Data Wrangling

Code

Practical assessment 2

Hardik Bhavesh Ramparia

Setup

Hide

```
# Load the necessary packages

library(kableExtra)
library(magrittr)
library(tidyr)
library(dplyr)
library(readr)
library(MVN)
library(forecast)
library(caret)
```

Student names, numbers and percentage of contributions

Group information

Student name	Student number	Percentage of contribution
Hardik Bhavesh Ramparia	S4105620	100

Executive Summary

The report contains 2 data sets, one of the data sets is a merged data set from the World Happiness Report of around 158 countries for 3 years, 2015, 2016 and 2017. The second data set is the GDP of 271 countries for the year 1960 to 2023.

The idea of using the two data sets is to compare the Happiness Score per GDP.

- The first step is to combine the 3 data sets of the year 2015-2017. The data were merged by appending the data one below the other.
- The GDP data was also loaded locally and two columns that are not useful for data analysis were dropped.
- In the understand section, the Happiness rank variable was converted to a factor and mistyped column in the GDP data was removed, because it contained only NA values.
- In the Tidy and Manipulate Data I section, the GDP data was pivoted longer because it contained the year variable as columns. It was then joined with the Happiness data.
- In Tidy and Manipulate Data II, a new variable Happiness/GDP was mutated in the data frame to easily understand which country has the highest happiness score per GDP. The country Comoros came out to have the highest Happiness/GDP of 4.095113e-09.

- In the Scan I section, the variables were analysed for missing values and the row values of year not 2015-2017 were dropped as they did not have any happiness data values.
- In the Transform section, different transformation techniques such as BoxCox, log and square were used to convert the data into a normal distribution.

Data

World Happiness Report - The scores are from nationally representative samples for the years 2013-2016 and use the Gallup weights to make the estimates representative. The columns following the happiness score estimate the extent to which each of six factors – economic production, social support, life expectancy, freedom, absence of corruption, and generosity – contribute to making life evaluations higher in each country than they are in Dystopia, a hypothetical country that has values equal to the world's lowest national averages for each of the six factors.

Variable Description:

- Country - Name of the country.
- Happiness Rank - Rank of the country based on the Happiness Score.
- Happiness Score - A metric measured in 2015 by asking the sampled people the question: "How would you rate your happiness".
- Economy (GDP per Capita) - The extent to which GDP contributes to the calculation of the Happiness Score.
- Family - The extent to which Family contributes to the calculation of the Happiness Score.
- Health (Life Expectancy) - The extent to which Life expectancy contributed to the calculation of the Happiness Score.
- Freedom - The extent to which Freedom contributed to the calculation of the Happiness Score.
- Trust (Government Corruption) - The extent to which Perception of Corruption contributes to Happiness Score.

World Happiness Data URL - https://www.kaggle.com/datasets/unsdsn/world-happiness (https://www.kaggle.com/datasets/unsdsn/world-happiness)

World GDP Data - The value of GDP in US Dollars for around 271 countries, for the year 1960 to 2023.

Variable Description:

- Country - Name of the country.
- Country Code - Abbreviation of the country.
- 1960 - 2023 - Columns of year that has GDP values of the corresponding country.

World GDP Data URL - https://data.worldbank.org/indicator/NY.GDP.MKTP.CD (https://data.worldbank.org/indicator/NY.GDP.MKTP.CD)

Hide

```
# loading the happiness data files
# loading the 2015 happiness data
hap2015 <- read_csv('C:/Users/rampa/Hardik/RMIT/SEM 1/Data Wrangling/Assignment 2/2015 Happin
ess Data.csv') %>%
  select(`Country`, `Happiness Rank`, `Happiness Score`, `Economy (GDP per Capita)`, `Family
`, `Health (Life Expectancy)`, `Freedom`, `Trust (Government Corruption)`, `Generosity`)
```

```
Rows: 158 Columns: 12
── Column specification ────────────────────────────────────────────────────────
────────────────
Delimiter: ","
chr  (2): Country, Region
dbl (10): Happiness Rank, Happiness Score, Standard Error, Economy (GDP per Capita), Family,
Health (Life E...

ℹ Use `spec()` to retrieve the full column specification for this data.
ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

Hide

```
# loading the 2016 happiness data
hap2016 <- read_csv('C:/Users/rampa/Hardik/RMIT/SEM 1/Data Wrangling/Assignment 2/2016 Happin
ess Data.csv') %>%
  select(`Country`, `Happiness Rank`, `Happiness Score`, `Economy (GDP per Capita)`, `Family
`, `Health (Life Expectancy)`, `Freedom`, `Trust (Government Corruption)`, `Generosity`)
```

```
Rows: 157 Columns: 13
── Column specification ────────────────────────────────────────────────────────
────────────────
Delimiter: ","
chr  (2): Country, Region
dbl (11): Happiness Rank, Happiness Score, Lower Confidence Interval, Upper Confidence Interv
al, Economy (G...

ℹ Use `spec()` to retrieve the full column specification for this data.
ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

Hide

```
# loading the 2017 happiness data
hap2017 <- read_csv('C:/Users/rampa/Hardik/RMIT/SEM 1/Data Wrangling/Assignment 2/2017 Happin
ess Data.csv') %>%
  select(`Country`, `Happiness.Rank`, `Happiness.Score`, `Economy..GDP.per.Capita.`, `Family
`, `Health..Life.Expectancy.`, `Freedom`, `Trust..Government.Corruption.`, `Generosity`)
```

```
Rows: 155 Columns: 12
── Column specification ────────────────────────────────────────────────────────
────────────────
Delimiter: ","
chr  (1): Country
dbl (11): Happiness.Rank, Happiness.Score, Whisker.high, Whisker.low, Economy..GDP.per.Capit
a., Family, Hea...

ℹ Use `spec()` to retrieve the full column specification for this data.
ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

Hide

```
# renaming the 2017 data frame columns
hap2017 <- hap2017 %>%
  rename('Happiness Rank' = 'Happiness.Rank' ,
         'Happiness Score' = 'Happiness.Score',
         'Economy (GDP per Capita)' = 'Economy..GDP.per.Capita.',
         'Health (Life Expectancy)' = 'Health..Life.Expectancy.',
         'Trust (Government Corruption)' = 'Trust..Government.Corruption.')

# adding the year column to all the data sets
hap2015 <- hap2015 %>% mutate(Year = 2015)
hap2016 <- hap2016 %>% mutate(Year = 2016)
hap2017 <- hap2017 %>% mutate(Year = 2017)

# merging the data frames by binding the rows
happiness_data <- bind_rows(hap2015, hap2016, hap2017)
glimpse(happiness_data)
```

```
Rows: 470
Columns: 10
$ Country                          <chr> "Switzerland", "Iceland", "Denmark", "Norway", "Canad
a", "Finland", "…
$ `Happiness Rank`                 <dbl> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 1
6, 17, 18, 19, 20…
$ `Happiness Score`                <dbl> 7.587, 7.561, 7.527, 7.522, 7.427, 7.406, 7.378, 7.36
4, 7.286, 7.284,…
$ `Economy (GDP per Capita)`       <dbl> 1.39651, 1.30232, 1.32548, 1.45900, 1.32629, 1.29025,
1.32944, 1.3317…
$ Family                           <dbl> 1.34951, 1.40223, 1.36058, 1.33095, 1.32261, 1.31826,
1.28017, 1.2890…
$ `Health (Life Expectancy)`       <dbl> 0.94143, 0.94784, 0.87464, 0.88521, 0.90563, 0.88911,
0.89284, 0.9108…
$ Freedom                          <dbl> 0.66557, 0.62877, 0.64938, 0.66973, 0.63297, 0.64169,
0.61576, 0.6598…
$ `Trust (Government Corruption)` <dbl> 0.41978, 0.14145, 0.48357, 0.36503, 0.32957, 0.41372,
0.31814, 0.4384…
$ Generosity                       <dbl> 0.29678, 0.43630, 0.34139, 0.34699, 0.45811, 0.23351,
0.47610, 0.3626…
$ Year                             <dbl> 2015, 2015, 2015, 2015, 2015, 2015, 2015, 2015, 2015,
2015, 2015, 201…
```

Hide

