International expansion of N&D holdings I

Anton Barrera Mora (me@antonio-barrera.cyou)

2023-07-15

Contents

Introduction

We will go through the first 3 phases of the project that will lead to the acquisition of a dataset with which we will tackle the modelling and testing phases in a second part of the project.

Phase 1. Understanding the business

The general shareholders' meeting of the business holding company 'night&day kabushiki gaisha' (N&D.hol) approved in its last meeting the conduct of a descriptive-projective study. This study is expected to encompass different candidate countries - from which one will eventually be selected - as well as an algorithm that allows monitoring the evolution of the various markets where the consortium operates. Furthermore, it will be necessary to take into account the different variables that could affect the success of international expansion, consistent with the various parameters and requirements that will be detailed below.

N&D.hol is a group of Japanese companies, a leader in its sector, with a mission to improve the well-being and health of its customers and a commitment to sustainable development. The main line of business is the management of wellness facilities for the elderly - such as senior residences - but they are also involved in other areas such as the production and distribution of medical supplies and equipment, construction of healthcare infrastructure, marketing of healthy food products, and research.

The holding has been implementing a corporate social responsibility plan for several years. Consequently, it has prioritized alignment with various goals and targets of sustainable development promoted by the UN and other international supranational institutions (Miluska.Jara, n.d.; "Measuring Progress Towards the Sustainable Development Goals - SDG Tracker," n.d.). Specifically, the corporate social responsibility plan has emphasized various initiatives such as:

- Reduction of poverty (**goal1?**):

- Decent work and economic growth (**goal8?**):

- Consistent with the nature of the business group, good health and social well-being (**goal3?**):

Therefore, the business objective is the selection of a data-driven country for the group to initiate international expansion. The key performance indicators would be established as follows:

- Recovery of the total investment in the chosen country within five years.

- 12% net profit in the sixth year.

- Achieve a 15% market share in the country within 3 years.

- Positive assessment and evolution of N&D.hol's perception as an entity committed to sustainable development goals in the markets where it operates, resulting in a 5% annual improvement in brand image perception through traditional surveys, social media, and other similar channels.

Likewise, the situation analysis leading to the selection of candidate countries should consider the following aspects Hennig (2015)

- Cost-benefit analysis.

The trio of countries must meet a preliminary requirement based on the density of the aging population in the coming years, ensuring a sustained high demand for the types of products and services offered by the holding. Additionally, based on the most up-to-date available data, there should be a consistent customer base.

The candidates should be nations that have experienced positive economic growth in recent years in terms of per capita income.

To operate in line with the sustainable development goals outlined in the corporate social responsibility plan, the candidate nations for N&D's international expansion should be countries facing issues such as poverty, inequality, decent work, and economic growth where the company can make a meaningful contribution. However, at the same time, the population should have sufficient income levels to generate demand for the products and services offered by the holding.

- Availability of resources.

The country or nation should have a skilled workforce - including doctors, nurses, and care staff - to support the operations and services that the holding plans to deploy.

Additionally, there should be a network of medical establishments and facilities that provide at least basic healthcare services.

- Risks and contingencies.

The candidates should have the best possible ratio in terms of labor legislation and regulation of private property rights.

Furthermore, they should have the lowest possible ratio of criminal activity.

- Project requirements.

The candidate countries or nations should have the best possible balance in the previously highlighted aspects or variables, accompanied by various data analyses that can help justify a data-driven decision.

The project should be scalable to incorporate new variables for analysis in the future.

Moreover, the outcome of this work should result in a predictive-descriptive model with the ability to update with new data over time, which can assist in future needs or objectives for international expansion.

Regarding the goals of the data mining project, the following are considered:

1. Candidate analysis performed using scientific and data analysis techniques with the established parameters and variables, followed by a ranking and evaluation of the results. The outcome should provide a deliverable with the top 3 candidates for investment.
2. Predictive analysis on:

- Economic trends.
- Evolution of different sustainable development indices.
- Evolution of various sociocultural aspects considered in the data mining project.

Therefore, as a summary, the expected deliverable is an updatable descriptive-predictive model that meets the present and future analysis needs based on the terms and variables established by the company's board of directors.

Finally, in terms of the project plan:

1. The project will be structured around the Cross Industry Standard Process for Data Mining (CRISP-DM) model. This model encompasses six sequential phases that form the basis for the data science process.

2. The “data understanding” (2), “data preparation” (3), “modeling” (3), and “evaluation” (4) phases will be carried out using the R programming language and the RStudio integrated development environment (IDE) with RMarkdown.

3. Given the nature of this work, the use of other collaborative tools or version control systems is not considered. However, under normal circumstances, tools like Git for version control and facilitating iteration between different project phases until reaching a final product could be employed. Additionally, depending on the team size, collaborative work tools such as Slack or Teams may be required.

