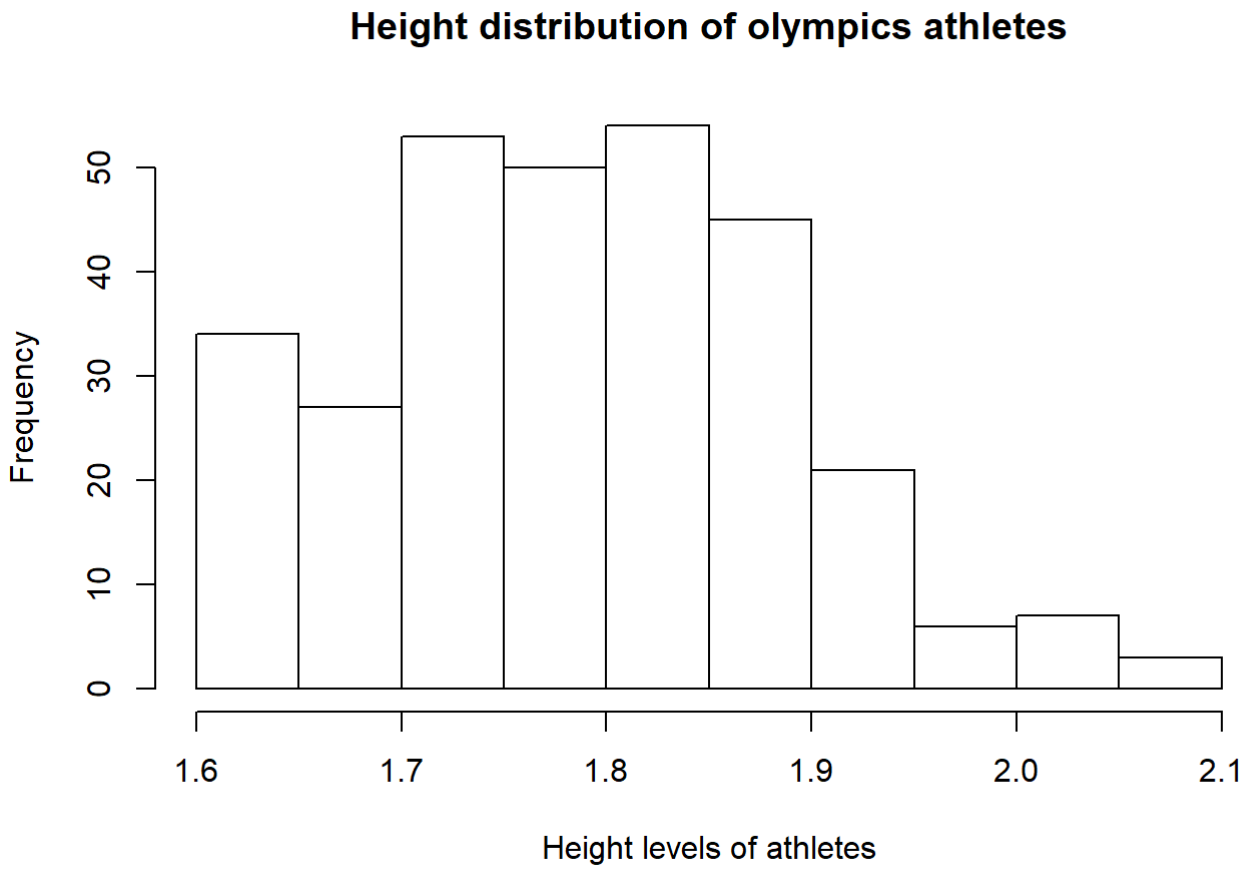
Using capping technique, we were able to remove the outliers from the numeric dataset. The the change can be viewed through the trasformed mean and maximum of individual variables.