```
# loading the GDP data
gdp_data <- read_csv('C:/Users/rampa/Hardik/RMIT/SEM 1/Data Wrangling/Assignment 2/World GDP
Data.csv', skip=4)
```

```
New names:
• `` -> `...69`
Rows: 266 Columns: 69
── Column specification ────────────────────────────────────────────────────────
────────────────
Delimiter: ","
chr  (4): Country Name, Country Code, Indicator Name, Indicator Code
dbl (64): 1960, 1961, 1962, 1963, 1964, 1965, 1966, 1967, 1968, 1969, 1970, 1971, 1972, 1973,
1974, 1975, 1...
lgl  (1): ...69

ℹ Use `spec()` to retrieve the full column specification for this data.
ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

Hide

```
# the Indicator name and Indicator code columns are of no use, we can drop them
gdp_data <- gdp_data %>%
  select(-`Indicator Name`, -`Indicator Code`)

head(gdp_data)
```

Country Name <chr>	**Country Code** <chr>	**1960** <dbl>	**1961** <dbl>	**1962** <dbl>	
Aruba	ABW	*NA*	*NA*	*NA*	
Africa Eastern and Southern	AFE	21216962290	22307471356	23702472100	257
Afghanistan	AFG	*NA*	*NA*	*NA*	
Africa Western and Central	AFW	11884128412	12685662255	13606829297	144
Angola	AGO	*NA*	*NA*	*NA*	
Albania	ALB	*NA*	*NA*	*NA*	

6 rows | 1-6 of 67 columns

Show

Understand

Hide

```
# understanding the structure of happiness data
str(happiness_data)
```

```
tibble [470 × 10] (S3: tbl_df/tbl/data.frame)
 $ Country                      : chr [1:470] "Switzerland" "Iceland" "Denmark" "Norway" ...
 $ Happiness Rank               : num [1:470] 1 2 3 4 5 6 7 8 9 10 ...
 $ Happiness Score              : num [1:470] 7.59 7.56 7.53 7.52 7.43 ...
 $ Economy (GDP per Capita)     : num [1:470] 1.4 1.3 1.33 1.46 1.33 ...
 $ Family                       : num [1:470] 1.35 1.4 1.36 1.33 1.32 ...
 $ Health (Life Expectancy)     : num [1:470] 0.941 0.948 0.875 0.885 0.906 ...
 $ Freedom                      : num [1:470] 0.666 0.629 0.649 0.67 0.633 ...
 $ Trust (Government Corruption): num [1:470] 0.42 0.141 0.484 0.365 0.33 ...
 $ Generosity                   : num [1:470] 0.297 0.436 0.341 0.347 0.458 ...
 $ Year                         : num [1:470] 2015 2015 2015 2015 2015 ...
```

Hide

```
# converting the Happiness Rank variable to a factor variable
happiness_data <- happiness_data %>% mutate('Happiness Rank' = factor('Happiness Rank', level
s=1:158, ordered=TRUE))

# checking the gdp_data data frame
str(gdp_data)
```

```
tibble [266 × 67] (S3: tbl_df/tbl/data.frame)
 $ Country Name: chr [1:266] "Aruba" "Africa Eastern and Southern" "Afghanistan" "Africa West
ern and Central" ...
 $ Country Code: chr [1:266] "ABW" "AFE" "AFG" "AFW" ...
 $ 1960        : num [1:266] NA 2.12e+10 NA 1.19e+10 NA ...
 $ 1961        : num [1:266] NA 2.23e+10 NA 1.27e+10 NA ...
 $ 1962        : num [1:266] NA 2.37e+10 NA 1.36e+10 NA ...
 $ 1963        : num [1:266] NA 2.58e+10 NA 1.44e+10 NA ...
 $ 1964        : num [1:266] NA 2.80e+10 NA 1.58e+10 NA ...
 $ 1965        : num [1:266] NA 3.04e+10 NA 1.69e+10 NA ...
 $ 1966        : num [1:266] NA 3.3e+10 NA 1.8e+10 NA ...
 $ 1967        : num [1:266] NA 3.59e+10 NA 1.65e+10 NA ...
 $ 1968        : num [1:266] NA 3.87e+10 NA 1.70e+10 NA ...
 $ 1969        : num [1:266] NA 4.30e+10 NA 1.93e+10 NA ...
 $ 1970        : num [1:266] NA 4.37e+10 NA 2.67e+10 NA ...
 $ 1971        : num [1:266] NA 4.76e+10 NA 2.45e+10 NA ...
 $ 1972        : num [1:266] NA 5.16e+10 NA 2.95e+10 NA ...
 $ 1973        : num [1:266] NA 6.66e+10 NA 3.69e+10 NA ...
 $ 1974        : num [1:266] NA 8.17e+10 NA 4.97e+10 NA ...
 $ 1975        : num [1:266] NA 8.69e+10 NA 5.73e+10 NA ...
 $ 1976        : num [1:266] NA 8.83e+10 NA 6.84e+10 NA ...
 $ 1977        : num [1:266] NA 9.89e+10 NA 7.18e+10 NA ...
 $ 1978        : num [1:266] NA 1.11e+11 NA 7.88e+10 NA ...
 $ 1979        : num [1:266] NA 1.30e+11 NA 9.67e+10 NA ...
 $ 1980        : num [1:266] NA 1.68e+11 NA 1.21e+11 NA ...
 $ 1981        : num [1:266] NA 1.78e+11 NA 2.17e+11 NA ...
 $ 1982        : num [1:266] NA 1.69e+11 NA 1.96e+11 NA ...
 $ 1983        : num [1:266] NA 1.80e+11 NA 1.51e+11 NA ...
 $ 1984        : num [1:266] NA 1.66e+11 NA 1.31e+11 NA ...
 $ 1985        : num [1:266] NA 1.43e+11 NA 1.38e+11 NA ...
 $ 1986        : num [1:266] 4.06e+08 1.55e+11 NA 1.09e+11 NA ...
 $ 1987        : num [1:266] 4.88e+08 1.86e+11 NA 1.12e+11 NA ...
 $ 1988        : num [1:266] 5.97e+08 2.02e+11 NA 1.11e+11 NA ...
 $ 1989        : num [1:266] 6.96e+08 2.14e+11 NA 1.04e+11 NA ...
 $ 1990        : num [1:266] 7.65e+08 2.51e+11 NA 1.24e+11 NA ...
 $ 1991        : num [1:266] 8.72e+08 2.74e+11 NA 1.30e+11 NA ...
 $ 1992        : num [1:266] 9.59e+08 2.39e+11 NA 1.25e+11 NA ...
 $ 1993        : num [1:266] 1.08e+09 2.40e+11 NA 1.30e+11 5.88e+09 ...
 $ 1994        : num [1:266] 1.25e+09 2.44e+11 NA 1.35e+11 4.43e+09 ...
 $ 1995        : num [1:266] 1.32e+09 2.73e+11 NA 2.07e+11 5.54e+09 ...
 $ 1996        : num [1:266] 1.38e+09 2.73e+11 NA 2.63e+11 6.54e+09 ...
 $ 1997        : num [1:266] 1.53e+09 2.88e+11 NA 2.76e+11 7.68e+09 ...
 $ 1998        : num [1:266] 1.67e+09 2.69e+11 NA 2.96e+11 6.51e+09 ...
 $ 1999        : num [1:266] 1.72e+09 2.65e+11 NA 1.39e+11 6.15e+09 ...
 $ 2000        : num [1:266] 1.87e+09 2.87e+11 3.52e+09 1.42e+11 9.13e+09 ...
 $ 2001        : num [1:266] 1.90e+09 2.61e+11 2.81e+09 1.49e+11 8.94e+09 ...
 $ 2002        : num [1:266] 1.96e+09 2.68e+11 3.83e+09 1.79e+11 1.53e+10 ...
 $ 2003        : num [1:266] 2.04e+09 3.56e+11 4.52e+09 2.07e+11 1.78e+10 ...
 $ 2004        : num [1:266] 2.25e+09 4.43e+11 5.22e+09 2.56e+11 2.36e+10 ...
 $ 2005        : num [1:266] 2.36e+09 5.17e+11 6.20e+09 3.12e+11 3.70e+10 ...
 $ 2006        : num [1:266] 2.47e+09 5.80e+11 6.97e+09 3.97e+11 5.24e+10 ...
 $ 2007        : num [1:266] 2.68e+09 6.66e+11 9.75e+09 4.66e+11 6.53e+10 ...
 $ 2008        : num [1:266] 2.84e+09 7.14e+11 1.01e+10 5.68e+11 8.85e+10 ...
 $ 2009        : num [1:266] 2.55e+09 7.15e+11 1.24e+10 5.09e+11 7.03e+10 ...
 $ 2010        : num [1:266] 2.45e+09 8.49e+11 1.59e+10 5.99e+11 8.38e+10 ...
```

```
 $ 2011        : num [1:266] 2.64e+09 9.45e+11 1.78e+10 6.82e+11 1.12e+11 ...
 $ 2012        : num [1:266] 2.62e+09 9.53e+11 1.99e+10 7.38e+11 1.28e+11 ...
 $ 2013        : num [1:266] 2.73e+09 9.63e+11 2.01e+10 8.34e+11 1.32e+11 ...
 $ 2014        : num [1:266] 2.79e+09 9.80e+11 2.05e+10 8.95e+11 1.36e+11 ...
 $ 2015        : num [1:266] 2.96e+09 8.99e+11 1.91e+10 7.69e+11 9.05e+10 ...
 $ 2016        : num [1:266] 2.98e+09 8.30e+11 1.81e+10 6.92e+11 5.28e+10 ...
 $ 2017        : num [1:266] 3.09e+09 9.40e+11 1.88e+10 6.86e+11 7.37e+10 ...
 $ 2018        : num [1:266] 3.28e+09 1.01e+12 1.81e+10 7.68e+11 7.95e+10 ...
 $ 2019        : num [1:266] 3.40e+09 1.01e+12 1.88e+10 8.24e+11 7.09e+10 ...
 $ 2020        : num [1:266] 2.56e+09 9.29e+11 2.00e+10 7.87e+11 4.85e+10 ...
 $ 2021        : num [1:266] 3.10e+09 1.09e+12 1.43e+10 8.46e+11 6.65e+10 ...
 $ 2022        : num [1:266] 3.54e+09 1.18e+12 1.45e+10 8.77e+11 1.04e+11 ...
 $ 2023        : num [1:266] NA 1.24e+12 NA 7.97e+11 8.47e+10 ...
 $ ...69       : logi [1:266] NA NA NA NA NA NA ...
```