4. Similarly, and in line with the previous point, a detailed plan with a schedule for each phase would be necessary.

Summary of tasks addressed in this phase:

- 1. Definition of the population: Countries with a population at or above the threshold of senescence, with a positive economic evolution.
- 2. Data collection: Due to the nature of the required data, primary data is unavailable as it pertains to “state” data not held by the companies. Therefore, we need to collect data from:
  - Secondary sources: Data collected by other institutions or derived from research studies.
  - Tertiary sources: Data collected from third parties. These sources do not directly come from the publisher but are reputable sources such as the United Nations (UN), World Health Organization (WHO), etc. These official sources compile data from other official sources, in this case, states or nations.

For this project, the data will originate from third-party sources as described above, which are official data collected by institutions with the social objective of conducting statistical work.

- 3. Identification of variables: In this phase, we have identified income, population density, age, infrastructure, resources, and population safety as key variables for the target population.

Phase 2: Understanding the Data

Objective of the phase: Identification, collection, and analysis of datasets that will help achieve the project objectives.

Introduction:

As established in the previous phase, the data required for this data science project is not held by the stakeholders or the company. We must necessarily rely on external sources to collect the data. Since the nature of the project involves comparing data from different countries, if it were a specific country, we could gather the data from the government’s official website. However, in our case, we need worldwide data, so we must necessarily turn to organizations that collect official data from third parties, such as the World Health Organization “Data at WHO” (n.d.) for international health aspects, or the United Nations United (n.d.) for the evolution of various international aspects.

Additionally, there are various companies and organizations that typically collect open data from institutions that, as mentioned earlier, do not directly collect the data but conduct research and work based on them. This could be the case with websites like “Statista” (“Statista - El portal de estadísticas” (n.d.)) or “Dataverse” (“The Dataverse Project - Dataverse.org” (2023)), to name a few. In data analysis and science, data that has not been collected firsthand or does not come from research or directly published works must be used with the utmost caution. Knowing the origin and all aspects related to the dataset is crucial, and tertiary sources of data always pose a risk.

With all this in mind, we have opted for a source that could be considered more than tertiary, even quaternary: the statistics service “Our World in Data” (“Our World in Data” (n.d.)). Behind this service is the non-profit organization “Global Change Data Lab” (“Global Change Data Lab” (n.d.)), which collaborates with the University of Oxford within “The Oxford Martin Programme” (“Global Development” (n.d.)). The management team and advisors are composed of prominent figures in academia and teaching. “Our World in Data” (OwD) collects data directly from official organizations, conducts various studies, and publishes works of diverse nature based on official data. They also provide extensive information, studies, and charts on the progress towards global sustainable development goals (“Measuring Progress Towards the Sustainable Development Goals - SDG Tracker” (n.d.)).

From an initial exploration, we have found that almost all of the variables we have identified as relevant for this work are presumably available in the service and openly accessible, except for some complementary datasets.

Therefore, now that the data source has been clarified, we will proceed with the milestones of this phase.

Collection of initial data.

To work with the variable “Income,” specifically the economic levels of different countries, we turn to the dataset “Annual growth of GDP per capita, 1961-2020” “Global Change Data Lab” (n.d.) . We are interested in theannual growth of GDP per capita for the past 10 years. [Source: World Development Indicators - World Bank (2022.05.26)]

Loading the dataset and name the table ‘income’:

```r
# Carga del conjunto de datos relativos a la renta
path = 'dataset/gdp-per-capita-growth.csv'
renta <- read.csv(path, row.names =NULL)
```

```r
str(renta)
```

```
## ’data.frame':    10463 obs. of  4 variables:
##  $ Entity                          : chr  "Afghanistan" "Afghanistan" "Afghanistan" "Afghanistan" ..
##  $ Code                            : chr  "AFG" "AFG" "AFG" "AFG" ...
##  $ Year                            : int  2003 2004 2005 2006 2007 2008 2009 2010 2011 2012 ...
##  $ GDP.per.capita.growth..annual...: num  3.87 -2.88 7.21 2.25 11.02 ...
```

We have 10,463 objects and 4 columns.

For the variable “Population,” we will work with datasets related to international population growth or decline. Therefore, we turn to the dataset “Population growth and demography” Roser (2013). We are interested in understanding the absolute and relative changes in population and segmenting the data by age groups. [Source: United Nations, Department of Economic and Social Affairs, Population Division (2022). World Population Prospects 2022, Online Edition.]