Data Transformation

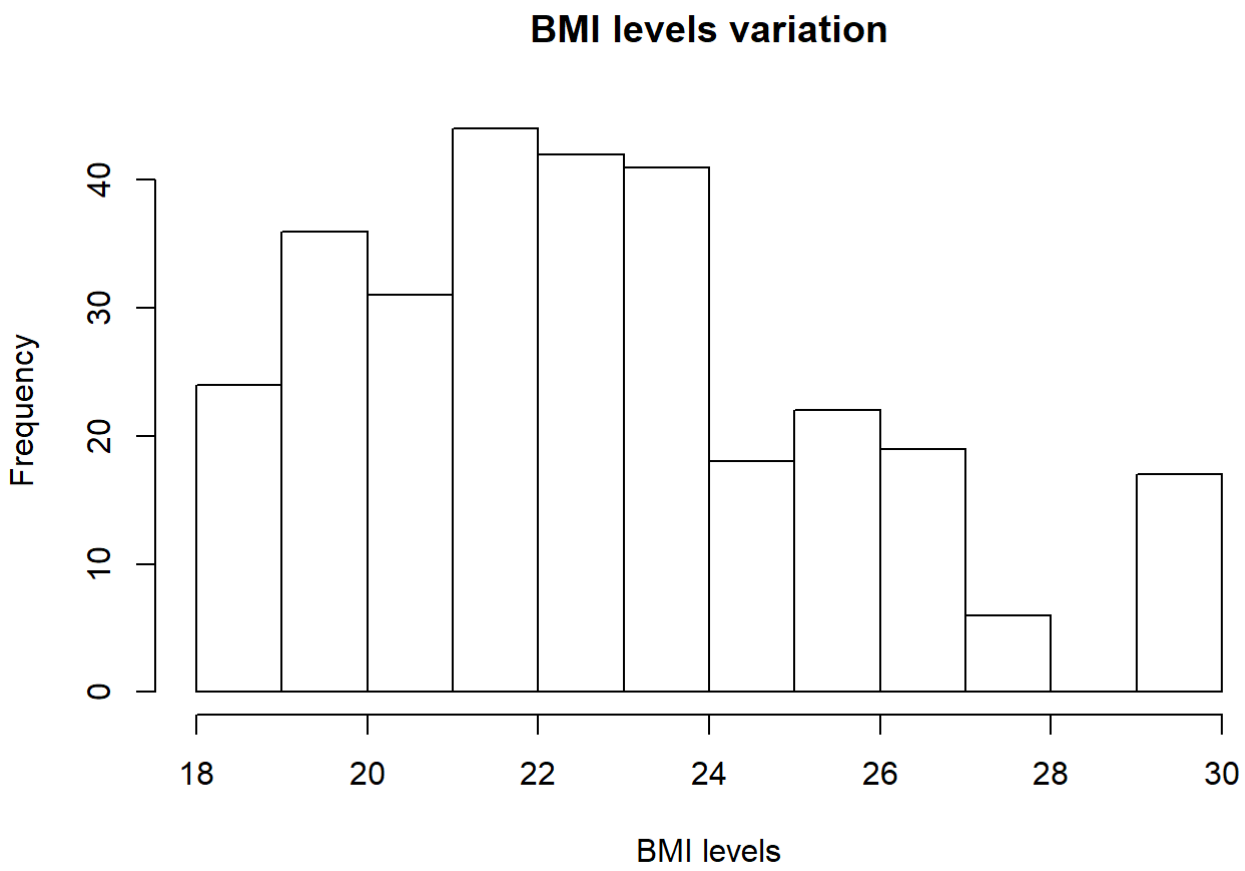
In the case of transformation of numerical variable, not all variables possess same scale of measurment and skewness of the variables also vary posing an issue for data analysis part.