Hide

```
# here we can see that there is a column '...69' that holds no value, we cna remove it from t
he data
gdp_data <- gdp_data %>% select(-`...69`)
```

In this section, we understand the structures of our data sets. In the Happiness data, we can see that the Country variable is of character type. We have converted the Happiness Rank variable into an ordered factor so that it shows an order of rank.

In the GDP Data, the variables were of the correct data type. On detailed inspection, we could see a column "…69" that contained only NA values. This column was dropped as it was of no use to further data analysis.

Tidy & Manipulate Data I

Hide

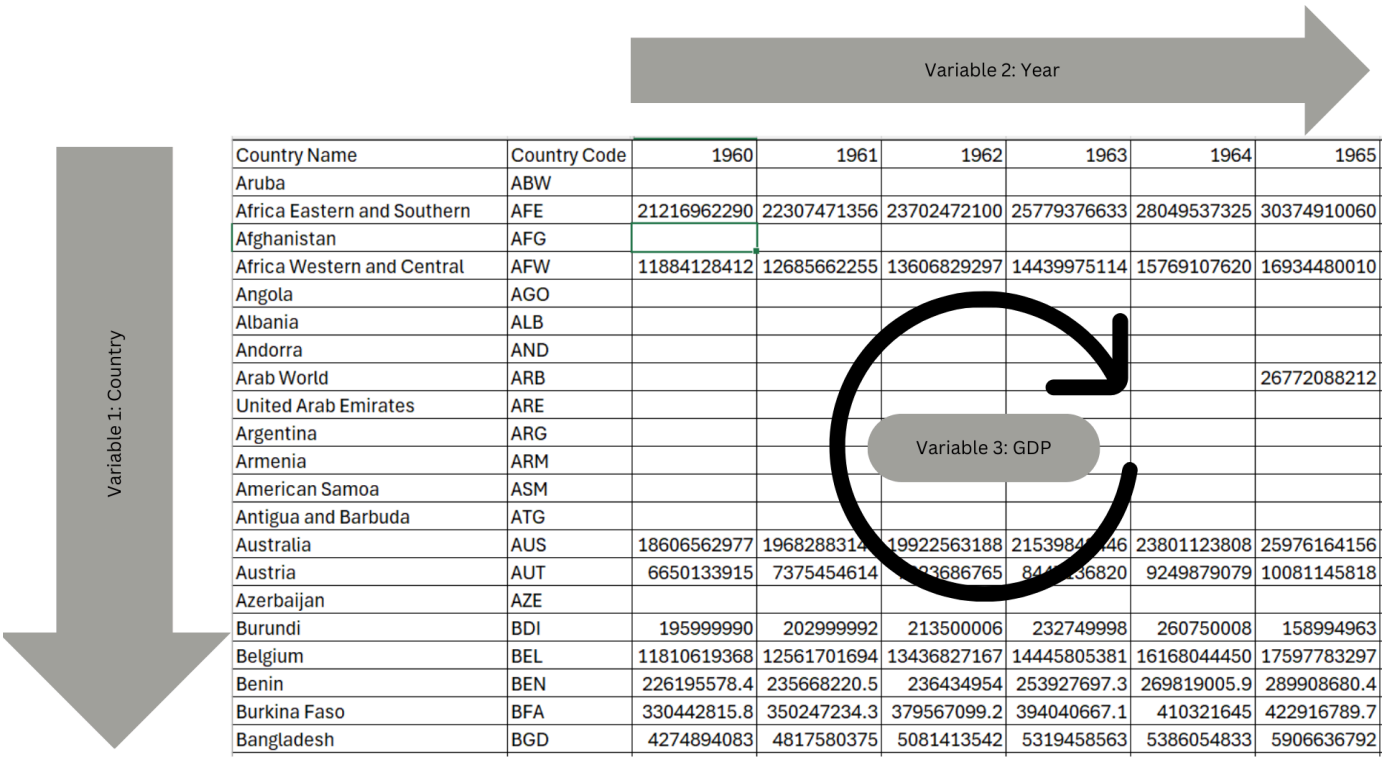
```
# since this GDP data is untidy, we need to pivot longer
gdp_long <- gdp_data %>% pivot_longer(names_to="Year", values_to="GDP", cols=3:66)
gdp_long <- gdp_long %>% rename('Country' = 'Country Name')

# converting the Year column in gdp_long to numeric data type
gdp_long$Year <- as.numeric(gdp_long$Year)

# joining the happiness and GDP data set
data <- left_join(gdp_long, happiness_data, by=c('Country', 'Year'))

glimpse(data)
```

```
Rows: 17,024
Columns: 12
$ Country                          <chr> "Aruba", "Aruba", "Aruba", "Aruba", "Aruba", "Aruba",
"Aruba", "Aruba…
$ `Country Code`                   <chr> "ABW", "ABW", "ABW", "ABW", "ABW", "ABW", "ABW", "AB
W", "ABW", "ABW",…
$ Year                             <dbl> 1960, 1961, 1962, 1963, 1964, 1965, 1966, 1967, 1968,
1969, 1970, 197…
$ GDP                              <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N
A, NA, NA, NA, N…
$ `Happiness Rank`                 <ord> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N
A, NA, NA, NA, N…
$ `Happiness Score`                <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N
A, NA, NA, NA, N…
$ `Economy (GDP per Capita)`       <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N
A, NA, NA, NA, N…
$ Family                           <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N
A, NA, NA, NA, N…
$ `Health (Life Expectancy)`       <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N
A, NA, NA, NA, N…
$ Freedom                          <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N
A, NA, NA, NA, N…
$ `Trust (Government Corruption)`  <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N
A, NA, NA, NA, N…
$ Generosity                       <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N
A, NA, NA, NA, N…
```