We load the dataset and name the table ‘population’:

```r
path= "dataset/population-and-demography.csv"
poblacion <- read.csv(path, row.names=NULL)
```

```
str(poblacion)
```

```
## 'data.frame':    18288 obs. of  24 variables:
##  $ Country.name                              : chr  "Afghanistan" "Afghanistan" "Afghanistan" "Afghan
##  $ Year                                      : int  1950 1951 1952 1953 1954 1955 1956 1957 1958 1959
##  $ Population                                : num  7480464 7571542 7667534 7764549 7864289 ...
##  $ Population.of.children.under.the.age.of.1 : num  301735 299368 305393 311574 317584 ...
##  $ Population.of.children.under.the.age.of.5 : int  1248282 1246857 1248220 1254725 1267817 1291129 1
##  $ Population.of.children.under.the.age.of.15: int  3068855 3105444 3145070 3186382 3231060 3281470 3
##  $ Population.under.the.age.of.25            : num  4494349 4552138 4613604 4676232 4741371 ...
##  $ Population.aged.15.to.64.years            : num  4198587 4250002 4303436 4356242 4408474 ...
##  $ Population.older.than.15.years            : num  4411609 4466098 4522464 4578167 4633229 ...
##  $ Population.older.than.18.years            : num  3946595 3993640 4041439 4088379 4136116 ...
##  $ Population.at.age.1                       : num  258652 254304 252906 258717 264765 ...
##  $ Population.aged.1.to.4.years              : num  946547 947489 942827 943151 950233 ...
##  $ Population.aged.5.to.9.years              : int  966210 991791 1017993 1039950 1055592 1062420 10
##  $ Population.aged.10.to.14.years            : int  854363 866796 878857 891707 907651 927921 951472
##  $ Population.aged.15.to.19.years            : int  757113 768616 781411 794308 806216 817550 828600
##  $ Population.aged.20.to.29.years            : int  1241348 1260904 1280288 1298803 1316768 1334989 1
##  $ Population.aged.30.to.39.years            : int  909953 922765 935638 948321 961484 975801 991166
##  $ Population.aged.40.to.49.years            : int  661807 667015 672491 678064 684153 691279 699431
##  $ Population.aged.50.to.59.years            : int  467170 468881 470898 472969 475117 477664 480665
##  $ Population.aged.60.to.69.years            : int  271905 273286 274852 276577 278210 279789 281376
##  $ Population.aged.70.to.79.years            : int  92691 94358 96026 97705 99298 100839 102385 1039
##  $ Population.aged.80.to.89.years            : int  9499 10155 10721 11254 11793 12342 12890 13428 13
##  $ Population.aged.90.to.99.years            : int  123 118 139 166 190 210 233 255 277 307 ...
##  $ Population.older.than.100.years           : num  0 0 0 0 0 0 0 0 0 0 ...
```

And as a result, it appears that the required variables for this work are present. There are numerous age segments, countries, etc. This table will require more attention and work. Regarding the variable “Infrastructure,” we have identified a suitable dataset called “Coverage of essential health services” (**goal3?**):, which captures the Universal Health Coverage (UHC) index globally. It is pertinent to note at this point that we are interested in understanding the infrastructure of each country, as it can be a good starting indicator. Constructing new facilities for the elderly without minimum healthcare conditions would be reckless, hence the relevance of this dataset. [Source: WHO, Global Health Observatory (2022)]

We verify the dataset and name the table ‘infra’:

```
path= "dataset/universal-health-coverage-index.csv"

infra <- read.csv(path, row.names = NULL)
```

```
str(infra)
```

```
## 'data.frame':    1248 obs. of  4 variables:
##  $ Entity                                       : chr  "Afghanistan" "Afghanistan" "Afghanistan"
##  $ Code                                         : chr  "AFG" "AFG" "AFG" "AFG" ...
##  $ Year                                         : int  2000 2005 2010 2015 2017 2019 2000 2005 20
##  $ Indicator.UHC.Service.Coverage.Index..SDG.3.8.1.: num  20.8 25.5 27.5 32.2 35.8 ...
```

And once again, we observe that the attributes we need for the project are present, primarily the UHC indicator.

Considering that the holding does not want to be involved in cases of human exploitation, as a socially responsible entity with values and objectives aligned with the achievement of sustainable development goals, we are interested in security. At this point, we will address the variable "Security," primarily focusing on legal security, specifically the degree of compliance with labor legislation. For this purpose, we have identified the dataset called "Compliance of labor rights." This dataset incorporates the International Labour Organization (ILO) index, which reflects respect for labor laws, including the rights to strike and association. [Source: International Labour Organization (ILO)]

We verify the dataset and rename the table as 'seguridad':

```r
path = "dataset/level-of-national-compliance-with-labor-rights.csv"

seguridad <- read.csv(path, row.names = NULL)
```

```r
str(seguridad)
```

```
## 'data.frame':    834 obs. of  4 variables:
##  $ Entity
##  $ Code
##  $ Year
##  $ X8.8.2...Level.of.national.compliance.with.labour.rights..freedom.of.association.and.collective.ba
```

We observe 264 objects and 4 variables, which correspond to our expectations, including the reference index.

Finally, regarding the variable "Recursos" (Resources), we will emphasize human resources with the appropriate training to support the holding's economic and social project as described earlier. We are interested in the dataset "Health Worker Density." Health worker density represents the size of qualified health personnel per 1,000 population. It is measured based on the density of physicians, surgeons, nurses and midwives, dentists, and pharmaceutical personnel. This variable ((**goal3?**):) is highly representative for the analytical objective of the project. Furthermore, given the nature of the company, having specialized personnel is crucial.

