Data Wrangling (Data Preprocessing)

Code

Practical Assessment 2

Patrick Adrianus

17 May 2023

Setup

Hide

```
# Load the necessary packages required to reproduce the report.

library(kableExtra)
library(magrittr)
library(readr)
library(tidyr)
library(dplyr)
library(zoo)
library(MVN)
```

Student names, numbers and percentage of contributions

Group information

Student name	Student number	Percentage of contribution
Patrick Adrianus	s3967271	60%
Anjali Lata Rambarath	s3943075	40%

Executive Summary

- In theory, both datasets are relatively untidy in similar aspects, and the importance of having a tidy dataset is so that the data extraction is simple.
- We read both datasets and manipulated them separately to prepare for merging them. We are to change the class types using the as.[class type] function to ensure there aren't abnormalities and count the missing values in both datasets to create a usable and appropriate dataset.
- Initially, tidying the datasets, we remove unnecessary values and rows using the pivot_longer function. We do this as the current dataset shows more than one observation in a row, making it hard to merge the dataset and perform other analyses on them. When introducing the pivot_longer function, we present two variable names, GDP and Population Growth, based on our analysis. In this section, we rename and set the class type for some variables, keeping only what we need.
- Only after this can we join the two datasets based on the country 'Code' and 'Year', removing the duplicate variable names.
- The variable we added is GDP growth (%), where we get the percentage of the GDP for each row.
- We scanned for missing data, and to combat the significant amount of missing data, a predicted model was used so that valuable data didn't get removed. After filling the dataset appropriately, we scan for special values and check Nan values. In the end, we check for duplicated rows and address them properly.

- We then visually represent the GDP and Population growth and check for any outliers. We make appropriate adjustments for the mvn function to work correctly and handle the outliers using the capping method.
- We transform the dataset by presenting the centred and scaled versions and attempting to transform the dataset to minimise the skewness and create a normal distribution.

Data

Provide explanations here. The datasets were a GDP dataset and a population growth dataset over time. The population growth dataset shows us how much the population increases each year as a percentage annually. This dataset was found on the world bank, https://data.worldbank.org/indicator/SP.POP.GROW (https://data.worldbank.org/indicator/SP.POP.GROW) The variables in this dataset include:

- Country Name: The name of the country
- Country Code: A three-letter code that represents the country and can be used as a unique identifier
- Indicator Name: Represents the population growth annually as a percentage. This variable can be dropped as it doesn't give us valuable data
- Indicator code: A code to represent the indicator name and can be used as a unique identifier. This variable can be dropped as it doesn't give us helpful data
- from 1960 - 2022: It presents the population growth annually as a percentage for each country and each year.

Similarly, the GDP dataset shows us the increase each year annually by $ (dollars). The dataset was found on Kaggle from https://www.kaggle.com/datasets/zgrcemta/world-gdpgdp-gdp-per-capita-and-annual-growths (https://www.kaggle.com/datasets/zgrcemta/world-gdpgdp-gdp-per-capita-and-annual-growths). While Kaggle hosts an array of data files, we have chosen to work with gdp.csv to obtain unprocessed data. The variables in this dataset include:

- Country Name: The name of the country
- Code: A three-letter code that represents the country and can be used as a unique identifier
- from 1960 - 2020: Presents the GDP of each country for each year

Hide

```
# Import the data, provide your R codes here.

#reading gdp dataset
gdp_data <- read_csv("gdp.csv")
```

```
Rows: 266 Columns: 64── Column specification ────────────────────────────────────────────────────────
────────────────────────

Delimiter: ","
chr  (2): Country Name, Code
dbl (61): 1960, 1961, 1962, 1963, 1964, 1965, 1966, 1967, 1968, 1969, 1970, 1971, 1972, 1973, 1974, 1975,
1976, 1977, 1978, 1979, 1980, 1981, 1982, 1983, 1984, 1985, 1986, 1987, 1988, 1989, 1990, 1991, 1992, 19
9...
lgl  (1): Unnamed: 65
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

```
#Reading population growth dataset
population_growth <- read_csv("API_SP.POP.GROW_DS2_en_csv_v2_5455041.csv", skip = 4)
```

```
New names:Rows: 266 Columns: 68── Column specification ────────────────────────────────────────────────────────
Delimiter: ","
chr  (4): Country Name, Country Code, Indicator Name, Indicator Code
dbl (61): 1961, 1962, 1963, 1964, 1965, 1966, 1967, 1968, 1969, 1970, 1971, 1972, 1973, 1974, 1975, 1976,
1977, 1978, 1979, 1980, 1981, 1982, 1983, 1984, 1985, 1986, 1987, 1988, 1989, 1990, 1991, 1992, 1993, 19
9...
lgl  (3): 1960, 2022, ...68
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.
```

Understand

Hide

```
# This is the R chunk for the Understand Section

# Understand the gdp dataset
head(gdp_data)
```

Country Name <chr>	C… <chr>	1960 <dbl>	1961 <dbl>	1962 <dbl>	1963 <dbl>	1964 <dbl>
Aruba	ABW	*NA*	*NA*	*NA*	*NA*	*NA*
Africa Eastern and Southern	AFE	19313106302	19723488057	21493920015	25733212134	23527443251
Afghanistan	AFG	537777811	548888896	546666678	751111191	800000044
Africa Western and Central	AFW	10404280784	11128050589	11943353288	12676515454	13838577015
Angola	AGO	*NA*	*NA*	*NA*	*NA*	*NA*
Albania	ALB	*NA*	*NA*	*NA*	*NA*	*NA*

6 rows | 1-7 of 64 columns

Hide

```
glimpse(gdp_data)
```

```
Rows: 266
Columns: 64
$ `Country Name` <chr> "Aruba", "Africa Eastern and Southern", "Afghanistan", "Africa Western and Centra
l", "Angola", "Albania", "Andorra", "Arab World", "United Arab Emirates", "Argentina", "Armenia", "America
n S…
$ Code           <chr> "ABW", "AFE", "AFG", "AFW", "AGO", "ALB", "AND", "ARB", "ARE", "ARG", "ARM", "ASM",
"ATG", "AUS", "AUT", "AZE", "BDI", "BEL", "BEN", "BFA", "BGD", "BGR", "BHR", "BHS", "BIH", "BLR", "BLZ",
"…
$ `1960`         <dbl> NA, 19313106302, 537777811, 10404280784, NA, NA, NA, NA, NA, NA, NA, NA, NA, 186067
86874, 6592693841, NA, 195999990, 11658722591, 226195579, 330442817, 4274893913, NA, NA, 169803922, NA, N
A,…
$ `1961`         <dbl> NA, 19723488057, 548888896, 11128050589, NA, NA, NA, NA, NA, NA, NA, NA, NA, 196830
55213, 7311749633, NA, 202999992, 12400145222, 235668222, 350247237, 4817580184, NA, NA, 190098039, NA, N
A,…
$ `1962`         <dbl> NA, 21493920015, 546666678, 11943353288, NA, NA, NA, NA, NA, 24450604878, NA, NA, N
A, 19922723709, 7756110210, NA, 213500006, 13264015675, 236434907, 379567023, 5081413340, NA, NA, 21225490
2…
$ `1963`         <dbl> NA, 25733212134, 751111191, 12676515454, NA, NA, NA, NA, NA, 18272123664, NA, NA, N
A, 21539926084, 8374175258, NA, 232749998, 14260017387, 253927646, 394040588, 5319458351, NA, NA, 23774509
8…
$ `1964`         <dbl> NA, 23527443251, 800000044, 13838577015, NA, NA, NA, NA, NA, 25605249382, NA, NA, N
A, 23801097547, 9169983886, NA, 260750008, 15960106681, 269818988, 410321618, 5386054619, NA, NA, 26666666
7…
$ `1965`         <dbl> NA, 26810567154, 1006666638, 14862472886, NA, NA, NA, NA, NA, 28344705967, NA, NA,
NA, 25977153097, 9994070616, NA, 158994963, 17371457608, 289908721, 422916848, 5906636557, NA, NA, 3003921
5…
$ `1966`         <dbl> NA, 29152157362, 1399999967, 15832846881, NA, NA, NA, NA, NA, 28630474728, NA, NA,
NA, 27309889125, 10887682273, NA, 165444571, 18651883472, 302925281, 433889832, 6439687598, NA, NA, 340000
0…
$ `1967`         <dbl> NA, 30173172663, 1673333418, 14426432397, NA, NA, NA, NA, NA, 24256667553, NA, NA,
NA, 30444618658, 11579431669, NA, 178297143, 19992040788, 306222000, 450753993, 7253575399, NA, NA, 390196
0…
$ `1968`         <dbl> NA, 32877055829, 1373333367, 14880350847, NA, NA, NA, 35087069740, NA, 26436857247,
NA, NA, NA, 32716989584, 12440625313, NA, 183200000, 21376353113, 326323097, 460442678, 7483685474, NA, N
A…
$ `1969`         <dbl> NA, 37744346869, 1408888922, 16882094303, NA, NA, NA, 38236378393, NA, 31256284544,
NA, NA, NA, 36686079068, 13582798556, NA, 190205714, 23710735895, 330748211, 478298598, 8471006101, NA, N
A…
$ `1970`         <dbl> NA, 4.031578e+10, 1.748887e+09, 2.350461e+10, NA, NA, 7.861921e+07, 4.323280e+10, N
A, 3.158421e+10, NA, NA, NA, 4.133722e+10, 1.537301e+10, NA, 2.427326e+08, 2.670620e+10, 3.336278e+08, 4.5
8…
$ `1971`         <dbl> NA, 4.447666e+10, 1.831109e+09, 2.083282e+10, NA, NA, 8.940982e+07, 4.993733e+10, N
A, 3.329320e+10, NA, NA, NA, 4.522231e+10, 1.785849e+10, NA, 2.528423e+08, 2.982166e+10, 3.350730e+08, 4.8
2…
$ `1972`         <dbl> NA, 4.830149e+10, 1.595555e+09, 2.526496e+10, NA, NA, 1.134082e+08, 5.949548e+10, N
A, 3.473300e+10, NA, NA, NA, 5.205140e+10, 2.205961e+10, NA, 2.468046e+08, 3.720942e+10, 4.103319e+08, 5.7
8…
$ `1973`         <dbl> NA, 6.298350e+10, 1.733333e+09, 3.127382e+10, NA, NA, 1.508201e+08, 7.547465e+10, N
A, 5.254400e+10, NA, NA, NA, 6.384497e+10, 2.951547e+10, NA, 3.043398e+08, 4.774380e+10, 5.043760e+08, 6.7
4…
$ `1974`         <dbl> NA, 7.825089e+10, 2.155555e+09, 4.421449e+10, NA, NA, 1.865587e+08, 1.430903e+11, N
A, 7.243678e+10, NA, NA, NA, 8.898158e+10, 3.518930e+10, NA, 3.452635e+08, 5.603308e+10, 5.546548e+08, 7.5
1…
$ `1975`         <dbl> NA, 8.343557e+10, 2.366667e+09, 5.144474e+10, NA, NA, 2.201272e+08, 1.582128e+11,
1.472067e+10, 5.243865e+10, NA, NA, NA, 9.733306e+10, 4.005921e+10, NA, 4.209867e+08, 6.567819e+10, 6.7687
01…
$ `1976`         <dbl> NA, 8.321043e+10, 2.555556e+09, 6.212940e+10, NA, NA, 2.272810e+08, 1.969369e+11,
1.921302e+10, 5.116950e+10, NA, NA, NA, 1.051015e+11, 4.295998e+10, NA, 4.484128e+08, 7.111388e+10, 6.9840
82…
```