According to tidy data rules outlined by Hadley Wickham (Hadley Wickham and Grolemund (2016)),

- Each variable must have its own column.
- Each observation must have its own row.
- Each value must have its own cell.

In our GDP data, the columns have the values of the Year variable, from 1960 to 2023. This violates tidy data principles. An illustration is given below:



Country Name	Country Code	1960	1961	1962	1963	1964	1965
Aruba	ABW						
Africa Eastern and Southern	AFE	21216962290	22307471356	23702472100	25779376633	28049537325	30374910060
Afghanistan	AFG						
Africa Western and Central	AFW	11884128412	12685662255	13606829297	14439975114	15769107620	16934480010
Angola	AGO						
Albania	ALB						
Andorra	AND						
Arab World	ARB						26772088212
United Arab Emirates	ARE						
Argentina	ARG						
Armenia	ARM						
American Samoa	ASM						
Antigua and Barbuda	ATG						
Australia	AUS	18606562977	19682883146	19922563188	21539846446	23801123808	25976164156
Austria	AUT	6650133915	7375454614	7823686765	8446436820	9249879079	10081145818
Azerbaijan	AZE						
Burundi	BDI	195999990	202999992	213500006	232749998	260750008	158994963
Belgium	BEL	11810619368	12561701694	13436827167	14445805381	16168044450	17597783297
Benin	BEN	226195578.4	235668220.5	236434954	253927697.3	269819005.9	289908680.4
Burkina Faso	BFA	330442815.8	350247234.3	379567099.2	394040667.1	410321645	422916789.7
Bangladesh	BGD	4274894083	4817580375	5081413542	5319458563	5386054833	5906636792

We need to pivot longer the year variable and keep the GDP as the values. This can be done by using the pivot_longer() function:

pivot_longer(data, names_to, values_to, cols)

The names_to argument specifies the name of the new column that will be created from the column names of the original data. The values_to argument specifies the name of the new column that will be created from the cell values of the original data. The cols argument refers to the columns that should be pivoted into a longer format.

Tidy & Manipulate Data II

Hide

```
# to compare the happiness with the GDP, we can have a happiness/GDP variable for the data fr
ame
data <- data %>%
  mutate(`Happiness/GDP` = `Happiness Score` / `GDP`)

# look at the highest happiness/gdp country
data %>% filter(`Happiness/GDP` == max(`Happiness/GDP`, na.rm = TRUE)) %>% select(`Country`,
`Happiness/GDP`)
```

Country <chr>	**Happiness/GDP** <dbl>
Comoros	4.095113e-09

1 row

We have merged the happiness score data and the GDP data. To actaully compare the rate of Happiness Score per GDP, we can create another column that divides the happiness score of the particular country with the GDP of the country in that year.

We use the mutate() function for this manipulation:

mutate(data, new_col = existing1 - existing2)

We can see that the country Comoros has the highest Happiness Score per GDP of 4.095113e-09. This variable can be an easy way to understand different countries that have high happiness levels, regardless of their GDP value.

Scan I

Hide

```
# checking the null values in the data
colSums(is.na(data))
```

```
                     Country                 Country Code                         Year 
                           0                            0                            0 
                         GDP                Happiness Rank              Happiness Score 
                        3045                        17024                        16619 
     Economy (GDP per Capita)                       Family      Health (Life Expectancy) 
                       16619                        16619                        16619 
                     Freedom Trust (Government Corruption)                   Generosity 
                       16619                        16619                        16619 
              Happiness/GDP 
                       16621 
```

Hide

```
# we can see that there are similar counts of missing values for the data
# this can be due to the fact that we have used 2015-2017 data for happiness score
# hence for all other years, the data has null values
# we can remove the other rows with the year not equal to 2015-2017
data_filtered <- data %>% filter(Year == c(2015, 2016, 2017))
new_data <- data_filtered %>% filter(!is.na(data_filtered$`Happiness Score`))
colSums(is.na(new_data))
```

```
                     Country                 Country Code                         Year 
                           0                            0                            0 
                         GDP                Happiness Rank              Happiness Score 
                           0                          115                            0 
     Economy (GDP per Capita)                       Family      Health (Life Expectancy) 
                           0                            0                            0 
                     Freedom Trust (Government Corruption)                   Generosity 
                           0                            0                            0 
              Happiness/GDP 
                           0 
```

In our data, we have the GDP value from the year 1960 to 2023. Since we had merged the data with the Happiness Data that had the data of the years 2015-2017, during left_join, the Happiness Scores for the other years was marked as NA (null value).

For effective data analysis, we can choose to remove the other years as we would not need them since we do not have the Happiness Score for the countries.