Verifying the dataset and rename the table as 'recursos':

```r
path = "dataset/physicians-per-1000-people.csv"

recursos <- read.csv(path, row.names =NULL)
```

```r
str(recursos)
```

```
## 'data.frame':    4686 obs. of  4 variables:
##  $ Entity                       : chr  "Afghanistan" "Afghanistan" "Afghanistan" "Afghanistan" ...
##  $ Code                         : chr  "AFG" "AFG" "AFG" "AFG" ...
##  $ Year                         : int  1960 1965 1970 1981 1986 1987 1989 1990 1993 1997 ...
##  $ Physicians..per.1.000.people.: num  0.035 0.063 0.065 0.077 0.183 ...
```

We observe that the table is error-free and contains the necessary attributes for this work.

On the other hand, although the variables and collected data seem to cover all the planned needs in the study, we believe it is necessary to implement a section for "complementary data" that can help us enhance the dimensions of analysis, especially for the supervised methodology, to further "feed" the model.

We believe that the variable "seguridad" (security) is not perfectly represented with data on respect for labor rights. Therefore, we will complement it with additional necessary data, using the table "Business

Confidence Index (BCI)” available on the OECD website (“Services Trade Restrictiveness Index | OECD Statistics on International Trade in Services | OECD iLibrary” (n.d.)).

This business confidence indicator provides information on future trends based on opinion surveys about production, orders, and finished product inventories in the industrial sector. It can be used to monitor production growth and anticipate turning points in economic activity. Figures above 100 suggest an increase in confidence regarding future business outcomes, while figures below 100 indicate pessimism about future results (“Leading Indicators - Business Confidence Index (BCI) - OECD Data” (n.d.)).

This table provides a powerful indicator that can help improve the perception of overall security in the target country.

Loading the table and rename it as ‘comp_seguridad’ for “security complements”:

```
path= "dataset/DP_LIVE_11052023100328461.csv"

comp_seguridad <- read.csv(path, row.names = NULL)
```

```
str(comp_seguridad)
```

```
## ’data.frame':    22927 obs. of  8 variables:
##  $ LOCATION  : chr  "ZAF" "ZAF" "ZAF" "ZAF" ...
##  $ INDICATOR : chr  "BCI" "BCI" "BCI" "BCI" ...
##  $ SUBJECT   : chr  "AMPLITUD" "AMPLITUD" "AMPLITUD" "AMPLITUD" ...
##  $ MEASURE   : chr  "LTRENDIDX" "LTRENDIDX" "LTRENDIDX" "LTRENDIDX" ...
##  $ FREQUENCY : chr  "M" "M" "M" "M" ...
##  $ TIME      : chr  "1974-06" "1974-07" "1974-08" "1974-09" ...
##  $ Value     : num  102 102 102 102 102 ...
##  $ Flag.Codes: logi  NA NA NA NA NA NA ...
```

And from the results, we obtain 22,927 objects. We observe that we have country codes and BCI values. The column names or variables differ from the bulk of the tables, but we can use ‘LOCATION’ as the primary or foreign key. The dates need to be modified.

In the same vein, we want to complement the economic information by including a table that captures public spending in relation to GDP and see the effects it has on the country’s long-term economic evolution and indicators.

For this purpose, we will use the table “Government spending in early-industrialized countries grew remarkably during the last century” Ortiz-Ospina and Roser (2016). This table captures the public spending of states with a very extensive time dimension, allowing us to observe how it has evolved in relation to well-being, statistics, and indices - social, economic, etc. [Source: Mauro, P., Romeu, R., Binder, A., & Zaman, A. (2015). A modern history of fiscal prudence and profligacy. Journal of Monetary Economics, 76, 55-70).

Loading the table and rename it as ‘comp_renta’ for “income complements”:

```
path= "dataset/total-gov-expenditure-gdp-wdi.csv"

comp_renta <- read.csv(path, row.names = NULL)
```

```
str(comp_renta)
```

```
## ’data.frame':    4039 obs. of  4 variables:
##  $ Entity            : chr  "Afghanistan" "Afghanistan" "Afghanistan" "Afghanistan" ...
##  $ Code              : chr  "AFG" "AFG" "AFG" "AFG" ...
##  $ Year              : int  2006 2007 2008 2009 2010 2011 2012 2013 2014 2015 ...
##  $ Expense....of.GDP.: num  20.6 24.2 50.7 44.3 50.9 ...
```

We observe 4,039 objects with 4 typical variables, including a unique variable that captures the IMF or reference index. The name of the fourth column, as is customary in datasets obtained from the “Our World in Data” web service, has a non-standard format.

We conclude this section by summarizing the dataset with which we will work after aggregating them into a single dataset:

Table 1: Summary table of variables and data nature

Variable	Data	Notes
Infrastructure	Coverage of essential health services	From the Universal Health Coverage dataset
Population	Population growth and demography	From the GDP per capita growth dataset
Resources	Density of qualified health personnel in the population	From the physicians per 1000 people dataset
Income	Per capita income levels	From the GDP per capita growth dataset
Security	Respect for labor rights	From the level of national compliance with labour rights dataset
Complementary (comp_security)	Business Confidence Index (BCI)	From the Business Confidence Index dataset
Complementary (comp_income)	Public expenditure index	From the “A modern history of fiscal prudence and profligacy. Journal of Monetary Economics dataset”

Data Description.