```
$ `1977`         <dbl> NA, 9.498895e+10, 2.953333e+09, 6.531501e+10, NA, NA, 2.540202e+08, 2.274142e+11,
2.487178e+10, 5.678100e+10, NA, NA, 7.749675e+07, 1.103877e+11, 5.154576e+10, NA, 5.475356e+08, 8.283991e+
10…
$ `1978`         <dbl> NA, 1.063461e+11, 3.300000e+09, 7.119972e+10, NA, NA, 3.080089e+08, 2.495894e+11,
2.377583e+10, 5.808287e+10, NA, NA, 8.787934e+07, 1.185360e+11, 6.205226e+10, NA, 6.102256e+08, 1.012465e+
11…
$ `1979`         <dbl> NA, 1.244982e+11, 3.697940e+09, 8.862841e+10, NA, NA, 4.115783e+08, 3.392759e+11,
3.122546e+10, 6.925233e+10, NA, NA, 1.090800e+08, 1.349415e+11, 7.393730e+10, NA, 7.824967e+08, 1.163155e+
11…
$ `1980`         <dbl> NA, 1.565127e+11, 3.641723e+09, 1.120313e+11, 5.934074e+09, NA, 4.464161e+08, 4.590
771e+11, 4.359875e+10, 7.696192e+10, NA, NA, 1.314310e+08, 1.500323e+11, 8.205891e+10, NA, 9.197267e+08,
1.…
$ `1981`         <dbl> NA, 1.603781e+11, 3.478788e+09, 2.110035e+11, 5.553824e+09, NA, 3.889587e+08, 4.749
222e+11, 4.933342e+10, 7.867684e+10, NA, NA, 1.478417e+08, 1.769534e+11, 7.103423e+10, NA, 9.690467e+08,
1.…
$ `1982`         <dbl> NA, 1.546694e+11, NA, 1.871637e+11, 5.553824e+09, NA, 3.758960e+08, 4.459653e+11,
4.662272e+10, 8.430749e+10, NA, NA, 1.643693e+08, 1.941046e+11, 7.127529e+10, NA, 1.013222e+09, 9.209593e+
10…
$ `1983`         <dbl> NA, 1.597575e+11, NA, 1.381152e+11, 5.787824e+09, NA, 3.278618e+08, 4.192552e+11,
4.280332e+10, 1.039791e+11, NA, NA, 1.821441e+08, 1.773336e+11, 7.212102e+10, NA, 1.082926e+09, 8.718424e+
10…
$ `1984`         <dbl> NA, 1.460219e+11, NA, 1.142627e+11, 6.135166e+09, 1.857338e+09, 3.300707e+08, 4.259
035e+11, 4.180795e+10, 7.909200e+10, NA, NA, 2.083728e+08, 1.935936e+11, 6.798534e+10, NA, 9.871439e+08,
8.…
$ `1985`         <dbl> NA, 1.304398e+11, NA, 1.165073e+11, 7.558613e+09, 1.897050e+09, 3.467380e+08, 4.249
330e+11, 4.060365e+10, 8.841667e+10, NA, NA, 2.409239e+08, 1.805738e+11, 6.938677e+10, NA, 1.149979e+09,
8.…
$ `1986`         <dbl> 4.054634e+08, 1.470252e+11, NA, 1.074975e+11, 7.076794e+09, 2.097326e+09, 4.820006e
+08, 4.138589e+11, 3.394361e+10, 1.109344e+11, NA, NA, 2.904401e+08, 1.823685e+11, 9.903616e+10, NA, 1.201
7…
$ `1987`         <dbl> 4.876025e+08, 1.797395e+11, NA, 1.103218e+11, 8.089279e+09, 2.080796e+09, 6.113164e
+08, 4.501093e+11, 3.638491e+10, 1.111062e+11, NA, NA, 3.371749e+08, 1.894005e+11, 1.241684e+11, NA, 1.131
4…
$ `1988`         <dbl> 5.964236e+08, 1.890033e+11, NA, 1.089435e+11, 8.775116e+09, 2.051236e+09, 7.214259e
+08, 4.268160e+11, 3.627567e+10, 1.262068e+11, NA, NA, 3.986377e+08, 2.360659e+11, 1.333394e+11, NA, 1.082
4…
$ `1989`         <dbl> 6.953044e+08, 1.945434e+11, NA, 1.017688e+11, 1.020792e+10, 2.253090e+09, 7.954493e
+08, 4.501890e+11, 4.146500e+10, 7.663690e+10, NA, NA, 4.387948e+08, 2.997679e+11, 1.331058e+11, NA, 1.113
9…
$ `1990`         <dbl> 7.648871e+08, 2.123361e+11, NA, 1.218022e+11, 1.123628e+10, 2.028554e+09, 1.029048e
+09, 6.226947e+11, 5.070144e+10, 1.413524e+11, 2.256839e+09, NA, 4.594704e+08, 3.113267e+11, 1.664634e+11,
…
$ `1991`         <dbl> 8.721387e+08, 2.207637e+11, NA, 1.174570e+11, NA, 1.099559e+09, 1.106929e+09, 4.387
866e+11, 5.155217e+10, 1.897200e+11, 2.069870e+09, NA, 4.817074e+08, 3.259030e+11, 1.737942e+11, NA, 1.167
3…
$ `1992`         <dbl> 9.584632e+08, 2.202188e+11, NA, 1.182823e+11, NA, 6.521750e+08, 1.210014e+09, 4.738
248e+11, 5.423917e+10, 2.287886e+11, 1.272835e+09, NA, 4.992815e+08, 3.254803e+11, 1.950781e+11, 4.463056e
+…
$ `1993`         <dbl> 1.082980e+09, 2.340352e+11, NA, 9.882641e+10, NA, 1.185315e+09, 1.007026e+09, 4.827
783e+11, 5.562517e+10, 2.367417e+11, 1.201313e+09, NA, 5.351741e+08, 3.121262e+11, 1.903797e+11, 1.570000e
+…
$ `1994`         <dbl> 1.245688e+09, 2.390883e+11, NA, 8.628177e+10, 3.390500e+09, 1.880952e+09, 1.017549e
+09, 5.077805e+11, 5.930509e+10, 2.574400e+11, 1.315159e+09, NA, 5.894296e+08, 3.228073e+11, 2.035352e+11,
…
$ `1995`         <dbl> 1.320475e+09, 2.696893e+11, NA, 1.082213e+11, 5.561222e+09, 2.392765e+09, 1.178739e
+09, 5.560714e+11, 6.574367e+10, 2.580318e+11, 1.468317e+09, NA, 5.772815e+08, 3.679158e+11, 2.410383e+11,
…
$ `1996`         <dbl> 1.379961e+09, 2.684404e+11, NA, 1.257630e+11, 7.526964e+09, 3.199641e+09, 1.223945e
+09, 6.142084e+11, 7.357123e+10, 2.721498e+11, 1.596969e+09, NA, 6.337296e+08, 4.010895e+11, 2.372509e+11,
```

…
$ `1997`   <dbl> 1.531944e+09, 2.822135e+11, NA, 1.270639e+11, 7.649716e+09, 2.258514e+09, 1.180597e
+09, 6.623337e+11, 7.883901e+10, 2.928590e+11, 1.639492e+09, NA, 6.806185e+08, 4.353240e+11, 2.127903e+11,
…
$ `1998`   <dbl> 1.665101e+09, 2.658377e+11, NA, 1.301068e+11, 6.506619e+09, 2.545965e+09, 1.211932e
+09, 6.445101e+11, 7.567434e+10, 2.989483e+11, 1.893726e+09, NA, 7.278593e+08, 3.994045e+11, 2.182599e+11,
…
$ `1999`   <dbl> 1.722799e+09, 2.621972e+11, NA, 1.375202e+11, 6.152937e+09, 3.212122e+09, 1.239876e
+09, 7.128534e+11, 8.444547e+10, 2.835230e+11, 1.845482e+09, NA, 7.662000e+08, 3.890989e+11, 2.171858e+11,
…
$ `2000`   <dbl> 1.873453e+09, 2.839525e+11, NA, 1.404080e+11, 9.129635e+09, 3.480355e+09, 1.429049e
+09, 8.160388e+11, 1.043374e+11, 2.842038e+11, 1.911564e+09, NA, 8.263704e+08, 4.155762e+11, 1.967998e+11,
…
$ `2001`   <dbl> 1.920112e+09, 2.588432e+11, NA, 1.480120e+11, 8.936064e+09, 3.922101e+09, 1.546926e
+09, 7.985250e+11, 1.033116e+11, 2.686968e+11, 2.118468e+09, NA, 8.004815e+08, 3.790839e+11, 1.973379e+11,
…
$ `2002`   <dbl> 1.941341e+09, 2.648950e+11, 4.055180e+09, 1.769334e+11, 1.528559e+10, 4.348068e+09,
1.755910e+09, 8.031423e+11, 1.098162e+11, 9.772400e+10, 2.376335e+09, 5.120000e+08, 8.143815e+08, 3.95342
7…
$ `2003`   <dbl> 2.021229e+09, 3.526921e+11, 4.515559e+09, 2.046419e+11, 1.781271e+10, 5.611496e+09,
2.361727e+09, 8.878742e+11, 1.243464e+11, 1.275870e+11, 2.807061e+09, 5.240000e+08, 8.563963e+08, 4.67390
8…
$ `2004`   <dbl> 2.228492e+09, 4.388756e+11, 5.226779e+09, 2.540903e+11, 2.355205e+10, 7.184686e+09,
2.894922e+09, 1.056272e+12, 1.478244e+11, 1.646579e+11, 3.576615e+09, 5.090000e+08, 9.197296e+08, 6.14166
3…
$ `2005`   <dbl> 2.330726e+09, 5.122599e+11, 6.209138e+09, 3.105543e+11, 3.697092e+10, 8.052074e+09,
3.159905e+09, 1.297924e+12, 1.806175e+11, 1.987371e+11, 4.900470e+09, 5.000000e+08, 1.022963e+09, 6.95075
2…
$ `2006`   <dbl> 2.424581e+09, 5.759762e+11, 6.971286e+09, 3.932967e+11, 5.238101e+10, 8.896073e+09,
3.456442e+09, 1.533188e+12, 2.221165e+11, 2.325573e+11, 6.384452e+09, 4.930000e+08, 1.157663e+09, 7.47556
2…
$ `2007`   <dbl> 2.615084e+09, 6.612422e+11, 9.747880e+09, 4.617776e+11, 6.526645e+10, 1.067732e+10,
3.952601e+09, 1.789560e+12, 2.579161e+11, 2.875305e+11, 9.206302e+09, 5.180000e+08, 1.312759e+09, 8.53955
4…
$ `2008`   <dbl> 2.745251e+09, 7.083543e+11, 1.010931e+10, 5.664257e+11, 8.853861e+10, 1.288135e+10,
4.085631e+09, 2.254219e+12, 3.154746e+11, 3.615580e+11, 1.166204e+10, 5.600000e+08, 1.370070e+09, 1.05512
7…
$ `2009`   <dbl> 2.498883e+09, 7.125580e+11, 1.241616e+10, 5.069964e+11, 7.030716e+10, 1.204421e+10,
3.674410e+09, 1.974361e+12, 2.535474e+11, 3.329765e+11, 8.647937e+09, 6.750000e+08, 1.228330e+09, 9.28043
0…
$ `2010`   <dbl> 2.390503e+09, 8.474095e+11, 1.585668e+10, 5.915830e+11, 8.379950e+10, 1.192692e+10,
3.449967e+09, 2.311237e+12, 2.897873e+11, 4.236274e+11, 9.260285e+09, 5.730000e+08, 1.148700e+09, 1.14758
9…
$ `2011`   <dbl> 2.549721e+09, 9.433782e+11, 1.780511e+10, 6.709626e+11, 1.117897e+11, 1.289076e+10,
3.629204e+09, 2.501780e+12, 3.506660e+11, 5.301633e+11, 1.014211e+10, 5.700000e+08, 1.137637e+09, 1.39790
8…
$ `2012`   <dbl> 2.534637e+09, 9.505214e+11, 1.990732e+10, 7.275714e+11, 1.280529e+11, 1.231983e+10,
3.188809e+09, 2.740553e+12, 3.745906e+11, 5.459824e+11, 1.061932e+10, 6.400000e+08, 1.199948e+09, 1.54650
9…
$ `2013`   <dbl> 2.727850e+09, 9.642424e+11, 2.014640e+10, 8.207876e+11, 1.367099e+11, 1.277622e+10,
3.193704e+09, 2.799579e+12, 3.901076e+11, 5.520251e+11, 1.112147e+10, 6.380000e+08, 1.181448e+09, 1.57633
5…
$ `2014`   <dbl> 2.790849e+09, 9.848071e+11, 2.049713e+10, 8.649666e+11, 1.457122e+11, 1.322815e+10,
3.271808e+09, 2.831681e+12, 4.031371e+11, 5.263197e+11, 1.160951e+10, 6.430000e+08, 1.249733e+09, 1.46750
5…
$ `2015`   <dbl> 2.962905e+09, 9.199300e+11, 1.913421e+10, 7.607297e+11, 1.161936e+11, 1.138685e+10,
2.789870e+09, 2.463580e+12, 3.581351e+11, 5.947493e+11, 1.055334e+10, 6.730000e+08, 1.336693e+09, 1.35053
4…
$ `2016`   <dbl> 2.983637e+09, 8.733549e+11, 1.811656e+10, 6.905430e+11, 1.011239e+11, 1.186120e+10,