We can do this by using the filter function:

filter(data, col=condition)

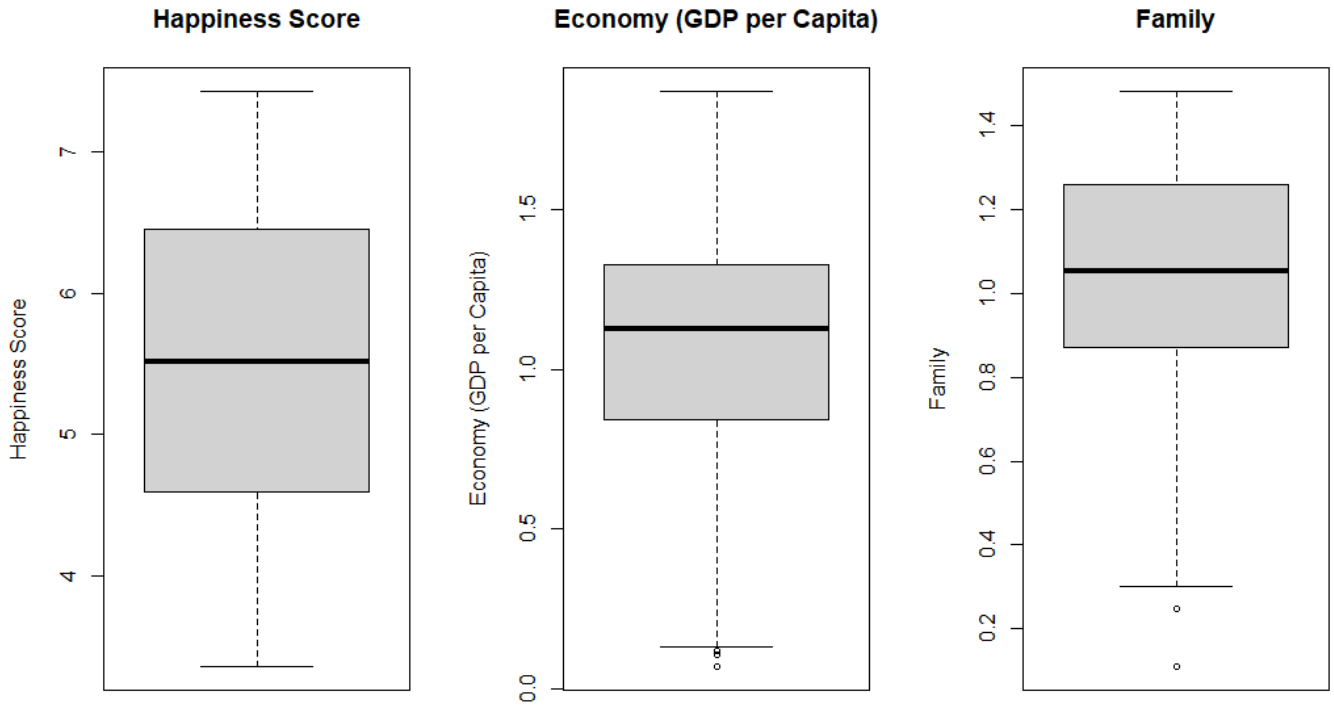
Scan II

Hide

```
# search for outliers in the data
# we have multiple numeric columns like GDP, Happiness Score, Family, Health, etc.
# To find the univariate outliers, we can use boxplots

par(mfrow = c(1, 3))  # Set the number of rows and columns for the plotting window

boxplot(new_data$`Happiness Score`, main = "Happiness Score", width=0.6, ylab="Happiness Scor
e")
boxplot(new_data$`Economy (GDP per Capita)`, main = "Economy (GDP per Capita)", width=0.6, yl
ab="Economy (GDP per Capita)")
```

Hide

```
boxplot(new_data$Family, main = "Family", width=0.6, ylab="Family")

par(mfrow = c(1, 4))
```



Hide

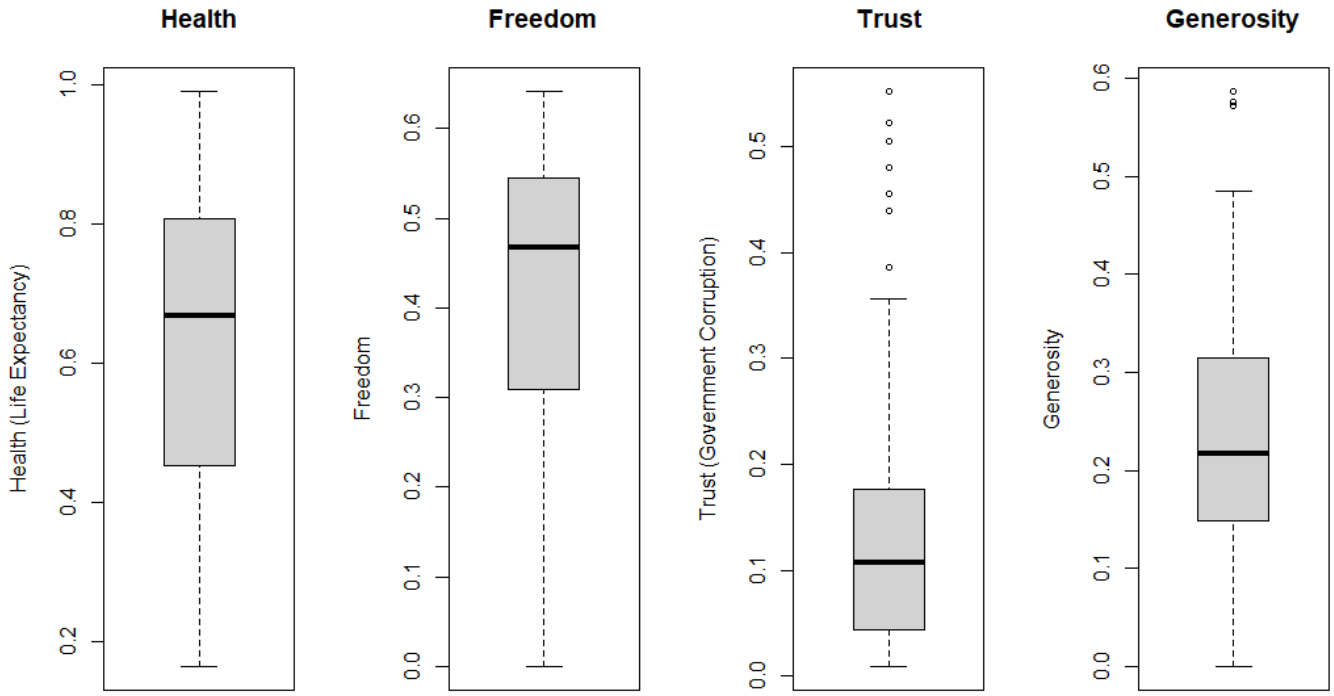
```
# boxplot of health (life expectancy)
new_data$`Health (Life Expectancy)` %>% boxplot(main='Health', ylab='Health (Life Expectanc
y)')

# boxplot of freedom
new_data$Freedom %>% boxplot(main='Freedom', ylab='Freedom')
```

Hide

```


# boxplot of Trust (Government Corruption)
new_data$`Trust (Government Corruption)` %>% boxplot(main='Trust', ylab='Trust (Government Co
rruption)')
# we can see many outliers in this variable

# boxplot of Generosity
new_data$Generosity %>% boxplot(main='Generosity', ylab='Generosity')
```

Hide

```
# we can see two groups of outliers in this variable

par(mfrow = c(1, 3))
```