In the ‘income’ table for the Income variable, we observe that we have 10463 instances and 4 variables, which we will describe below:

ENTITY: Refers to the countries for which the reference index (GDP) has been calculated.

CODE: Country code, acts as the primary key.

GDP.per.capita.growth.annual: Corresponds to the numeric value of GDP. It represents the Gross Domestic Product (GDP) per capita in constant local currency. This is the aspect we aim to study.

YEAR: Corresponds to the temporal dimension.

In the ‘population’ table for the Population variable, we find 18288 objects and 24 variables.

COUNTRY.NAME: The names of the countries. There is no country code that can act as a primary or foreign key.

POPULATION: The total number of individuals in the population.

POPULATION.[. . .]: Different segmentations of the population based on age groups.

YEAR: The temporal dimension of the data.

In the ‘infra’ table for the Infrastructure variable, we find 1248 objects and 4 variables, which we will break down below:

ENTITY: Refers to the countries for which the reference index regarding basic health coverage has been calculated.

CODE: Key that identifies the countries.

YEAR: Temporal dimension of the data.

INDICATOR.UHC [...]: Corresponds to the values assigned to the UHC index.

In the 'resources' table for the Resources variable, we find 4686 instances of objects and 4 variables:

ENTITY: Refers to the countries for which the reference index regarding medical personnel per thousand people has been calculated.

CODE: Key that identifies the countries.

YEAR: Temporal dimension of the data.

PHYSICIANS..PER.1000.PEOPLE: Index of medical personnel density per 1000 individuals, calculated using the formula:

(medical_staff / total_population) * 1000

In the 'security' table for the Security variable, we observe exactly the same attributes described above, with the exception of PHYSICIANS..PER.1000.PEOPLE, which is replaced by:

X8.8.2... LEVEL.OF [...]: Corresponds to the previously described ILO index.

In the 'comp_security' table, we find the BCI index.

VALUE: Corresponds to the BCI.

In the 'comp_income' table, we find the index on government spending.

Expense...of.GDP: Corresponds to the portion of GDP dedicated to government expenditure.

Data Exploration.

A deeper analysis of the tables involves visualizing the data itself, establishing the relationships between them, and gaining insights into their analytical value. Additionally, we will standardize and clean the column names of the data frames, as we have already identified issues such as excessively long names, non-alphanumeric characters, and other problems. If we do not perform this task before the actual cleaning, we may encounter issues when applying functions like **`summary()`**, as the format of these variables may not be compatible.

At this point, we will use the **`clean_names()`** function from the **`janitor`** library to clean and standardize the column names Firke et al. (2023). We will also use the **`glimpse()`** function from the **`dplyr`** library, which is part of the **`tidyverse`** package Wickham and RStudio (2023). It provides a more concise view of the data structure compared to **`summary()`**, including the number of observations, variable names, and data types.

We will work on each table individually:

Table 'comp_income':

```r
# visualizamos los primeros registros del dataset comp_renta

head(comp_renta)
```

```
##        Entity Code Year Expense....of.GDP.
## 1 Afghanistan  AFG 2006           20.57817
## 2 Afghanistan  AFG 2007           24.24326
## 3 Afghanistan  AFG 2008           50.71930
## 4 Afghanistan  AFG 2009           44.31784
## 5 Afghanistan  AFG 2010           50.86300
## 6 Afghanistan  AFG 2011           59.48478
```

As mentioned earlier, the name of the fourth column does not seem appropriate. We will rename it using **rename()** and proceed with **clean_names()** to ensure that the variables have proper names:

```
# Aplicamos 'rename()':
comp_renta_clean <- comp_renta %>% rename(imf_index = Expense....of.GDP.)

# Aplicamos 'clean_names'
comp_renta_clean <- comp_renta_clean %>% clean_names()

# visualizamos el resultado
names(comp_renta_clean)
```

```
## [1] "entity"    "code"      "year"      "imf_index"
```

```
# Aplicamos 'glimpse()'

glimpse(comp_renta_clean)
```

```
## Rows: 4,039
## Columns: 4
## $ entity    <chr> "Afghanistan", "Afghanistan", "Afghanistan", "Afghanistan", ~
## $ code      <chr> "AFG", "AFG", "AFG", "AFG", "AFG", "AFG", "AFG", "AFG", "AFG~
## $ year      <int> 2006, 2007, 2008, 2009, 2010, 2011, 2012, 2013, 2014, 2015, ~
## $ imf_index <dbl> 20.57817, 24.24326, 50.71930, 44.31784, 50.86300, 59.48478, ~
```

We have the variables and the right format; we continue with the following table:

TABLE ‘comp_security’:

```
# visualizamos los primeros registros del dataset comp_seguridad

head(comp_seguridad)
```

```
##   LOCATION INDICATOR  SUBJECT   MEASURE FREQUENCY    TIME    Value Flag.Codes
## 1      ZAF       BCI AMPLITUD LTRENDIDX         M 1974-06 102.2216         NA
## 2      ZAF       BCI AMPLITUD LTRENDIDX         M 1974-07 102.0966         NA
## 3      ZAF       BCI AMPLITUD LTRENDIDX         M 1974-08 101.9844         NA
## 4      ZAF       BCI AMPLITUD LTRENDIDX         M 1974-09 101.9051         NA
## 5      ZAF       BCI AMPLITUD LTRENDIDX         M 1974-10 101.8678         NA
## 6      ZAF       BCI AMPLITUD LTRENDIDX         M 1974-11 101.8198         NA
```

The variable types and their value columns appear to have non-standard names, and we have unnecessary variables or those with no records that do not fully match the other tables. We will rename them using **rename()** and proceed with **clean_names()** to ensure that the variables have appropriate names in the standard format:

```
# Aplicamos 'rename()' al indice:
comp_segur_clean <- comp_seguridad %>% rename(bci_index = Value)

# Aplicamos 'rename()' a location
comp_segur_clean <- comp_segur_clean %>% rename(entity = LOCATION)
```

```
# Aplicamos 'clean_names'
comp_segur_clean <- comp_segur_clean %>% clean_names()

# visualizamos el resultado
names(comp_segur_clean)
```

```
## [1] "entity"     "indicator"  "subject"    "measure"    "frequency"
## [6] "time"       "bci_index"  "flag_codes"
```

```
# Aplicamos 'glimpse()'

glimpse(comp_segur_clean)
```

```
## Rows: 22,927
## Columns: 8
## $ entity     <chr> "ZAF", "ZAF", "ZAF", "ZAF", "ZAF", "ZAF", "ZAF", "ZAF", "ZA~
## $ indicator  <chr> "BCI", "BCI", "BCI", "BCI", "BCI", "BCI", "BCI", "BCI", "BC~
## $ subject    <chr> "AMPLITUD", "AMPLITUD", "AMPLITUD", "AMPLITUD", "AMPLITUD",~
## $ measure    <chr> "LTRENDIDX", "LTRENDIDX", "LTRENDIDX", "LTRENDIDX", "LTREND~
## $ frequency  <chr> "M", "M", "M", "M", "M", "M", "M", "M", "M", "M", "M", "M",~
## $ time       <chr> "1974-06", "1974-07", "1974-08", "1974-09", "1974-10", "197~
## $ bci_index  <dbl> 102.22160, 102.09660, 101.98440, 101.90510, 101.86780, 101.~
## $ flag_codes <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
```

Once corrected and ensured of errors, we observe that apart from the variable of interest, 'bci_index', we have unnecessary columns and an inappropriate date format that we will note for rectification in the next phase.

TABLE 'infra':

We visualize the first records of the 'infra' table:

```
head(infra)
```

```
##        Entity Code Year Indicator.UHC.Service.Coverage.Index..SDG.3.8.1.
## 1 Afghanistan  AFG 2000                                         20.75875
## 2 Afghanistan  AFG 2005                                         25.52582
## 3 Afghanistan  AFG 2010                                         27.51316
## 4 Afghanistan  AFG 2015                                         32.19185
## 5 Afghanistan  AFG 2017                                         35.78093
## 6 Afghanistan  AFG 2019                                         37.32812
```

The pattern of other tables is repeated: Four variables, three of them common. A fourth one is unique, with an exceptionally long name. We use the same procedure:

```
# Aplicamos 'rename()':
infra_clean <- infra %>% rename(uhc_index = Indicator.UHC.Service.Coverage.Index..SDG.3.8.1.)

# Aplicamos 'clean_names'
infra_clean <- infra_clean %>% clean_names()

# visualizamos el resultado
names(infra_clean)
```

```
## [1] "entity"    "code"      "year"      "uhc_index"
```

```r
# Aplicamos 'glimpse()' a la tabla 'infra'

glimpse(infra_clean)
```

```
## Rows: 1,248
## Columns: 4
## $ entity    <chr> "Afghanistan", "Afghanistan", "Afghanistan", "Afghanistan", ~
## $ code      <chr> "AFG", "AFG", "AFG", "AFG", "AFG", "AFG", "", "", "", "", ""~
## $ year      <int> 2000, 2005, 2010, 2015, 2017, 2019, 2000, 2005, 2010, 2015, ~
## $ uhc_index <dbl> 20.75875, 25.52582, 27.51316, 32.19185, 35.78093, 37.32812, ~
```

Corrected and ensured of errors, we observe that indeed we have 4 variables (text strings, decimals, integers) compatible with what we have visualized in other tables: an entity name, a code acting as a key, a temporal dimension. Also, as the unique record of this table on which we will work, we have the UHC index on basic health coverage, previously described.