```
2.896679e+09, 2.411981e+12, 3.570451e+11, 5.575314e+11, 1.054614e+10, 6.710000e+08, 1.436585e+09, 1.20668
5…
$ `2017`         <dbl> 3.092430e+09, 9.853557e+11, 1.875347e+10, 6.837416e+11, 1.221238e+11, 1.301969e+10,
3.000181e+09, 2.466443e+12, 3.856055e+11, 6.436287e+11, 1.152746e+10, 6.120000e+08, 1.467978e+09, 1.32688
3…
$ `2018`         <dbl> 3.202189e+09, 1.012853e+12, 1.805323e+10, 7.416916e+11, 1.013532e+11, 1.515643e+10,
3.218316e+09, 2.730780e+12, 4.222150e+11, 5.248197e+11, 1.245794e+10, 6.390000e+08, 1.605944e+09, 1.42853
0…
$ `2019`         <dbl> NA, 1.009910e+12, 1.879945e+10, 7.945725e+11, 8.941719e+10, 1.540024e+10, 3.155065e
+09, 2.776469e+12, 4.172156e+11, 4.519324e+11, 1.361929e+10, 6.480000e+08, 1.687533e+09, 1.391953e+12, 4.4
5…
$ `2020`         <dbl> NA, 9.207923e+11, 2.011614e+10, 7.845876e+11, 5.837598e+10, 1.488763e+10, NA, 2.447
584e+12, 3.588688e+11, 3.892881e+11, 1.264121e+10, 7.090000e+08, 1.370281e+09, 1.327836e+12, 4.332585e+11,
…
$ `Unnamed: 65`  <lgl> 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, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N
A…
```

Hide

```
gdp_na_count <- sum(is.na(gdp_data))
gdp_na_count
```

```
[1] 3652
```

Hide

```
# Convert 'Country Name' and 'Code' in gdp_data to factor
gdp_data$`Country Name` <- as.factor(gdp_data$`Country Name`)
gdp_data$Code <- as.factor(gdp_data$Code)

# Understand the population growth dataset
head(population_growth)
```

Country Name <chr>	**Country Code** <chr>	**Indicator Name** <chr>	**Indicator Code** <chr>	**1…** <lgl>
Aruba	ABW	Population growth (annual %)	SP.POP.GROW	*NA*
Africa Eastern and Southern	AFE	Population growth (annual %)	SP.POP.GROW	*NA*
Afghanistan	AFG	Population growth (annual %)	SP.POP.GROW	*NA*
Africa Western and Central	AFW	Population growth (annual %)	SP.POP.GROW	*NA*
Angola	AGO	Population growth (annual %)	SP.POP.GROW	*NA*
Albania	ALB	Population growth (annual %)	SP.POP.GROW	*NA*

6 rows | 1-5 of 68 columns

Hide

```
glimpse(population_growth)
```

```
Rows: 266
Columns: 68
$ `Country Name`   <chr> "Aruba", "Africa Eastern and Southern", "Afghanistan", "Africa Western and Centra
l", "Angola", "Albania", "Andorra", "Arab World", "United Arab Emirates", "Argentina", "Armenia", "America
n…
$ `Country Code`   <chr> "ABW", "AFE", "AFG", "AFW", "AGO", "ALB", "AND", "ARB", "ARE", "ARG", "ARM", "AS
M", "ATG", "AUS", "AUT", "AZE", "BDI", "BEL", "BEN", "BFA", "BGD", "BGR", "BHR", "BHS", "BIH", "BLR", "BL
Z",…
$ `Indicator Name` <chr> "Population growth (annual %)", "Population growth (annual %)", "Population growt
h (annual %)", "Population growth (annual %)", "Population growth (annual %)", "Population growth (annual
%…
$ `Indicator Code` <chr> "SP.POP.GROW", "SP.POP.GROW", "SP.POP.GROW", "SP.POP.GROW", "SP.POP.GROW", "SP.PO
P.GROW", "SP.POP.GROW", "SP.POP.GROW", "SP.POP.GROW", "SP.POP.GROW", "SP.POP.GROW", "SP.POP.GROW", "SP.PO
P.…
$ `1960`           <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N
A, 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,
…
$ `1961`           <dbl> 2.1790590, 2.6601801, 1.9259516, 2.1157891, 1.5583550, 3.1208554, 7.8681392, 2.57
17183, 5.5099392, 1.6130294, 3.4775227, 2.6579148, 1.6185034, 1.9897400, 0.5484724, 3.8101646, 2.4933518,
0…
$ `1962`           <dbl> 1.5485717, 2.7326331, 2.0148789, 2.1457231, 1.4607384, 3.0567305, 7.5212072, 2.61
93879, 5.4474053, 1.6292819, 3.3819277, 3.0839238, 1.3474461, 2.4406394, 0.6128958, 2.9805345, 2.5048255,
0…
$ `1963`           <dbl> 1.3893371, 2.7532483, 2.0789966, 2.1908269, 1.4104253, 2.9537488, 7.2231976, 2.67
00510, 5.3163560, 1.6216614, 3.2228670, 3.1329926, 1.3416472, 1.9178165, 0.6423627, 2.9641604, 2.0816338,
0…
$ `1964`           <dbl> 1.2157206, 2.8069150, 2.1396508, 2.2113600, 1.3017452, 2.8806864, 6.9415115, 2.70
19459, 5.1508231, 1.5995232, 3.0343411, 3.1703137, 1.5218170, 1.9623544, 0.6665482, 3.3371547, 2.8453003,
0…
$ `1965`           <dbl> 1.0328409, 2.8407869, 2.2160069, 2.2425675, 1.1110406, 2.7540212, 6.6531215, 2.72
68617, 5.0095699, 1.5775330, 2.8615281, 3.1926615, 1.6567306, 1.9597169, 0.6497299, 3.0044750, 2.7609655,
0…
$ `1966`           <dbl> 0.8621838, 2.8136095, 2.2535240, 2.2728164, 0.8758063, 2.6345639, 6.9970545, 2.77
18144, 4.8911535, 1.5721436, 2.7132413, 3.0772954, 1.6148830, 2.2831844, 0.7013961, 2.8764856, 2.3844545,
0…
$ `1967`           <dbl> 0.3888468, 2.8305053, 2.2926379, 2.2757206, 0.6966981, 2.6301903, 7.9206813, 2.83
85586, 4.7586891, 1.5673349, 2.6015050, 2.9652979, 1.6395419, 1.2622769, 0.7474252, 2.5965411, 2.5083757,
0…
$ `1968`           <dbl> -0.08571933, 2.88158852, 2.34735064, 2.29595318, 0.69598474, 2.84251071, 8.132678
41, 2.87096869, 10.96344564, 1.55205695, 2.53400807, 3.01275363, 1.67535602, 1.76415866, 0.51925426, 2.382
4…
$ `1969`           <dbl> -0.2373719, 2.8985320, 2.3789165, 2.3239975, 1.0203546, 2.8960834, 7.7160530, 2.8
793594, 17.0399744, 1.5415439, 2.4485263, 3.0307681, 1.7152319, 2.0930230, 0.3453316, 2.2933944, 2.125913
0,…
$ `1970`           <dbl> -0.37826382, 2.91152614, 2.43318262, 2.36819833, 1.69452577, 2.55085118, 7.369749
01, 2.86717322, 16.29547485, 1.56729965, 2.31585112, 2.53983094, 1.20850690, 1.97018888, 0.34921895, 2.125
0…
$ `1971`           <dbl> -0.4918515, 2.9148559, 2.4153692, 2.3975679, 2.4143391, 2.4229720, 7.1031745, 2.8
611312, 14.4756500, 1.5835830, 2.2066335, 1.8951324, 0.3898339, 3.3802935, 0.4462456, 2.1215526, 2.379692
6,…
$ `1972`           <dbl> 0.06628651, 2.87088143, 2.42939569, 2.43533977, 2.99313194, 2.49497311, 6.8423657
2, 2.87807728, 13.00683493, 1.59481479, 2.18085819, 1.68898653, -0.11431751, 1.83814626, 0.58119035, 2.032
9…
$ `1973`           <dbl> 0.8628035, 2.9039383, 2.5244212, 2.5117473, 3.2993743, 2.3625522, 6.6133145, 2.91
60948, 11.8082030, 1.6432617, 2.1963571, 1.6469131, -0.3592503, 1.5288169, 0.5540414, 1.8824020, -0.216623
9…
$ `1974`           <dbl> 1.11062929, 2.98629565, 2.51300653, 2.59704681, 3.35235975, 2.29721418, 6.3915710
1, 2.98937329, 10.81050440, 1.70070710, 2.22108033, 1.67884237, -0.38078706, 2.53122026, 0.17020577, 1.807
4…
```