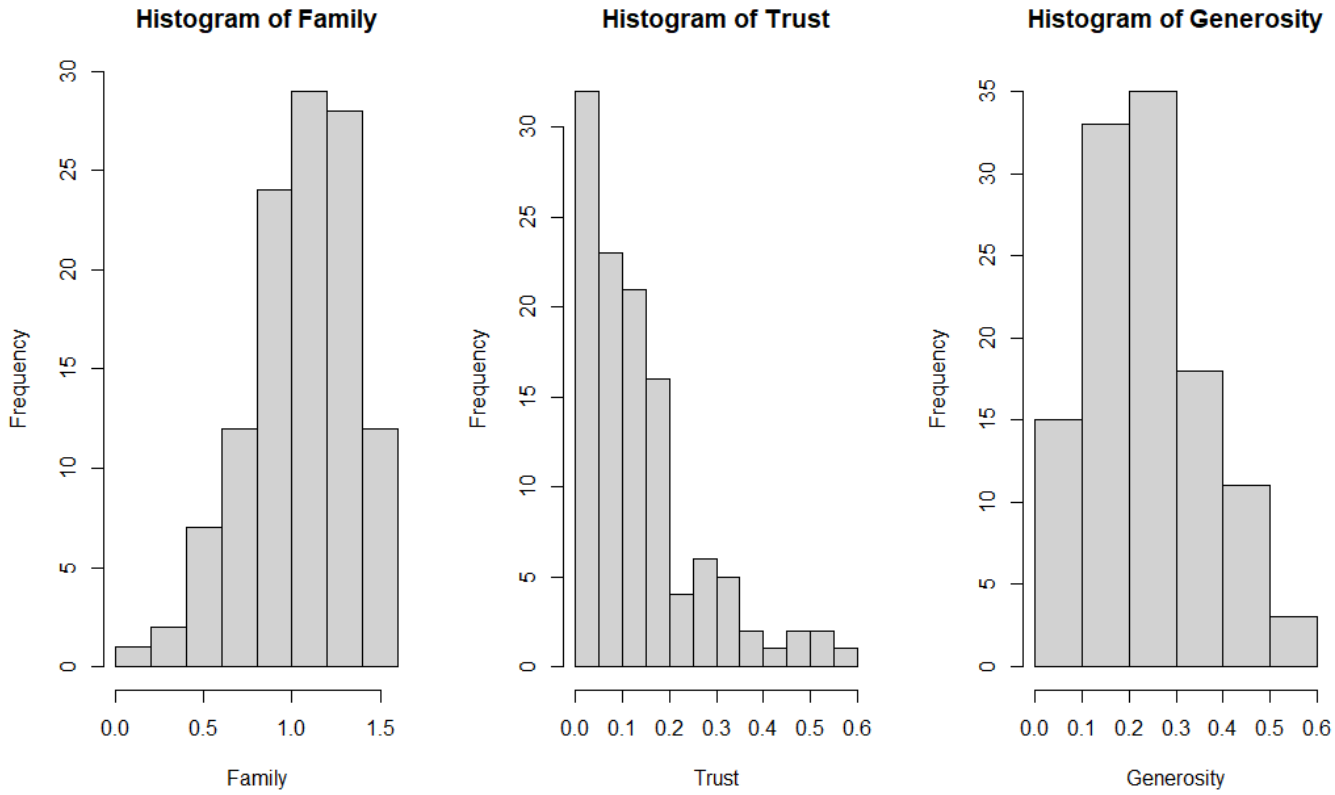


Hide

```
# we have outliers for the Family, Trust and Generosity variables
# checking their histogram plot to see if they are normally distributed
hist(new_data$Family, main = "Histogram of Family", xlab="Family")

# histogram of Trust
hist(new_data$`Trust (Government Corruption)`, main = "Histogram of Trust", xlab="Trust")
```

Hide

```


# histogram  of Generosity
hist(new_data$Generosity, main = "Histogram of Generosity", xlab="Generosity")

# since none of the variables are normally distributed, we cannot use the z-score test to che
ck for outliers

# checking bivariate outliers
par(mfrow = c(1, 1))
```



Hide

```
# happiness score vs family
plot(new_data$`Happiness Score`, new_data$Family, xlab="Happiness Score",
     ylab="Family", main="Happiness Score by Family")
# here we can see a few points to the bottom left of the graph that may be outliers
# using the Mahalanobis distance with QQ plots
par(mfrow = c(1, 3))
```

Hide

```
subset <- new_data %>% select(`Happiness Score`, `Family`)
results <- mvn(data=subset, multivariateOutlierMethod = "quan", showOutliers = TRUE)
results$multivariateOutliers
```

	Observation <chr>	Mahalanobis Distance <dbl>	Outlier <chr>
1	1	17.843	TRUE
2	2	13.830	TRUE
3	3	11.709	TRUE
4	4	8.303	TRUE

4 rows

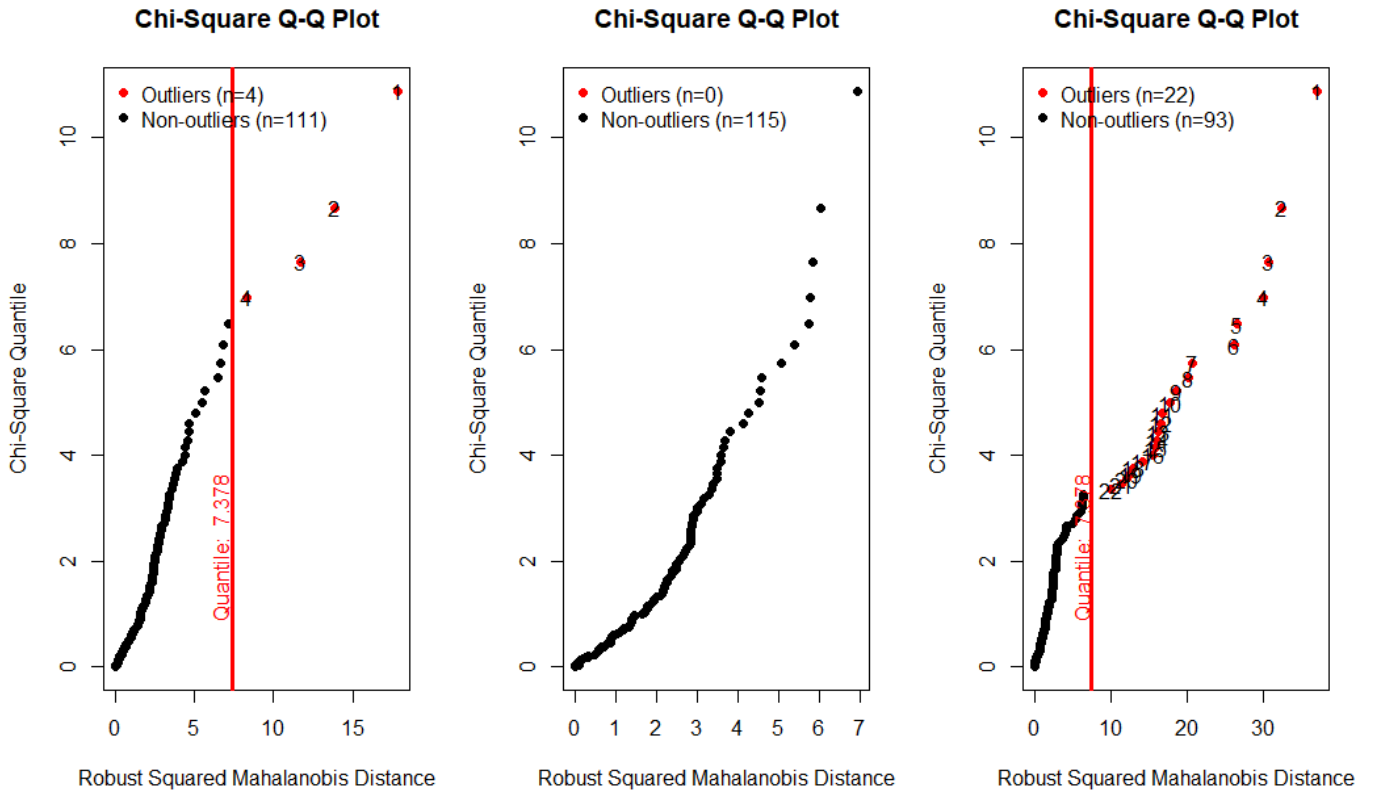
Hide

```
# we can see that there are 14 outliers in the data

# similarly for Happiness Score and Health
subset <- new_data %>% select(`Happiness Score`, `Health (Life Expectancy)`)
results <- mvn(data=subset, multivariateOutlierMethod = "quan", showOutliers = TRUE)
results$multivariateOutliers
```