Consider the height and BMI variable:

```
# R chunk to visualise the scale and skewness of height and BMI variable

hist(athlete_tidy$height, main = "Height distribution of olympics athletes", xlab = "Height l
evels of athletes")
```

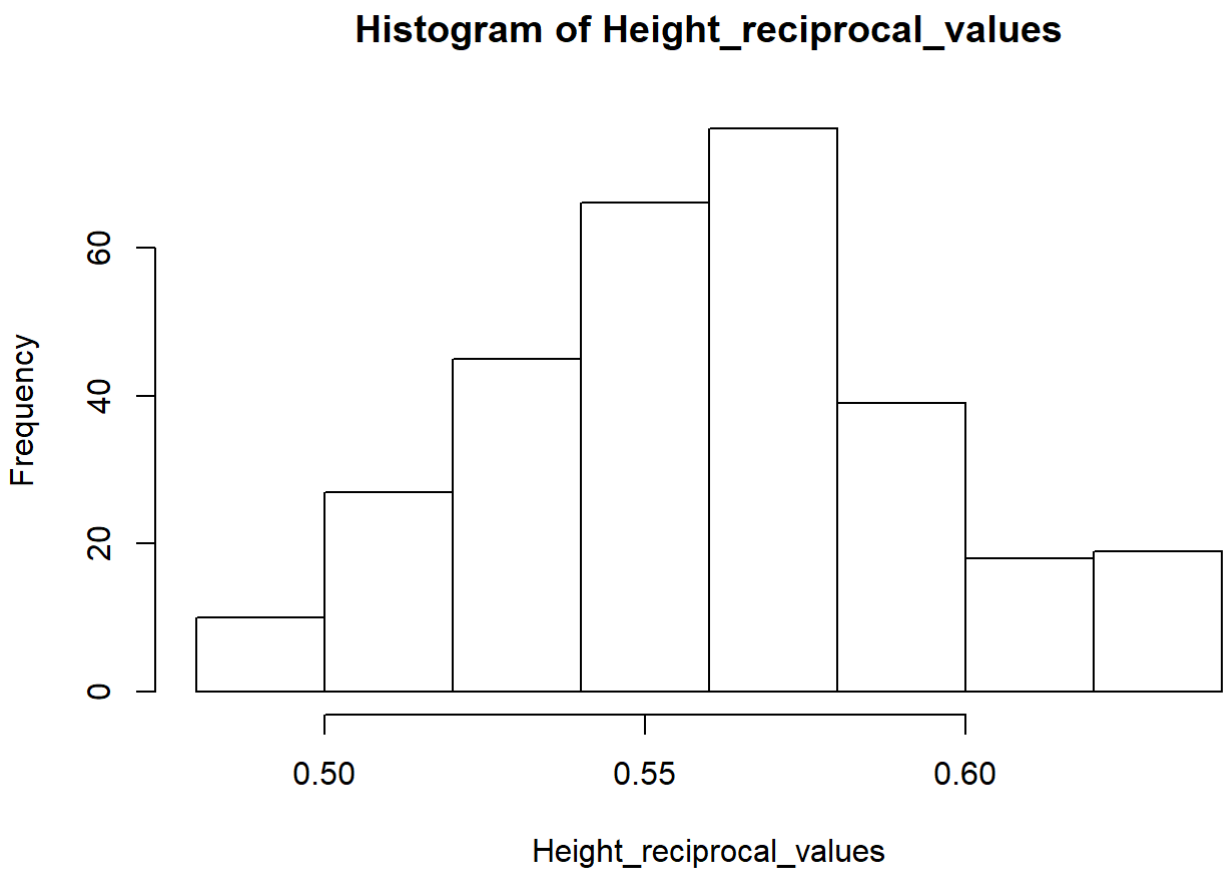



```
hist(athlete_tidy$BMI, main = "BMI levels variation", xlab = "BMI levels")
```