```
$ `1975`           <dbl> 1.13796645, 2.98240910, 2.39328715, 2.68355099, 3.32832392, 2.30115381, 6.1052082
9, 3.19129855, 9.94474064, 1.65927396, 2.25615113, 1.91522454, -0.28537364, 1.23118589, -0.26531941, 1.683
0…
$ `1976`           <dbl> 0.78420194, 2.96980991, 2.17951744, 2.69707336, 3.27407285, 2.20823525, 5.8118718
0, 3.56895909, 12.24485049, 1.58995977, 2.28792076, 2.09139141, -0.07029877, 1.00265826, -0.17667228, 1.60
3…
$ `1977`           <dbl> 0.44351032, 2.88409602, 2.08695113, 2.73573182, 3.31744868, 2.21325218, 5.4957971
5, 3.50439803, 14.05508810, 1.59811229, 2.36155499, 2.02583386, 0.19827182, 1.12667268, 0.03839049, 1.6325
1…
$ `1978`           <dbl> 0.44317175, 3.01273283, 1.96337689, 2.81403788, 3.39870125, 2.07574190, 5.1173735
7, 3.14630092, 13.02951516, 1.60546261, 2.42197260, 1.96651797, 0.33010011, 1.16288526, -0.08096104, 1.687
4…
$ `1979`           <dbl> 0.43315299, 3.14259397, 0.36649281, 2.85041024, 3.43544752, 1.98945702, 4.7013873
8, 3.36397122, 12.06128593, 1.60065891, 1.91794944, 1.94419496, 0.43431795, 1.08064227, -0.17046365, 1.579
6…
$ `1980`           <dbl> 0.4200435875, 2.9654062360, -3.9241716935, 2.8450107720, 3.5039846551, 2.04796386
14, 4.2126280936, 3.5137507046, 10.9963468211, 1.5849187137, 1.2215779148, 2.5905870336, 0.4324397849, 1.2
1…
$ `1981`           <dbl> 0.555730484, 3.017564028, -11.275324328, 2.849793534, 3.554422209, 2.002974306,
3.791192066, 3.226781403, 8.152356145, 1.580609601, 1.039292401, 3.504727956, 0.287769983, 1.561794161, 0.
25…
$ `1982`           <dbl> 0.79854079, 3.13076216, -10.05300625, 2.89550029, 3.59219488, 2.11327178, 4.26339
621, 3.15920309, 5.96957864, 1.57328621, 1.07177181, 3.76848671, 0.23789984, 1.73374393, 0.07171702, 1.582
2…
$ `1983`           <dbl> 0.894334781, 3.012800811, -1.365707744, 2.720131076, 3.617136533, 2.120885334, 4.
642108899, 3.222302386, 5.798180645, 1.558782118, 1.126667159, 3.795079709, 0.270975908, 1.368501459, -0.1
6…
$ `1984`           <dbl> 0.768049326, 2.939088114, 2.894334707, 2.615808644, 3.609872412, 2.103936618, 4.2
34833204, 3.156908918, 5.560088354, 1.537152775, 1.038687034, 3.826748914, -0.070352530, 1.200542766, -0.0
0…
$ `1985`           <dbl> 0.472593610, 2.970292947, 2.587696933, 2.705308376, 3.603751009, 2.055994609, 3.7
86939304, 3.047306020, 5.299493162, 1.513402899, 0.876135492, 3.842129498, -0.481559093, 1.332096835, 0.04
6…
$ `1986`           <dbl> 0.11625115, 3.02226384, -0.60856087, 2.71153904, 3.56585604, 1.93322081, 4.017808
39, 3.04128075, 6.26284024, 1.51132783, 0.79317527, 3.85290720, -0.51788032, 1.44650206, 0.06354900, 1.490
3…
$ `1987`           <dbl> -0.15968624, 3.06281661, -1.21019040, 2.71360186, 3.43867374, 1.99703996, 4.35504
255, 2.96043637, 7.04566355, 1.51097418, 0.72417277, 3.85710946, -0.55631027, 1.52113890, 0.06328420, 1.53
9…
$ `1988`           <dbl> -0.18325548, 2.91918187, 0.58631831, 2.73106066, 3.36704278, 1.88671053, 4.151116
60, 2.67489807, 6.61502193, 1.50808699, 0.46065978, 3.86891020, -0.57817592, 1.63614876, 0.14157085, 1.433
2…
$ `1989`           <dbl> 0.409531473, 2.844887621, 2.751877293, 2.766997853, 3.371350950, 2.687861989, 3.9
19824976, 2.607711150, 6.226055690, 1.487655378, 2.387427069, 3.867471637, -0.542244586, 1.692879694, 0.45
0…
$ `1990`           <dbl> 1.71290678, 2.91101287, 0.20243396, 2.65493103, 3.34514365, 1.79908559, 3.6693341
6, 3.15126868, 5.86903318, 1.45640251, 3.09395853, 3.25193894, -0.48517785, 1.48004694, 0.76200161, 1.3844
1…
$ `1991`           <dbl> 3.22241453, 2.79241036, 0.46988041, 2.62836880, 3.32613717, -0.60280968, 3.422258
14, 2.17496133, 5.53965640, 1.42406316, 1.70315108, 2.42548818, 0.48203493, 1.27462179, 0.99841810, 1.3304
4…
$ `1992`           <dbl> 3.37286419, 2.63368226, 11.52252442, 2.73042096, 3.24885002, -0.60643473, 3.28107
717, 2.49581315, 5.23067556, 1.38743464, -1.19786988, 2.19247704, 1.59793880, 1.11959770, 1.10055229, 1.51
1…
$ `1993`           <dbl> 3.04193161, 2.65852640, 14.96445487, 2.67136264, 3.16154667, -0.61016579, 3.21741
267, 2.70328024, 4.93487058, 1.35796579, -3.33385824, 2.16498007, 1.80091155, 0.88953983, 0.82462794, 1.51
5…
$ `1994`           <dbl> 3.19602915, 2.61995454, 9.86426207, 2.63060971, 3.19843254, -0.61388051, 3.130221
54, 2.71691367, 4.64639136, 1.34702379, -2.44881922, 2.15353087, 1.86302471, 0.96309249, 0.38486955, 1.348
```

4…
$ `1995`   <dbl> 3.084060978, 2.627107030, 6.046535552, 2.693235279, 3.289675826, -0.617703658, 3.
051097802, 2.486878866, 4.408176870, 1.317554245, -1.521153442, 2.106257315, 1.957691518, 1.113734221, 0.1
5…
$ `1996`   <dbl> 3.02578894, 2.69309773, 4.10302205, 2.68749941, 3.33013250, -0.62151140, 1.918610
89, 2.40435281, 5.54384710, 1.26041059, -0.72139105, 2.03712293, 2.02617716, 1.21385498, 0.13501983, 1.011
7…
$ `1997`   <dbl> 3.0273588, 2.5803841, 3.9106052, 2.6890289, 3.3351538, -0.6254301, 0.8305631, 2.3
687982, 6.5219107, 1.1982645, -0.8364943, 1.9586738, 2.0140239, 1.0820399, 0.1133166, 0.9646738, -0.112110
6…
$ `1998`   <dbl> 3.00480452, 2.51465393, 3.88293060, 2.72194072, 3.27906775, -0.62933439, 0.776176
63, 2.31857288, 6.17818856, 1.15817764, -0.94804616, 1.82745478, 1.88335771, 0.99673472, 0.10972837, 0.949
1…
$ `1999`   <dbl> 2.93441293, 2.55966248, 4.07787948, 2.70629131, 3.22644785, -0.63335227, 0.716903
74, 2.28668226, 5.86451456, 1.15204385, -1.07096569, 1.73564083, 1.70640715, 1.09397580, 0.19456315, 0.877
5…
$ `2000`   <dbl> 2.53923444, 2.58357924, 1.44380302, 2.74959972, 3.24412147, -0.63735683, 0.670960
07, 2.28593432, 5.58038700, 1.13327702, -1.17678629, 1.09822902, 1.65779341, 1.14447285, 0.24046665, 0.821
5…
$ `2001`   <dbl> 1.76875662, 2.58996060, 0.74251683, 2.79965352, 3.28521723, -0.93847043, 2.573377
66, 2.24560150, 5.31707596, 1.09917146, -1.12320863, 0.16129866, 1.53371156, 1.28396809, 0.38279939, 0.774
7…
$ `2002`   <dbl> 1.19471806, 2.60659803, 6.44932148, 2.81145269, 3.33513161, -0.29987670, 4.369371
50, 2.21670748, 5.06487258, 1.07353835, -0.90078308, -0.25235848, 1.27763944, 1.13753874, 0.49198046, 0.74
6…
$ `2003`   <dbl> 0.99739555, 2.61776428, 7.54101896, 2.81648065, 3.41332121, -0.37414917, 4.225669
44, 2.18467652, 4.82934269, 1.03236086, -0.67651025, -0.40648362, 1.13352151, 1.15019273, 0.48713389, 0.75
7…
$ `2004`   <dbl> 0.90098923, 2.64496843, 3.93317769, 2.82864223, 3.50638882, -0.41793138, 4.012736
71, 2.17779358, 4.60952674, 1.01533668, -0.59699223, -0.54513967, 1.10308348, 1.06919812, 0.62041313, 0.87
5…
$ `2005`   <dbl> 1.00307718, 2.66624159, 3.57650800, 2.84121122, 3.55765899, -0.51179012, 3.691435
27, 2.33332704, 6.95572690, 1.03347622, -0.60523748, -0.64763458, 1.16870545, 1.21729073, 0.68126725, 1.02
2…
$ `2006`   <dbl> 1.18156555, 2.70252112, 4.13967804, 2.83277974, 3.59201331, -0.63091124, 0.493606
01, 2.54065843, 13.48367209, 1.03467174, -0.68360214, -0.73099869, 1.27642250, 1.34945083, 0.49479778, 1.0
9…
$ `2007`   <dbl> 1.22771081, 2.74710885, 1.79319573, 2.82840930, 3.63958930, -0.75571876, -2.59249
693, 2.67571743, 18.12798398, 1.00629733, -0.73266596, -0.80198275, 1.37623327, 1.82499680, 0.32414654, 1.
1…
$ `2008`   <dbl> 1.24139738, 2.76581598, 2.00233326, 2.83291615, 3.67090920, -0.76734296, -2.74035
939, 2.70585751, 17.39908600, 0.99229409, -0.70049222, -0.87643292, 1.49457906, 2.00391141, 0.31304144, 2.
0…
$ `2009`   <dbl> 1.23323132, 2.75574530, 3.56128837, 2.84403720, 3.69348248, -0.67389405, -2.93936
722, 2.63233227, 13.42292060, 1.01428389, -0.64310612, -0.94376785, 1.52936795, 2.06083316, 0.26195319, 2.
0…
$ `2010`   <dbl> 1.13154104, 2.75792914, 2.89490410, 2.84653574, 3.73479766, -0.49646196, -3.20999
419, 2.43784493, 5.93976537, 0.25558240, -0.60917975, -0.93817319, 1.36406605, 1.55570626, 0.24039430, 1.1
8…
$ `2011`   <dbl> 0.93935591, 2.73818485, 3.68950830, 2.84882577, 3.75879639, -0.26901733, -1.34005
355, 2.17420488, 1.09556271, 1.15305928, -0.58948961, -0.98755830, 1.19938337, 1.38952732, 0.33708084, 1.3
0…
$ `2012`   <dbl> 0.81023059, 2.74040483, 4.07762773, 2.81285093, 3.75870252, -0.16515104, 0.630034
57, 2.15719619, 1.04134461, 1.13690569, -0.49817016, -1.14629823, 1.08370756, 1.74582000, 0.45593747, 1.32
8…
$ `2013`   <dbl> 0.74930104, 2.78015724, 3.46678830, 2.76183898, 3.73552543, -0.18321138, 0.497261
88, 2.29982350, 0.99764183, 1.11910920, -0.44829631, -1.30478202, 0.93432629, 1.72115144, 0.58938725, 1.29
3…
$ `2014`   <dbl> 0.69161526, 2.77499163, 3.65757607, 2.75073057, 3.68442897, -0.20704700, 0.355274