0 rows

Hide

```


# similarly for Happienss Score and Freedom
subset <- new_data %>% select(`Happiness Score`, `Freedom`)
results <- mvn(data=subset, multivariateOutlierMethod = "quan", showOutliers = TRUE)
```



Hide

```
results$multivariateOutliers
```

	Observation <chr>	Mahalanobis Distance <dbl>	Outlier <chr>
1	1	37.124	TRUE
2	2	32.432	TRUE
3	3	30.674	TRUE
4	4	30.061	TRUE
5	5	26.595	TRUE
6	6	26.251	TRUE
7	7	20.709	TRUE
8	8	20.207	TRUE
9	9	18.678	TRUE
10	10	17.872	TRUE

1-10 of 22 rows Previous **1** 2 3 Next

Show

In this section, the histogram plot was used to identify univariate outliers for multiple variables. One of the simplest methods for detecting univariate outliers is the use of boxplots. A boxplot is a graphical display for describing the distribution of the data using the median, the first (Q1) and third quartiles (Q3), and the inter-quartile range (IQR=Q3−Q1).

In the boxplot, the “Tukey’s method of outlier detection” is used to detect outliers. According to this method, outliers are defined as the values in the data set that fall beyond the range of −1.5×IQR to 1.5×IQR. These −1.5×IQR and 1.5×IQR limits are called “outlier fences” and any values lying outside the outlier fences are depicted using an “o” or a similar symbol on the boxplot.

The variables that had outliers were Family, Trust and Generosity. Since these variables were not normally distributed, we cannot use the z-score approach for identifying outliers.

Multivariate Outliers for the pair Happiness Score and Family, Happiness Score and Health, Happiness Score and Freedom were identified using the Mahalanobis Distance method.

The Mahalanobis distance is the most commonly used distance metric to detect outliers for the multivariate setting. The Mahalanobis distance is simply an extension of the univariate z-score, which also accounts for the correlation structure between all the variables. Mahalanobis distance follows a Chi-square distribution with n (number of variables) degrees of freedom, therefore any Mahalanobis distance greater than the critical chi-square value is treated as outliers. The MVN package was used to create the Chi-Square Q-Q plot and outliers were marked in red.

results$multivariateOutliers is very useful in terms of providing the locations of outliers in the data set. This can be used to further remove the outliers from the data set.

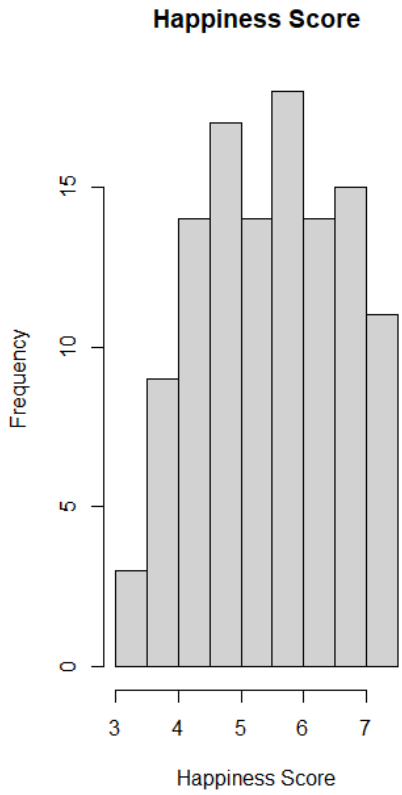
Transform

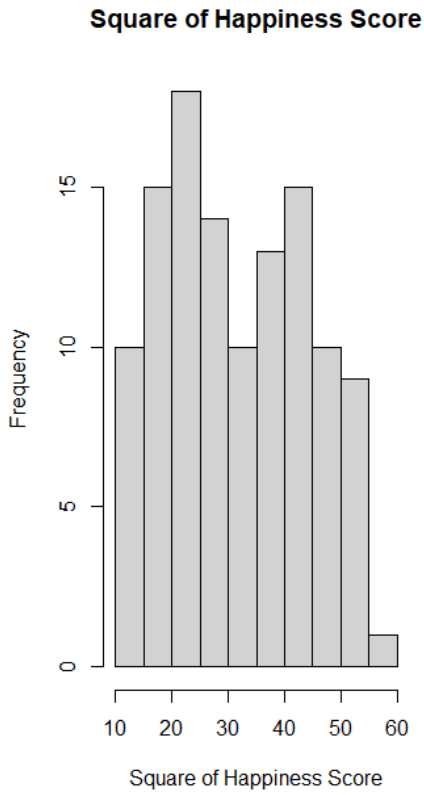
Hide

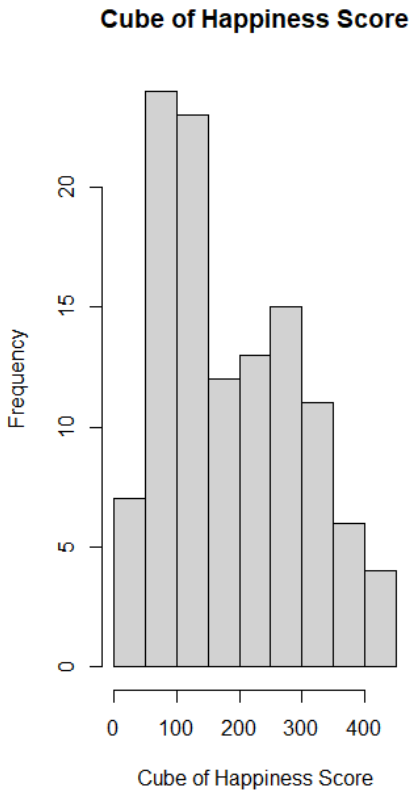
```
par(mfrow = c(1, 3))
# checking the distribution of the Happiness Score data
hist(new_data$`Happiness Score`, main="Happiness Score", xlab="Happiness Score")

# we can see that the data is slightly left skewed
# to convert this to a normal distribution, we can apply the square transformation
happiness_square <- new_data$`Happiness Score` ^ 2
hist(happiness_square, main="Square of Happiness Score", xlab="Square of Happiness Score")
```

Hide

```


happiness_cube <- new_data$`Happiness Score` ^ 3
hist(happiness_cube, main="Cube of Happiness Score", xlab="Cube of Happiness Score")
```