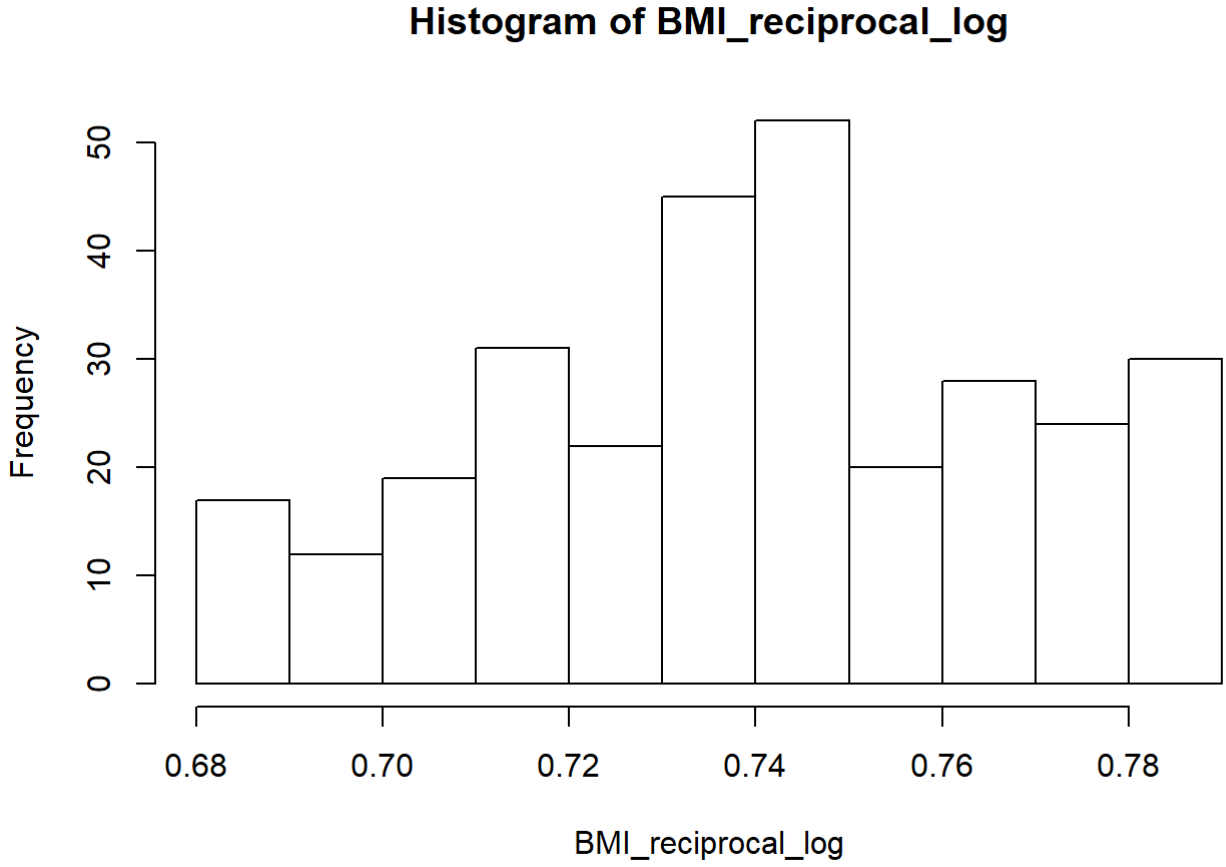


As shown from above histograms, some amount of skewness does exist for height and weight variable. In order to avoid this type of skewness, we shall try to transform the skewed distribution to that of normal distribution. For height variable we shall apply reciprocal function and for BMI we reciprocal logarithm function to convert the distribution to normal.

```
# R chunk to transform height and BMI to normal distribution

Height_reciprocal_values <- 1/(athlete_tidy$height)
hist(Height_reciprocal_values)
```