```
94, 2.25922567, 0.95639762, 1.09946109, -0.39559288, -1.47894571, 0.83158925, 1.49156649, 0.78154163, 1.24
8…
$ `2015`           <dbl> 0.63795916, 2.80258714, 3.12134123, 2.72331734, 3.61767752, -0.29120579, 0.174377
69, 2.15596631, 0.91195004, 1.07800134, -0.39299525, -1.63927019, 0.78693542, 1.43921665, 1.12099250, 1.19
1…
$ `2016`           <dbl> 0.59006249, 2.72815978, 2.58154940, 2.71305851, 3.58621101, -0.15988041, 1.100602
99, 2.10969713, 0.86386892, 1.05718158, -0.44425717, -1.80723077, 0.69028833, 1.56194050, 1.08139630, 1.11
7…
$ `2017`           <dbl> 0.53729571, 2.65567278, 2.86649215, 2.70626608, 3.55098664, -0.09197229, 1.772182
71, 2.06873885, 0.81974447, 1.03713390, -0.48662526, -1.97181875, 0.61095614, 1.65339054, 0.69462110, 0.98
1…
$ `2018`           <dbl> 0.49479526, 2.68837259, 2.88520797, 2.66923867, 3.46445698, -0.24673204, 1.580147
09, 2.09619402, 0.78945011, 1.01580835, -0.54025108, -2.12293567, 0.55487297, 1.50299298, 0.48707192, 0.86
6…
$ `2019`           <dbl> 0.45196966, 2.69113552, 2.90852909, 2.63398207, 3.39527779, -0.42600737, 1.757491
26, 2.06268690, 0.77908718, 0.99339749, -0.56406554, -2.30413870, 0.53444340, 1.48520845, 0.44467364, 0.84
6…
$ `2020`           <dbl> 0.13425530, 2.67818482, 3.13474691, 2.61564614, 3.26834840, -0.57420696, 1.761891
30, 1.75789901, 0.81769439, 0.97005399, -0.53300661, -2.42124971, 0.59205394, 1.23570108, 0.41517673, 0.68
4…
$ `2021`           <dbl> -0.04504462, 2.60747274, 2.85135765, 2.57337740, 3.16602986, -0.92691806, 1.70228
823, 1.62333519, 0.83481279, 0.94749096, -0.52296324, -2.53017093, 0.59715159, 0.12772830, 0.43567168, 0.4
4…
$ `2022`           <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N
A, 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,
…
$ ...68            <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N
A, 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,
…
```

Hide

```
population_na_count <- sum(is.na(population_growth))
population_na_count
```

```
[1] 890
```

Hide

```
# Convert 'Country Name' and 'Country Code' in population_growth to factor
population_growth$`Country Name` <- as.factor(population_growth$`Country Name`)
population_growth$`Country Code` <- as.factor(population_growth$`Country Code`)
```

Provide explanations here. For better understanding in the gdp and population growth data, Data must be loaded and examined, the amount of missing values must be counted, and some variables must be changed to factor type. We do the following:

1. The GDP Dataset

- head(gdp_data): This will print the first 6 rows of the gdp_data dataframe.
- glimpse(gdp_data): This function gives the structure of the dataframe, including the number of observations, variables, and the type of each variable.
- gdp_na_count <- sum(is.na(gdp_data)) and gdp_na_count: These lines count the total number of missing values in the gdp dataset.
- gdp_data\('Country Name' <- as.factor(gdp_data\)'Country Name'): These lines convert the 'Country Name' column to a factor.
- gdp_data\(Code <- as.factor(gdp_data\)Code): These lines convert the 'Code' column to a factor.

2. The Population Growth Dataset

- head(population_growth): This will print the first 6 rows of the population_growth dataframe.
- glimpse(population_growth): This function gives the structure of the dataframe, including the number of observations, variables, and the type of each variable.
- population_na_count <- sum(is.na(population_growth)) and population_na_count: These lines count the total number of missing values in the population_growth dataset.
- population_growth\(`Country Name` <- as.factor(population_growth\) `Country Name`): These lines convert the 'Country Name' column to a factor.
- population_growth\(`Country Code` <- as.factor(population_growth\) `Country Code`): These lines convert the 'Country Code' column to a factor.

Tidy & Manipulate Data I

Hide

```
# This is the R chunk for the Tidy & Manipulate Data I
# Tidying the gdp_data dataset
tidy_gdp_data <- gdp_data %>%
  select(-'Unnamed: 65') %>%
  pivot_longer(cols = `1960`:`2020`, names_to = "Year", values_to = "GDP") %>%
  mutate(Year = as.integer(Year))

str(tidy_gdp_data)
```

```
tibble [16,226 × 4] (S3: tbl_df/tbl/data.frame)
 $ Country Name: Factor w/ 266 levels "Afghanistan",..: 13 13 13 13 13 13 13 13 13 13 ...
 $ Code        : Factor w/ 266 levels "ABW","AFE","AFG",..: 1 1 1 1 1 1 1 1 1 1 ...
 $ Year        : int [1:16226] 1960 1961 1962 1963 1964 1965 1966 1967 1968 1969 ...
 $ GDP         : num [1:16226] NA NA NA NA NA NA NA NA NA NA ...
```

Hide

```
# Tidying the population_growth dataset
tidy_population_growth <- population_growth %>%
  pivot_longer(cols =  -c(1:4), names_to = "Year", values_to = "Population Growth (annual %)") %>%
  mutate(Year = as.integer(Year)) %>%
  select(-c(`Indicator Name`, `Indicator Code`)) %>%
  rename(Code = `Country Code`)
```

```
Warning: There was 1 warning in `mutate()`.
ℹ In argument: `Year = as.integer(Year)`.
Caused by warning:
! NAs introduced by coercion
```

Hide

```
str(tidy_population_growth)
```

```
tibble [17,024 × 4] (S3: tbl_df/tbl/data.frame)
 $ Country Name               : Factor w/ 266 levels "Afghanistan",..: 13 13 13 13 13 13 13 13 13 13 ...
 $ Code                       : Factor w/ 266 levels "ABW","AFE","AFG",..: 1 1 1 1 1 1 1 1 1 1 ...
 $ Year                       : int [1:17024] 1960 1961 1962 1963 1964 1965 1966 1967 1968 1969 ...
 $ Population Growth (annual %): num [1:17024] NA 2.18 1.55 1.39 1.22 ...
```

Hide

```
# Joining the datasets
joined_data <- inner_join(tidy_gdp_data, tidy_population_growth, by = c("Code", "Year"))

# Removing the duplicate Country Name
joined_data <- joined_data %>%
  select(-`Country Name.y`) %>%
  rename(Country = `Country Name.x`)
head(joined_data)
```

Country <fctr>	Code <fctr>	Year <int>	GDP <dbl>	Population Growth (annual %) <dbl>
Aruba	ABW	1960	*NA*	*NA*
Aruba	ABW	1961	*NA*	2.179059
Aruba	ABW	1962	*NA*	1.548572
Aruba	ABW	1963	*NA*	1.389337
Aruba	ABW	1964	*NA*	1.215721
Aruba	ABW	1965	*NA*	1.032841

6 rows

In Tidy & Manipulate Data I, we wanted to tidy and select all of the relevant column to be merged together.

After dropping the ‘Unnamed: 65’ column in gdp_data, we used the pivot_longer function to reshape the data from a wide format to a long format. In this step, the columns from ‘1960’ to ‘2020’, which presumably represent years, are gathered into a single column named “Year”, and the corresponding values are stored in a new column named “GDP”. We then converted the “Year” column to integer type for easier analysis later on.

Similarly, we tidied the population_growth dataset. We reshaped the data to long format using the pivot_longer function, this time excluding the first four columns from reshaping. The remaining columns are gathered into a “Year” column and their corresponding values into a “Population Growth” column. The “Year” column is again converted to integer type.

We also dropped the columns “Indicator Name” and “Indicator Code” as they aren’t necessary for our analysis. The Country Code column is renamed to Code for consistency and the Population growth is renamed to Population Growth (annual %).

Next, we joined the two datasets, tidy_gdp_data and tidy_population_growth, using an inner join operation. This operation returns all rows from both tables where there is a match based on the specified conditions, in this case, the “Code” and “Year”. This means we’re looking to analyze the GDP and population growth for each country and each year that is available in both datasets.

Finally, to avoid confusion with duplicate columns, we dropped the Country Name.y column (which presumably comes from the tidy_population_growth dataset) and renamed the Country Name.x column (from the tidy_gdp_data dataset) to just “Country”. This leaves us with a single “Country” column in our final, joined dataset.

Tidy & Manipulate Data II

Hide

```
# This is the R chunk for the Tidy & Manipulate Data II
joined_data <- joined_data %>%
  arrange(Code, Year) %>%
  group_by(Code) %>%
  mutate(`GDP_growth (%)` = (GDP/lag(GDP) - 1)*100)%>%
  mutate(`GDP_growth (%)`= round(`GDP_growth (%)`, 2))

head(joined_data)
```

Country <fctr>	Code <fctr>	Year <int>	G… <dbl>	Population Growth (annual %) <dbl>	GDP_growth (%) <dbl>
Aruba	ABW	1960	*NA*	*NA*	*NA*
Aruba	ABW	1961	*NA*	2.179059	*NA*
Aruba	ABW	1962	*NA*	1.548572	*NA*
Aruba	ABW	1963	*NA*	1.389337	*NA*
Aruba	ABW	1964	*NA*	1.215721	*NA*
Aruba	ABW	1965	*NA*	1.032841	*NA*

6 rows

In this step, we're creating a new variable from existing variables in our dataset. "GDP_growth" will represent the annual percentage change in GDP for each country and year. This can provide us with valuable insights about the rate of economic growth in each country, year by year. The code explanation is as following:

- arrange(Code, Year): We're arranging (sorting) the data first by "Code" (which presumably represents each country) and then by "Year". This ensures that our data is in the correct order for our next calculations.
- group_by(Code): We're grouping the data by "Code". This means that the following operations (specifically, the mutate function) will be applied within each group of rows that have the same "Code". In other words, it's ensuring that our calculations are done for each country separately.
- mutate(GDP_growth (%)= (GDP/lag(GDP) - 1)*100): This is where we're actually creating the new "GDP_growth" variable. The lag(GDP) function gives us the GDP value from the previous row (which, because of how we arranged our data, represents the previous year's GDP for the same country). We're then subtracting this value from the current row's GDP, dividing by the lagged GDP to get the relative change, and multiplying by 100 to convert it to a percentage.
- round('GDP_growth (%)', 2) will round the "GDP_growth (%)" variable to 2 decimal places. This will keep "GDP_growth" as a numeric variable while ensuring that it only has two numbers after the decimal point. This can make the data easier to read and interpret, while still retaining its numerical properties for further calculations.

Scan I

Hide

```
# This is the R chunk for the Scan I
#Make a new data
joined_data_fill <- joined_data

#GDP Value added
# fit a linear regression model using the available data
model <- lm(GDP ~ `Population Growth (annual %)`, data = joined_data_fill[!is.na(joined_data_fill$GDP),])

# predict missing GDP values
predicted_GDP <- predict(model, newdata = joined_data_fill)

# replace only NA GDP values with predicted values
joined_data_fill$GDP[is.na(joined_data_fill$GDP)] <- predicted_GDP[is.na(joined_data_fill$GDP)]