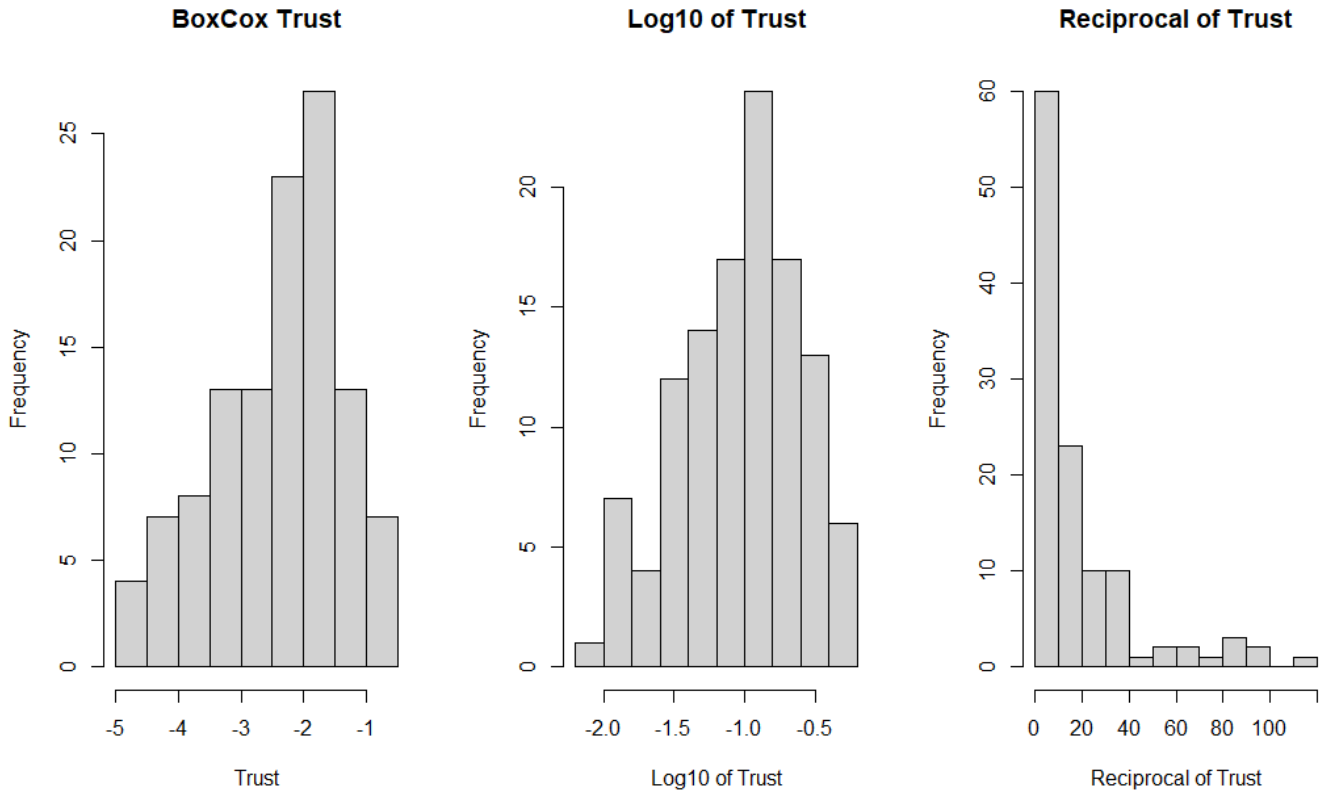


Hide

```
# if we apply a higher power, the data may become right skewed, so it is best to leave it at
cube

# applying boxcox transformation for Trust
trust_box_cox <- BoxCox(new_data$`Trust (Government Corruption)`, lambda="auto")
hist(trust_box_cox, main = "BoxCox Trust", xlab="Trust")
# the data became a little right skewed

# we can try applying log transformation
trust_log <- log10(new_data$`Trust (Government Corruption)`)
hist(trust_log, main="Log10 of Trust", xlab="Log10 of Trust")
```

Hide

```
# taking the reciprocal
trust_reci <- 1 / new_data$`Trust (Government Corruption)`
hist(trust_reci, main="Reciprocal of Trust", xlab="Reciprocal of Trust")
```



Hide

```
# this did not do anything
# we can conclude that log transformation worked best for the Trust variable

# performing PCA on the data
filtered_data <- new_data %>% select(-c('Country', 'Country Code', 'Year', 'Happiness Rank'))
pca1 <- preProcess(filtered_data, method="pca", thresh=0.90)
pca1
```

```
Created from 115 samples and 9 variables

Pre-processing:
  - centered (9)
  - ignored (0)
  - principal component signal extraction (9)
  - scaled (9)

PCA needed 6 components to capture 90 percent of the variance
```

Hide

```
# inspecting the extracted components
pca1$rotation
```

```
                                PC1         PC2         PC3         PC4         PC5
PC6
GDP                          0.2181445  0.04665352 -0.71250863 -0.55204879  0.35585483  0.0
8170007
Happiness Score              0.4863175 -0.02208114 -0.05760249  0.13922820 -0.08537822 -0.1
4622405
Economy (GDP per Capita)     0.4486154  0.22665697  0.08566905  0.10444120  0.07350594 -0.2
3248169
Family                       0.3546084  0.17948057 -0.03115681  0.36956310  0.12413984  0.7
6604955
Health (Life Expectancy)     0.4273491  0.21962259  0.04840270  0.09906532  0.04530639 -0.4
9623533
Freedom                      0.3298556 -0.46436562  0.15598680  0.02335103  0.02531816  0.1
7196205
Trust (Government Corruption)  0.1911712 -0.43177554  0.50968649 -0.43362738  0.35547302  0.0
3432880
Generosity                   0.1798741 -0.52946653 -0.33938446  0.02597595 -0.63603417 -0.0
2430368
Happiness/GDP               -0.1690333 -0.42791913 -0.28118791  0.57385201  0.55827401 -0.2
3124533
```

In the Transformation section of the report, the variables Happiness Score and Trust were taken for transformations. The Happiness Score variable shows a left skewness. For left skewed data, we can use power transformations to increase the impact of smaller data values. We use the square and cube transformations but even while doing so, the data did not get into a proper normal distribution.

For the Trust variable, the data is right skewed. We use the low level transformations like BoxCox, reciprocal and log10 to compress the higher values. The Trust variable showed somewhat normal distribution with the log10 transformation.

As an extra exercise, I also applied Principal Component Analysis on a subset of the data. This method is an unsupervised algorithm that creates linear combinations of the original features. The new extracted features are orthogonal, which means that they are uncorrelated. The extracted components are ranked in order of their “explained variance”. For example, the first principal component (PC1) explains the most variance in the data, PC2 explains the second-most variance, and so on. Then you can decide to keep only as many principal components as needed to reach a cumulative explained variance of 90%. Note that the advantage of this technique is fast and simple to implement and works well in practice. However, the new principal components are not interpretable, because they are linear combinations of original features.

PCA is used to analyse each variable’s (i.e. component’s) relation to the variance. It gives the variance for each component, and then allows the user to decide as to which components should be included in the analysis to prevent overfitting the model. The first component outputted has the highest variance, the second is lower and so on.

In our report, we use the caret package to implement PCA. The preProcess function of the caret package allows us to pass the data and the required threshold that is the cumulative explained variance that we need to achieve. From the summary, we see that PCA extracted 6 components. These components are uncorrelated (orthogonal to each other) and they have no specific interpretation. However, the advantage is extracting and using 6 new components instead of 9 would definitely help us with the “curse of dimensionality” problem.

Presentation

Presentation of the report:

Presentation (https://rmit-arc.instructuremedia.com/embed/caa7a732-a506-4ef1-a266-46942b0380d3)