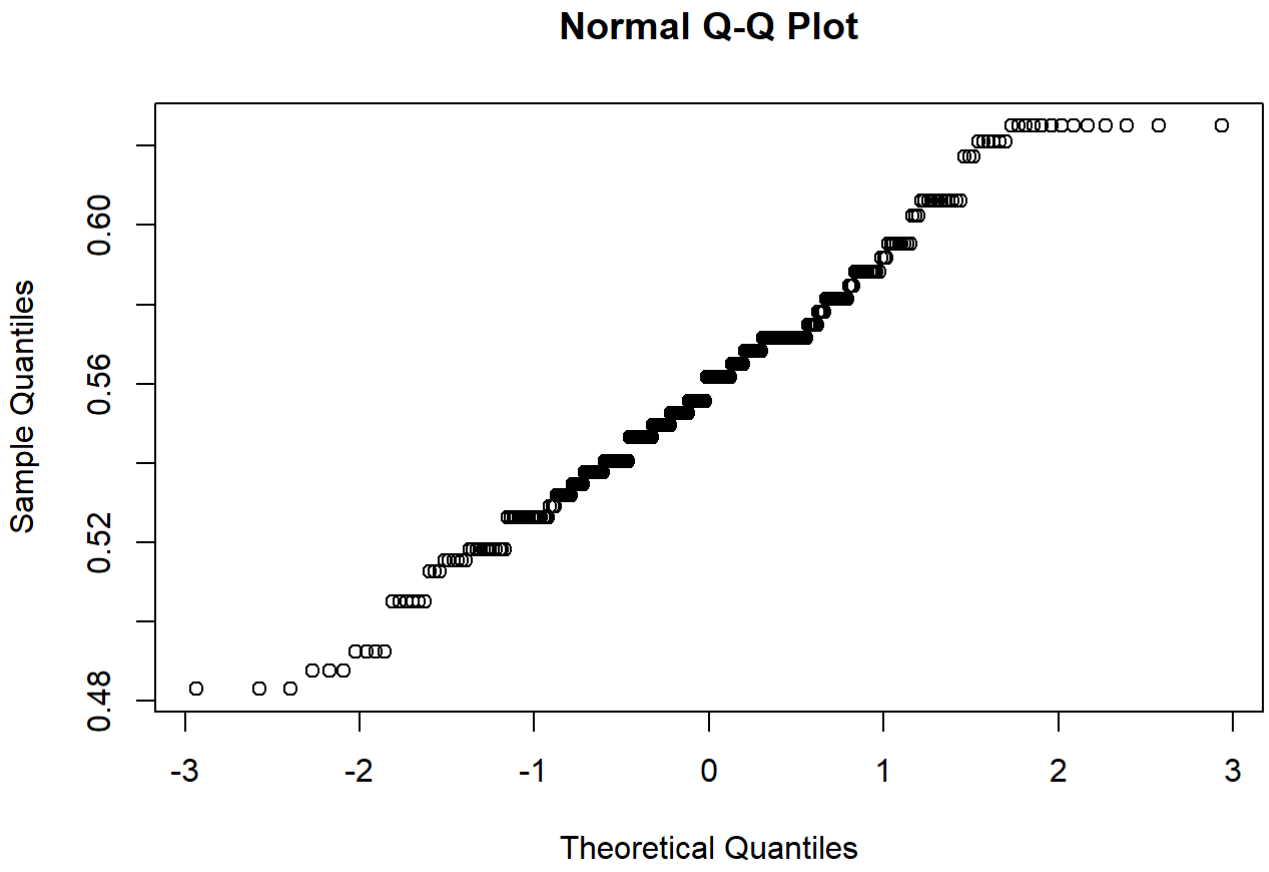

```
BMI_reciprocal_log <- 1/log10(athlete_tidy$BMI)
hist(BMI_reciprocal_log)
```