#Fill the remaining NA values
joined_data_fill <- joined_data_fill %>%
  group_by(Code) %>%
  mutate(GDP = ifelse(Year == min(Year), mean(GDP, na.rm = TRUE), GDP))

#Remove the row if it is still NA
joined_data_fill <- joined_data_fill[!is.na(joined_data_fill$GDP), ]
which(is.na(joined_data_fill$GDP))
```

```
integer(0)
```

Hide

```
# Update the 'GDP_growth' after 'GDP' with NA variables filled
joined_data_fill <- joined_data_fill %>%
  arrange(Code, Year) %>%
  group_by(Code) %>%
  mutate(`GDP_growth (%)` = (GDP/lag(GDP) - 1)*100) %>%
  mutate(`GDP_growth (%)`= round(`GDP_growth (%)`, 2)) %>%
  mutate(`GDP_growth (%)` = ifelse(Year == min(Year), 0, `GDP_growth (%)`))

# Population Growth Value Added
# Filled the NA value in the first year to 0
joined_data_fill <- joined_data_fill %>%
  group_by(Code) %>%
  mutate(`Population Growth (annual %)` = ifelse(Year == min(Year), 0, `Population Growth (annual %)`))

#Filled New Zealand population growth missing value no in the first year
joined_data_fill <- joined_data_fill %>%
  group_by(Code) %>%
  mutate(`Population Growth (annual %)` = na.approx(`Population Growth (annual %)`, na.rm = TRUE))

# Scan again after alteration on the joined_data
head(sapply(joined_data_fill, is.infinite))
```

```
     Country  Code  Year   GDP Population Growth (annual %) GDP_growth (%)
[1,]   FALSE FALSE FALSE FALSE                         FALSE          FALSE
[2,]   FALSE FALSE FALSE FALSE                         FALSE          FALSE
[3,]   FALSE FALSE FALSE FALSE                         FALSE          FALSE
[4,]   FALSE FALSE FALSE FALSE                         FALSE          FALSE
[5,]   FALSE FALSE FALSE FALSE                         FALSE          FALSE
[6,]   FALSE FALSE FALSE FALSE                         FALSE          FALSE
```

Hide

```
head(sapply(joined_data_fill, is.finite))
```

```
     Country Code Year  GDP Population Growth (annual %) GDP_growth (%)
[1,]    TRUE TRUE TRUE TRUE                         TRUE           TRUE
[2,]    TRUE TRUE TRUE TRUE                         TRUE           TRUE
[3,]    TRUE TRUE TRUE TRUE                         TRUE           TRUE
[4,]    TRUE TRUE TRUE TRUE                         TRUE           TRUE
[5,]    TRUE TRUE TRUE TRUE                         TRUE           TRUE
[6,]    TRUE TRUE TRUE TRUE                         TRUE           TRUE
```

Hide

```
#check again data for Nan values
na_positions <- lapply(joined_data_fill, function(x) which(is.na(x)))
print(na_positions)
```

```
$Country
integer(0)

$Code
integer(0)

$Year
integer(0)

$GDP
integer(0)

$`Population Growth (annual %)`
integer(0)

$`GDP_growth (%)`
integer(0)
```

Hide

```
# Check for duplicate rows
duplicate_rows <- duplicated(joined_data_fill)
print(sum(duplicate_rows))
```

```
[1] 0
```

First we inspect the data and found that GDP is a field with significant missing data. In some cases, GDP data is entirely absent for certain countries during the early years of data collection. This could be due to a variety of factors, such as insufficient data recording practices or changes in national boundaries and governance.

To address this problem, the script employs a linear regression model to predict missing GDP values. The model uses the ‘Population Growth’ variable as an independent predictor, assuming there is a positive relationship between a country’s population growth and its GDP. This is a reasonable assumption in many cases because increased population can lead to increased demand, productivity, and, therefore, GDP.

The linear regression model is trained on the non-missing values in the GDP dataset. The predicted GDP values are then used to fill in the gaps in the original data where GDP was missing.

However, there may still be missing GDP values if the first year or multiple initial years of data for a country are missing. This is because the linear regression model cannot extrapolate beyond the range of years in the training data.

To handle these remaining missing values, the script then uses a group-wise operation to fill missing GDP values with the average GDP of the respective country. This operation assumes that the missing GDP values would be similar to the mean GDP value for the given country.

Finally, the script updates the 'GDP_growth (%)' field based on the now filled-in GDP data. GDP growth is calculated as the percentage change in GDP from the previous year, and missing values in the first year are filled with 0 as a reasonable assumption.

At the end of these operations, we expect that all missing values in GDP have been filled appropriately based on the available data and assumptions.

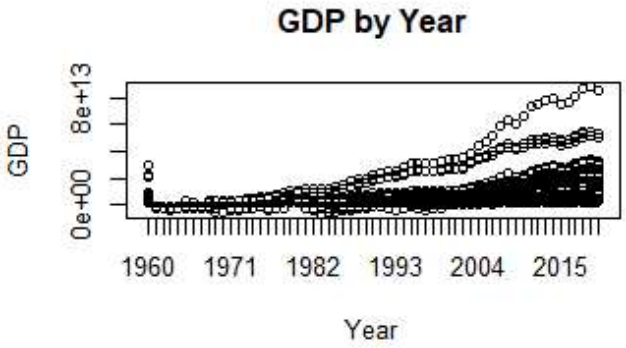
Scan II

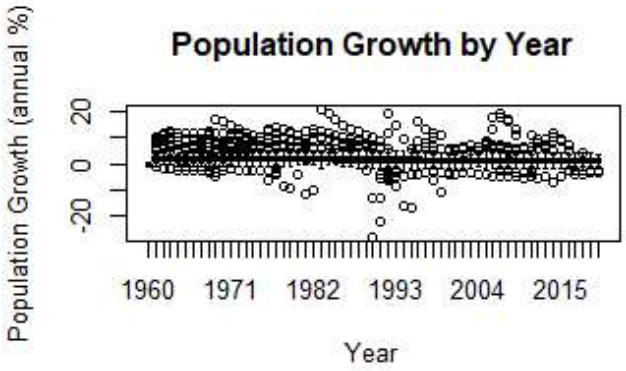
Hide

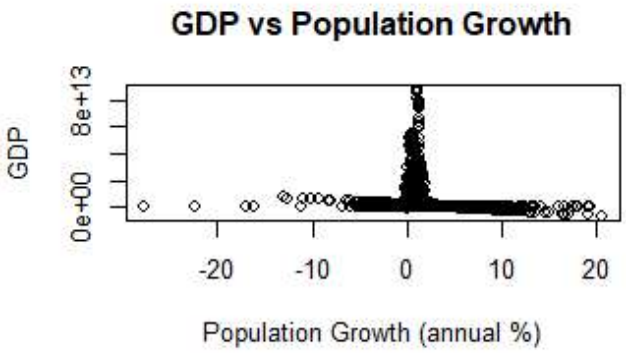
```
# This is the R chunk for the Scan II
#understanding the relationship between gdp and population growth therefore use the multivariate outliers
method

#bivariate
#shows us the outlier for each year for gdp and population growth separately
# Set up the graphics to allow two plots
par(mfrow = c(2, 2))

# Plot the first boxplot
boxplot(joined_data_fill$GDP ~ joined_data_fill$`Year`,
        main = "GDP by Year",
        xlab = "Year",
        ylab = "GDP")

# Plot the second boxplot
boxplot(joined_data_fill$`Population Growth (annual %)` ~ joined_data_fill$`Year`,
        main = "Population Growth by Year",
        xlab = "Year",
        ylab = "Population Growth (annual %)")
```

Hide

```


# Plot the multivariate scatterplot
plot(joined_data_fill$GDP ~ joined_data_fill$`Population Growth (annual %)`,
     main = "GDP vs Population Growth",
     xlab = "Population Growth (annual %)",
     ylab = "GDP")
#shows the distributions of the GDP over the population growth
#the gdp has the most growth when the population growth is between -5 and 5

# Reset graphics parameters to default
par(mfrow = c(1, 1))
```







Hide

```
#excluding values that aren't necessary and doesn't help us investigate what we want to
joined_data_gdp <- joined_data_fill %>% select(Year, GDP)
```

```
Adding missing grouping variables: `Code`
```

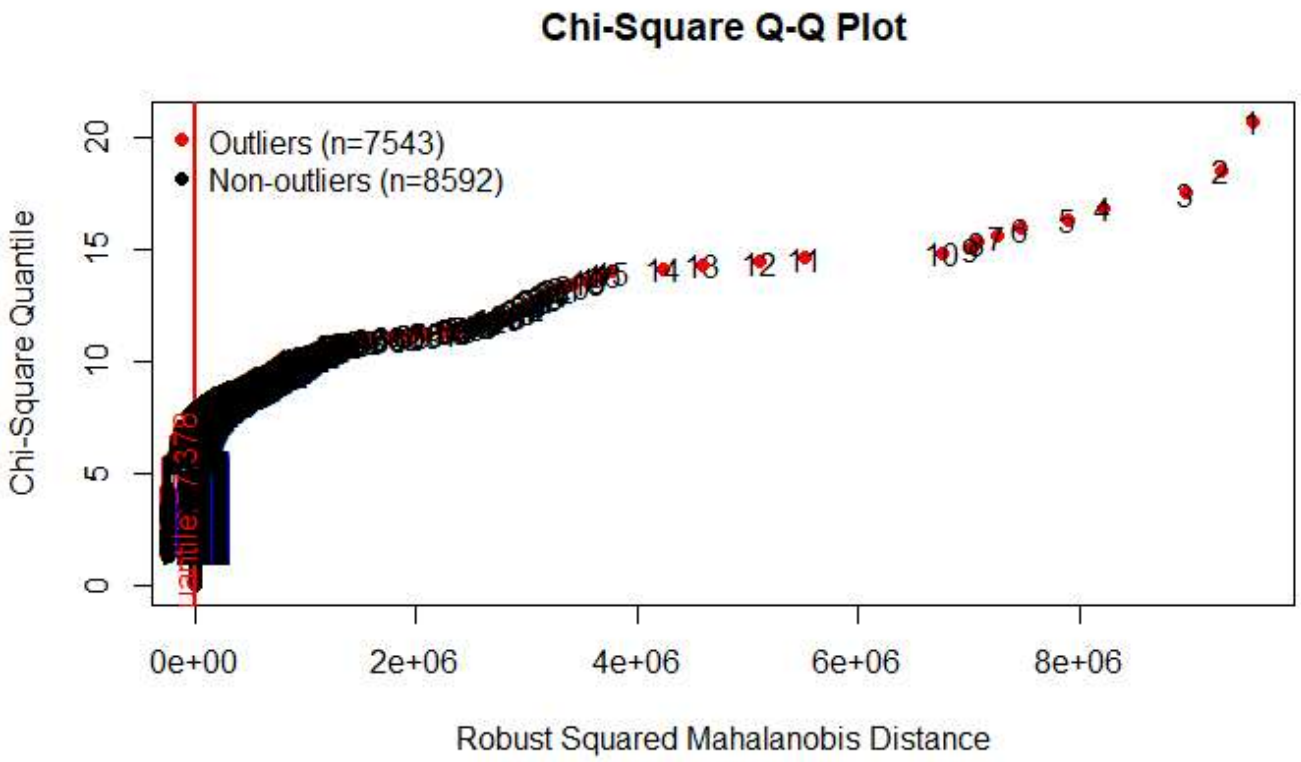
Hide