TABLE ‘poblacion’: We visualize the first records of the ‘poblacion’ table:

In this table we have twenty-four variables with a different format from the other tables such as country name and code. It also has the same variable of the total population specified in age intervals which is very convenient for the purposes of this study. We proceed with the arrangements of the column names.

```r
# Copiamos la tabla
poblacion_clean <- poblacion

# Cambiamos nombre de country_name a entity
poblacion_clean <- poblacion_clean %>% rename(entity = Country.name)

# Aplicamos 'rename()' para edad menos de 1:
poblacion_clean <- poblacion_clean %>% rename(u_1 = Population.of.children.under.the.age.of.1)

# Aplicamos 'rename()' para edad menos de 5:
poblacion_clean <- poblacion_clean %>% rename(u_5 = Population.of.children.under.the.age.of.5)

# Aplicamos 'rename()' para edad menos de 15:
poblacion_clean <- poblacion_clean %>% rename(u_15 = Population.of.children.under.the.age.of.15)

# Aplicamos 'rename()' para edad menos de 25:
poblacion_clean <- poblacion_clean %>% rename(u_25 = Population.under.the.age.of.25)

# Aplicamos 'rename()' para edad de entre 15 y 64:
poblacion_clean <- poblacion_clean %>% rename(btn_15_64 = Population.aged.15.to.64.years)

# Aplicamos 'rename()' para edad mayores de 15:
poblacion_clean <- poblacion_clean %>% rename(old_15 = Population.older.than.15.years)

# Aplicamos 'rename()' para edad mayores de 18:
poblacion_clean <- poblacion_clean %>% rename(old_18 = Population.older.than.18.years)

# Aplicamos 'rename()' para 1 anyo:
poblacion_clean <- poblacion_clean %>% rename(at_1 = Population.at.age.1)
```

```
# Aplicamos 'rename()' para edad entre 1 y 4:
poblacion_clean <- poblacion_clean %>% rename(btn_1_4 = Population.aged.1.to.4.years)

# Aplicamos 'rename()' para edad de entre 5 y 9:
poblacion_clean <- poblacion_clean %>% rename(btn_5_9 = Population.aged.5.to.9.years)

# Aplicamos 'rename()' para edad de entre 10 y 14:
poblacion_clean <- poblacion_clean %>% rename(btn_10_14 = Population.aged.10.to.14.years)

# Aplicamos 'rename()' para edad de entre 15 y 19:
poblacion_clean <- poblacion_clean %>% rename(btn_15_19 = Population.aged.15.to.19.years)

# Aplicamos 'rename()' para edad de entre 20 y 29:
poblacion_clean <- poblacion_clean %>% rename(btn_20_29 = Population.aged.20.to.29.years)

# Aplicamos 'rename()' para edad de entre 30 y 39:
poblacion_clean <- poblacion_clean %>% rename(btn_30_39 = Population.aged.30.to.39.years)

# Aplicamos 'rename()' para edad de entre 40 y 49:
poblacion_clean <- poblacion_clean %>% rename(btn_40_49 = Population.aged.40.to.49.years)

# Aplicamos 'rename()' para edad de entre 50 y 59:
poblacion_clean <- poblacion_clean %>% rename(btn_50_59 = Population.aged.50.to.59.years)

# Aplicamos 'rename()' para edad de entre 60 y 69:
poblacion_clean <- poblacion_clean %>% rename(btn_60_69 = Population.aged.60.to.69.years)

# Aplicamos 'rename()' para edad de entre 70 y 79:
poblacion_clean <- poblacion_clean %>% rename(btn_70_79 = Population.aged.70.to.79.years)

# Aplicamos 'rename()' para edad de entre 80 y 89:
poblacion_clean <- poblacion_clean %>% rename(btn_80_89 = Population.aged.80.to.89.years)

# Aplicamos 'rename()' para edad de entre 90 y 99:
poblacion_clean <- poblacion_clean %>% rename(btn_90_99 = Population.aged.90.to.99.years)

# Aplicamos 'rename()' para mayores de 100:
poblacion_clean <- poblacion_clean %>% rename(old_100 = Population.older.than.100.years)

# Aplicamos 'clean_names'
poblacion_clean <- poblacion_clean %>% clean_names()

# visualizamos el resultado
names(poblacion_clean)
```

```
##  [1] "entity"     "year"       "population" "u_1"        "u_5"
##  [6] "u_15"       "u_25"       "btn_15_64"  "old_15"     "old_18"
## [11] "at_1"       "btn_1_4"    "btn_5_9"    "btn_10_14"  "btn_15_19"
## [16] "btn_20_29"  "btn_30_39"  "btn_40_49"  "btn_50_59"  "btn_60_69"
## [21] "btn_70_79"  "btn_80_89"  "btn_90_99"  "old_100"
```

After correcting the fields, we observe that we indeed have all the variable names and intervals, and they have been appropriately renamed and ensured the format. We notice that it lacks an identifying entity code

‘code’. We will decide on this aspect later when we unify and create a table to work with.