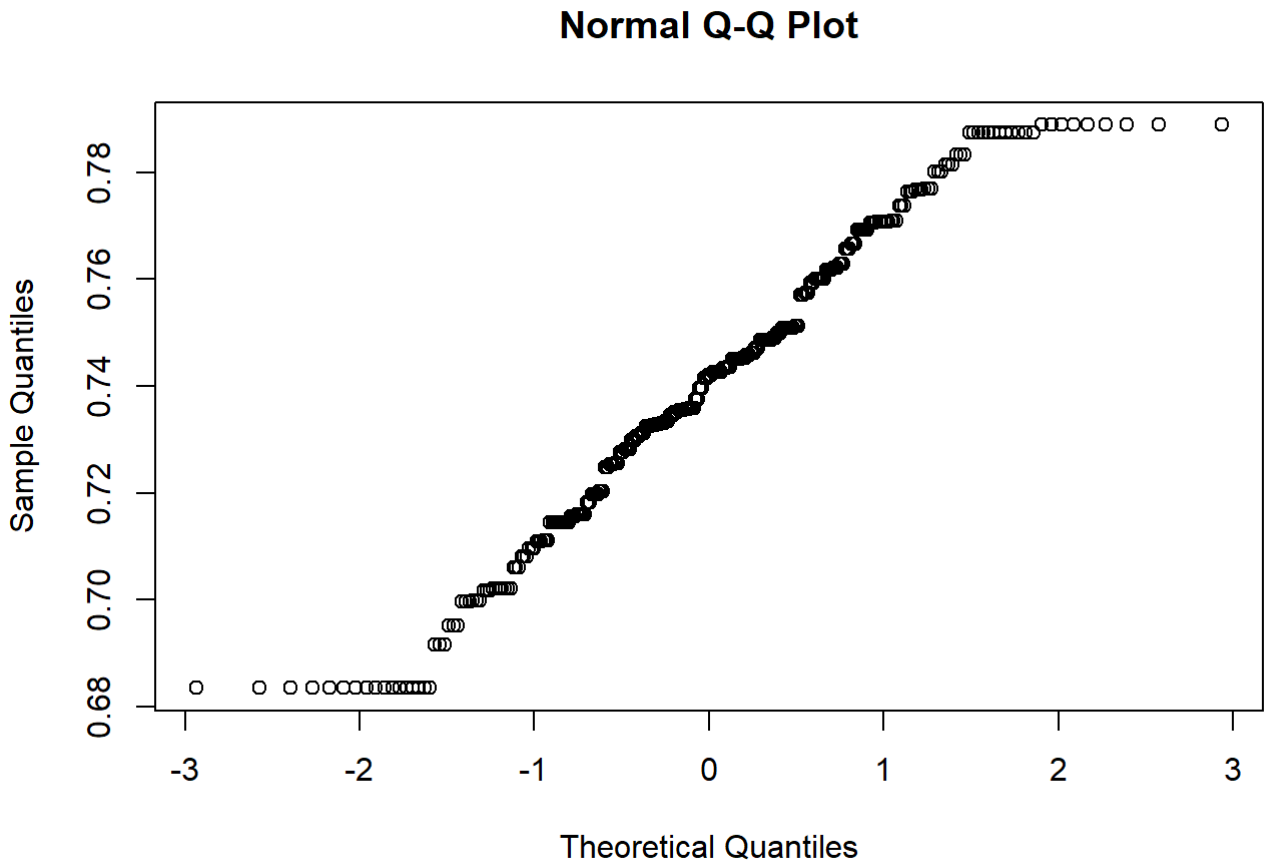
From the above output, we can say that height and BMI variable to some extent has been transformed to normal distribution.

To verify the normal distribution of the variables, we can have a look at their qqnorm() plot which confirms the normality of the variable provided we obtain a near straight diagonal line across x-axis possibly originating from near origin.

```
# R chunk to verify the normality of height and BMI variables

qqnorm(Height_reciprocal_values)
```




```
qqnorm(BMI_reciprocal_log)
```



As purported from above qqnorm() plot, height and BMI variables to some extent form a straight diagonal line across x-axis originating from near origin. So, we can say that height and BMI has been transformed to normal distribution.

All in all, in this report, we were able to import dataset, merge them, analyze each variable, tidy the data, checked for missing values, scan for outliers and finally transform the data.