```
joined_data_population <- joined_data_fill %>% select(Year, `Population Growth (annual %)`)
```

```
Adding missing grouping variables: `Code`
```

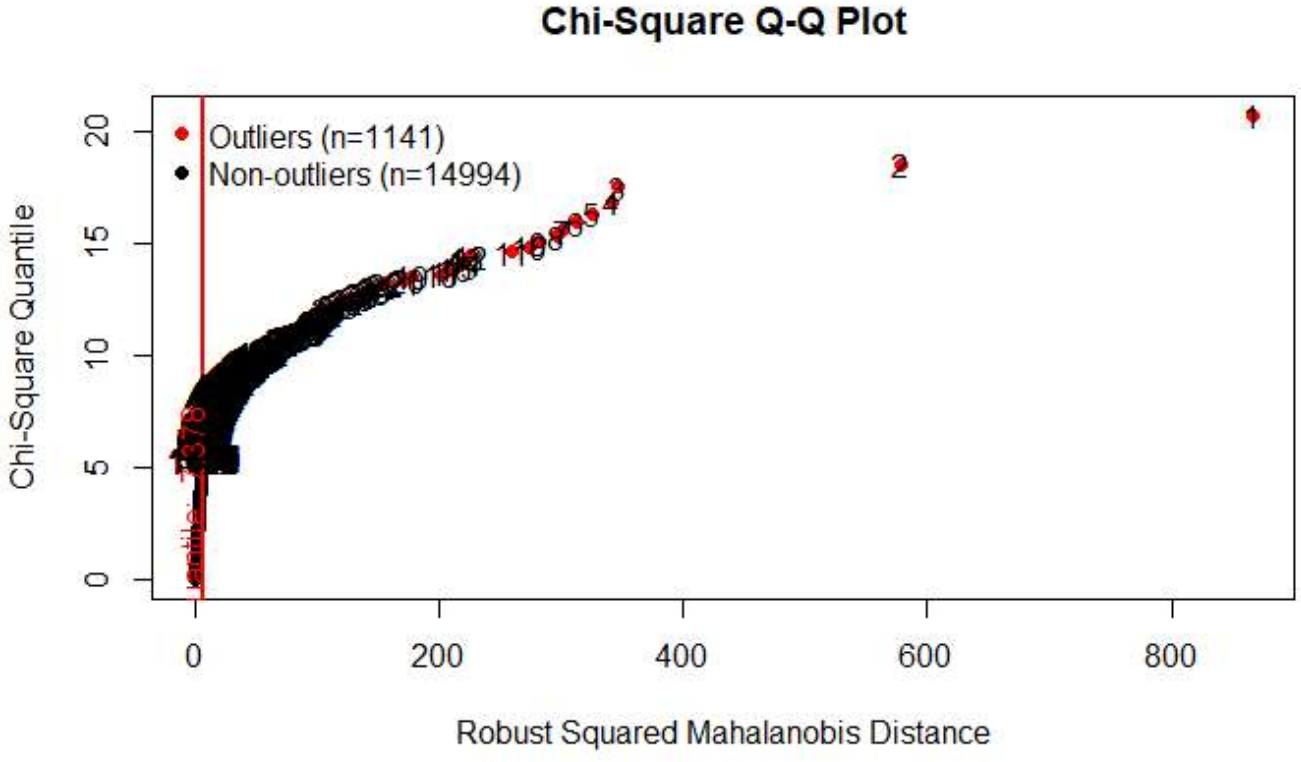
Hide

```
#removing Code which is the variable that joined the two variables as we are now directly working with the
Year and gdp and the year and population growth
gdp <- joined_data_gdp[ , unlist(lapply(joined_data_gdp, is.numeric))]
pop_growth <- joined_data_population[ , unlist(lapply(joined_data_population, is.numeric))]


#using argument in MVN package
#better to use second approach as it is a built in function and doing so manually may result in incorrect
imputations
#finding and noting the outliers
results_gdp <- mvn(data = gdp,
                   multivariateOutlierMethod = "quan",
                   showOutliers = TRUE,
                   showNewData = TRUE)
```



Hide

```
results_population <- mvn(data = pop_growth,
                          multivariateOutlierMethod = "quan",
                          showOutliers = TRUE,
                          showNewData = TRUE)
```



Hide

```
#Handle the outliers
cap <- function(x){
    quantiles <- quantile( x, c(0.05, 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}

gdp_capped <- joined_data_fill$GDP %>% cap()
summary(joined_data_gdp)
```

```
      Code            Year           GDP
 ABW    :   61   Min.   :1960   Min.   :-6.397e+12
 AFE    :   61   1st Qu.:1975   1st Qu.: 3.608e+09
 AFG    :   61   Median :1990   Median : 5.197e+10
 AFW    :   61   Mean   :1990   Mean   : 1.117e+12
 AGO    :   61   3rd Qu.:2005   3rd Qu.: 8.063e+11
 ALB    :   61   Max.   :2020   Max.   : 8.757e+13
 (Other):15769
```

Hide

```
summary(gdp_capped)
```

```
      Min.    1st Qu.     Median       Mean    3rd Qu.       Max.
-1.190e+12  3.608e+09  5.197e+10  5.660e+11  8.063e+11  3.388e+12
```

Hide

```
pop_capped <- joined_data_fill$`Population Growth (annual %)`%>% cap()
summary(joined_data_population)
```

```
      Code            Year      Population Growth (annual %)
 ABW    :   61   Min.   :1960   Min.   :-27.7222
 AFE    :   61   1st Qu.:1975   1st Qu.:  0.7344
 AFG    :   61   Median :1990   Median :  1.7422
 AFW    :   61   Mean   :1990   Mean   :  1.7510
 AGO    :   61   3rd Qu.:2005   3rd Qu.:  2.6209
 ALB    :   61   Max.   :2020   Max.   : 20.4732
 (Other):15769
```

Hide

```
summary(pop_capped)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
-2.0813  0.7344  1.7422  1.6961  2.6209  5.4474
```

- Our primary and merged dataset is multivariate; therefore, we need to be able to use the multivariate/bivariate outlier method. Our main point is to build an understanding of the relationship between the GDP and Population growth.
- We first use the bivariate outlier detection method to create a boxplot to see the relationship between the Year and the GDP and the Year and the population growth. This shows us how each variable is affected over time and visually represents potential outliers.
- We also use the multivariate scatterplot to depict the relationship between GDP and population growth, showing that GDP slightly increases as the population increases. However, once the population growth percentage exceeds 10%,

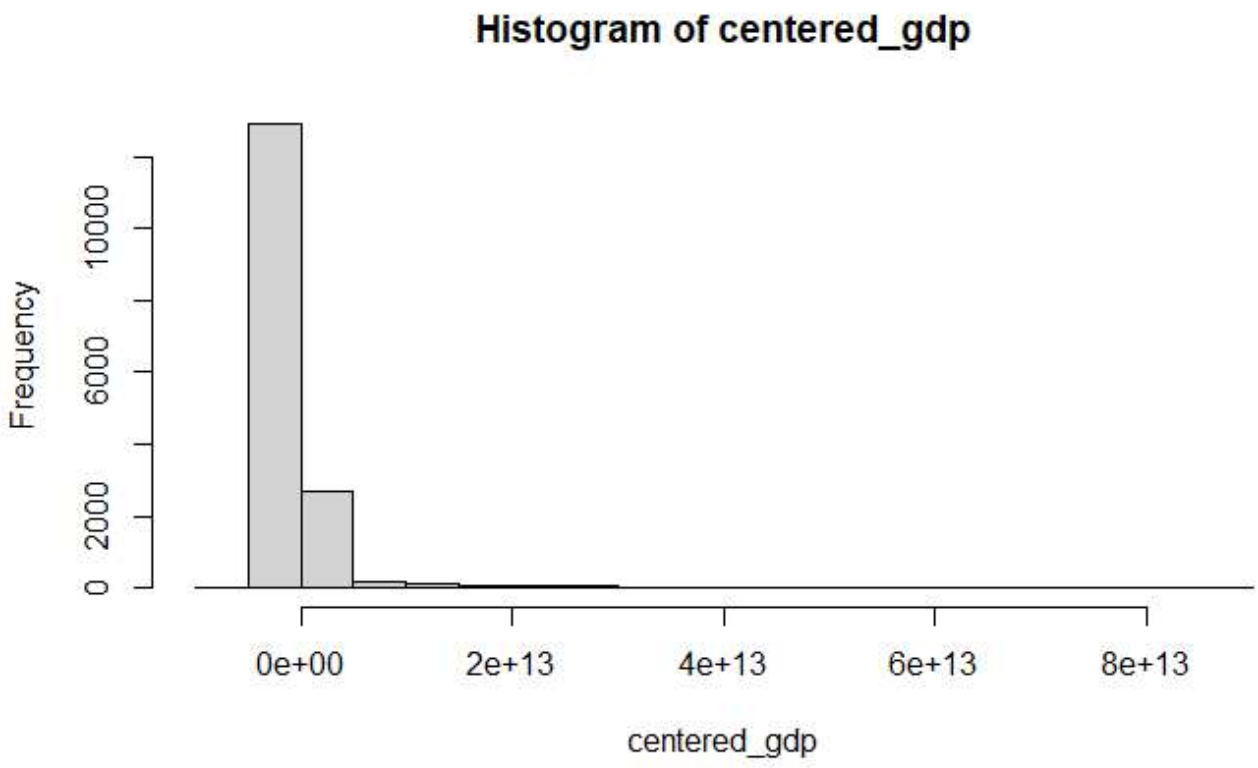
there is minimal GDP increase. The scatterplot is also able to help us detect abnormal observations.

- We then subset the data to only reference the numerical variables we are directly referencing, which is the relationship between the Year and GDP and between the Year and the Population Growth.
- As earlier on, we merged the two datasets using a common variable of 'Code', which is a factor, we have to unlist in to do computation using the MVN package and make sure that the class of the variables in question is class numeric.
- We use an MVN built-in function to visually represent the outliers using our subsetted GDP and Population growth data. Using the built-in function multivariateOutlierMethod = "quan" allows us to detect outliers using the chi-squared distribution and present them on a QQ plot.
- We used the capping method to handle the outliers we were presented with, as it felt irresponsible to delete the outliers as they can still contain valuable information by creating a cap function detailing the mathematics behind the outliers and assigning variables to them.
- Utilising the function, we use them with the joined_data_fill dataset, which doesn't contain missing variables and directly references GDP and population growth.
- By showing the summary of the joined_data_fill dataset and the capped dataset, we can see how the function manipulates the minimum values, maximum values, quartile 1, median, quartile three and the mean.

Transform

Hide

```
# This is the R chunk for the Transform Section
#centering and scaling
centered_gdp <- scale(joined_data_fill$GDP, center = TRUE, scale = FALSE)
hist(centered_gdp)
```



Hide

```
#squared
sqr_gdp<- joined_data_fill$GDP^2

#cubed
cubed_gdp <- joined_data_fill$GDP^3

#log
log_gdp <- log10(joined_data_fill$GDP)
```

```
Warning: NaNs produced
```

Hide

```
#ln
ln_gdp <- log(joined_data_fill$GDP)
```

```
Warning: NaNs produced
```

Hide

```
#sqrt
sqrt_gdp <- sqrt(joined_data_fill$GDP)
```

```
Warning: NaNs produced
```

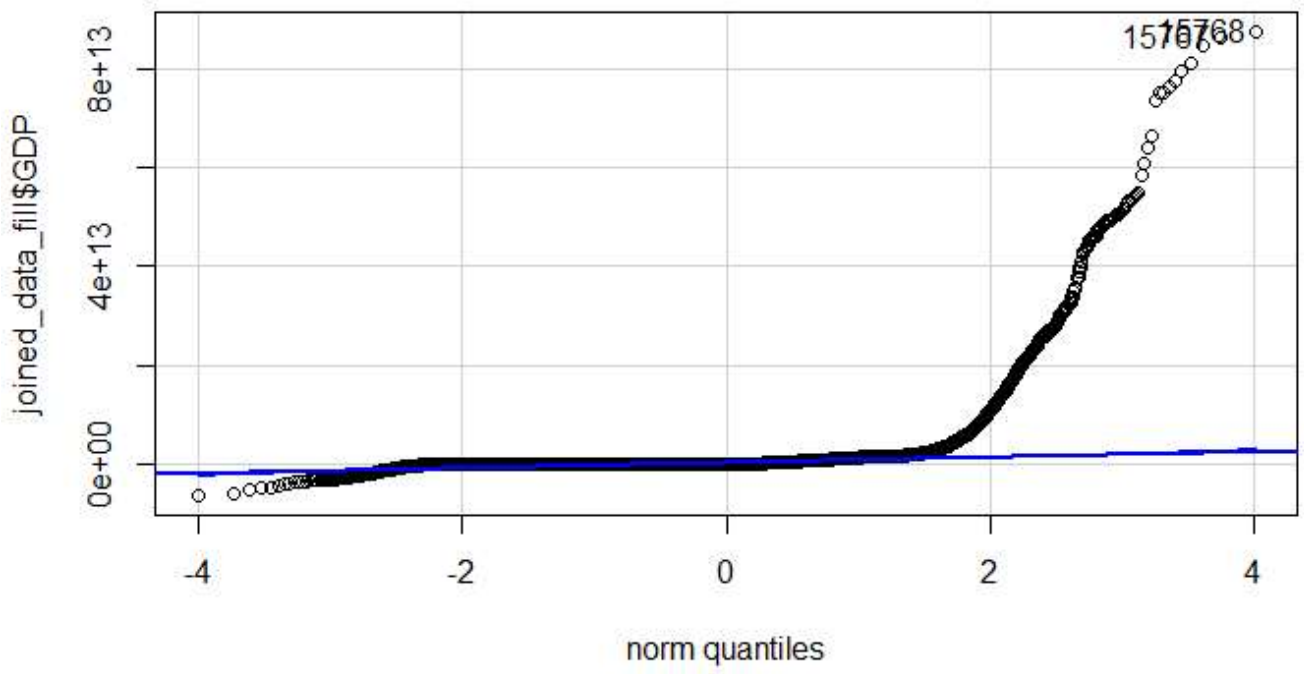
Hide

```
#cuberoot
cubrt_gdp <- joined_data_fill$GDP^(1/3)

#reciprocol
rec_gdp <- 1/joined_data_fill$GDP

#check normality
#qqplot
car :: qqPlot(joined_data_fill$GDP, dist = "norm")
```

```
[1] 15768 15767
```

Hide

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



Hide

```
hist(joined_data_fill$GDP, breaks = 30, main = "Histogram of GDP")
hist(sqr_gdp, breaks = 30, main = "Histogram of Squared\n Tranformation GDP")
```

Hide

```
hist(cubed_gdp, breaks = 30, main = "Histogram of Cubed\n Transformation GDP")
hist(log_gdp, breaks = 30, main = "Histogram of Log\n Transformation GDP")
```

Hide

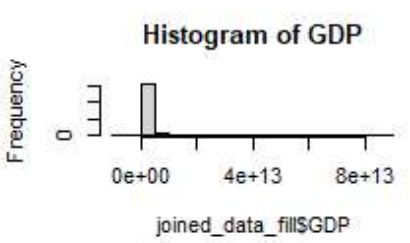
```
hist(ln_gdp, breaks = 30, main = "Histogram of Natural Log\n Transformation GDP")
hist(sqrt_gdp, breaks = 30, main = "Histogram of Square Root\n Transformation of GDP")
```

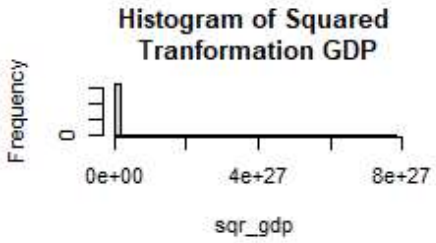
Hide

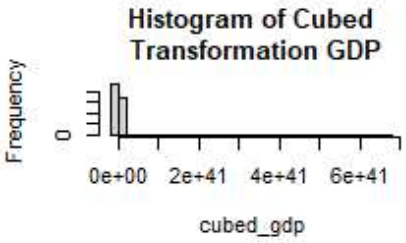
```
hist(cubrt_gdp, breaks = 30, main = "Histogram of Cubed Root\n Transformation of GDP")
hist(rec_gdp, breaks = 30, main = "Histogram of Recipricol\n Transformation of GDP")
```

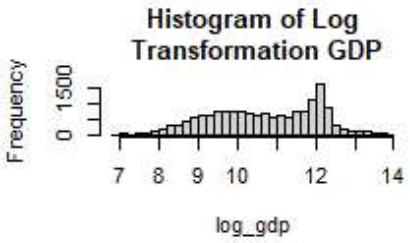
Hide

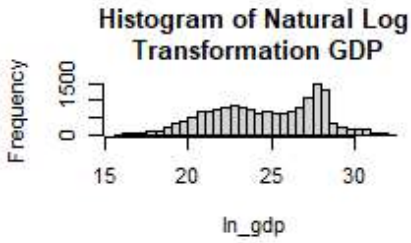
```
par(mfrow = c(1,1))

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

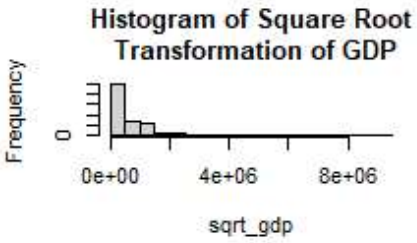


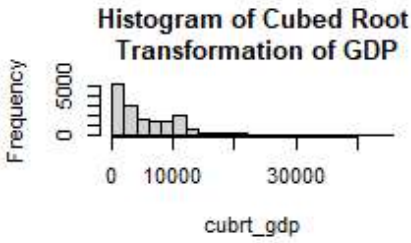


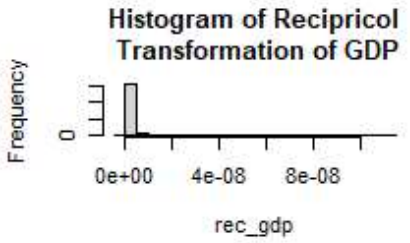












Hide

```
car::qqPlot(sqr_gdp)
```

```
[1] 15768 15767
```

Hide

```
car::qqPlot(cubed_gdp)
```

```
[1] 15768 15767
```

Hide

```
car::qqPlot(log_gdp)
```

```
[1] 14885  5371
```

Hide

```
car::qqPlot(ln_gdp)
```

```
[1] 14885  5371
```

Hide

```
car::qqPlot(sqrt_gdp)
```

```
[1] 15768 15767
```

Hide

```
car::qqPlot(cubrt_gdp)
```

```
[1] 15768 15767
```

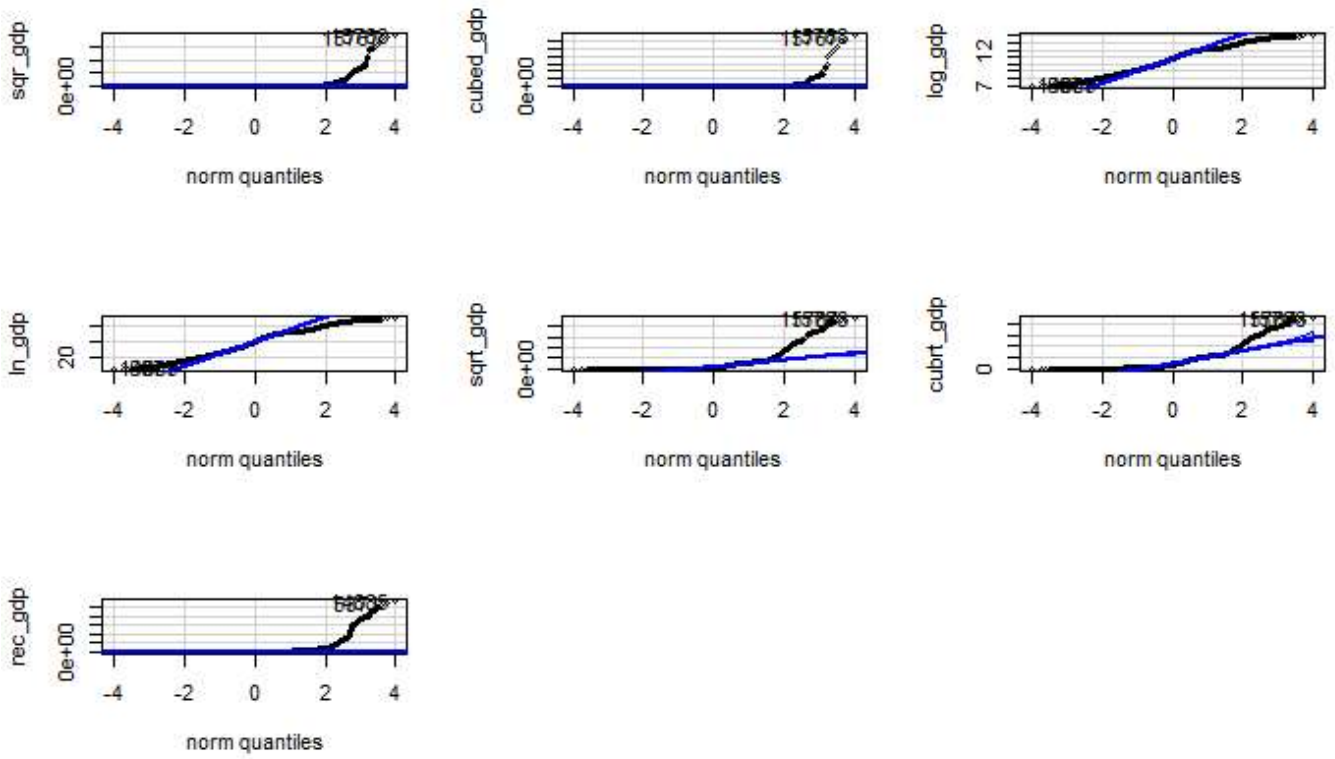
Hide

```
car::qqPlot(rec_gdp)
```

```
[1] 14885  5371
```

Hide

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



The variable we are transforming is the GDP to decrease the skewness and make the distribution normal distribution. By doing a series of transformations, we can see which transformation method is closest to a normal distribution. To do this, we first centre and scale the dataset based on GDP so that in each column, it is transformed so that the variable, GDP, has zero means. This method involves the division of the values to the standard deviation. We also compare the squared, cubed, log, natural log, square root, cubed root and reciprocal transformation to see if we can decrease the skewness and create something closer to a normal distribution where we use breaks of 30 as that can show a decent pattern without overplotting and having everything look that same. To check the normality, we review each of these attributes against the QQ Plot to see if there is a stronger correlation and to estimate the distribution parameters of the fitted distribution.