We visualize the table statistics with glimpse():

```
# Aplicamos 'glimpse()' a la tabla 'infra'

glimpse(poblacion_clean)
```

```
## Rows: 18,288
## Columns: 24
## $ entity     <chr> "Afghanistan", "Afghanistan", "Afghanistan", "Afghanistan",~
## $ year       <int> 1950, 1951, 1952, 1953, 1954, 1955, 1956, 1957, 1958, 1959,~
## $ population <dbl> 7480464, 7571542, 7667534, 7764549, 7864289, 7971933, 80877~
## $ u_1        <dbl> 301735, 299368, 305393, 311574, 317584, 323910, 330888, 337~
## $ u_5        <int> 1248282, 1246857, 1248220, 1254725, 1267817, 1291129, 13223~
## $ u_15       <int> 3068855, 3105444, 3145070, 3186382, 3231060, 3281470, 33370~
## $ u_25       <dbl> 4494349, 4552138, 4613604, 4676232, 4741371, 4812348, 48892~
## $ btn_15_64  <dbl> 4198587, 4250002, 4303436, 4356242, 4408474, 4462830, 45200~
## $ old_15     <dbl> 4411609, 4466098, 4522464, 4578167, 4633229, 4690463, 47507~
## $ old_18     <dbl> 3946595, 3993640, 4041439, 4088379, 4136116, 4187921, 42430~
## $ at_1       <dbl> 258652, 254304, 252906, 258717, 264765, 270904, 277351, 284~
## $ btn_1_4    <dbl> 946547, 947489, 942827, 943151, 950233, 967219, 991454, 101~
## $ btn_5_9    <int> 966210, 991791, 1017993, 1039950, 1055592, 1062420, 1063212~
## $ btn_10_14  <int> 854363, 866796, 878857, 891707, 907651, 927921, 951472, 975~
## $ btn_15_19  <int> 757113, 768616, 781411, 794308, 806216, 817550, 828600, 839~
## $ btn_20_29  <int> 1241348, 1260904, 1280288, 1298803, 1316768, 1334989, 13539~
## $ btn_30_39  <int> 909953, 922765, 935638, 948321, 961484, 975801, 991166, 100~
## $ btn_40_49  <int> 661807, 667015, 672491, 678064, 684153, 691279, 699431, 708~
## $ btn_50_59  <int> 467170, 468881, 470898, 472969, 475117, 477664, 480665, 484~
## $ btn_60_69  <int> 271905, 273286, 274852, 276577, 278210, 279789, 281376, 282~
## $ btn_70_79  <int> 92691, 94358, 96026, 97705, 99298, 100839, 102385, 103932, ~
## $ btn_80_89  <int> 9499, 10155, 10721, 11254, 11793, 12342, 12890, 13428, 1395~
## $ btn_90_99  <int> 123, 118, 139, 166, 190, 210, 233, 255, 277, 307, 341, 372,~
## $ old_100    <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1,~
```

We observe that some fields are of type floating point <dbl>, which does not make sense when counting people. Therefore, this table requires an additional operation to adjust the <dbl> fields and convert them to integers <int>.

```
# cambiamos el tipo de campos con un condicional 'if':

poblacion_clean <- poblacion_clean %>%
  mutate_if(is.double, as.integer)

# Y volvemos a revisar el resultado con glimpse:

glimpse(poblacion_clean)
```

```
## Rows: 18,288
## Columns: 24
## $ entity     <chr> "Afghanistan", "Afghanistan", "Afghanistan", "Afghanistan",~
## $ year       <int> 1950, 1951, 1952, 1953, 1954, 1955, 1956, 1957, 1958, 1959,~
## $ population <int> 7480464, 7571542, 7667534, 7764549, 7864289, 7971933, 80877~
## $ u_1        <int> 301735, 299368, 305393, 311574, 317584, 323910, 330888, 337~
## $ u_5        <int> 1248282, 1246857, 1248220, 1254725, 1267817, 1291129, 13223~
## $ u_15       <int> 3068855, 3105444, 3145070, 3186382, 3231060, 3281470, 33370~
## $ u_25       <int> 4494349, 4552138, 4613604, 4676232, 4741371, 4812348, 48892~
## $ btn_15_64  <int> 4198587, 4250002, 4303436, 4356242, 4408474, 4462830, 45200~
## $ old_15     <int> 4411609, 4466098, 4522464, 4578167, 4633229, 4690463, 47507~
## $ old_18     <int> 3946595, 3993640, 4041439, 4088379, 4136116, 4187921, 42430~
## $ at_1       <int> 258652, 254304, 252906, 258717, 264765, 270904, 277351, 284~
## $ btn_1_4    <int> 946547, 947489, 942827, 943151, 950233, 967219, 991454, 101~
## $ btn_5_9    <int> 966210, 991791, 1017993, 1039950, 1055592, 1062420, 1063212~
## $ btn_10_14  <int> 854363, 866796, 878857, 891707, 907651, 927921, 951472, 975~
## $ btn_15_19  <int> 757113, 768616, 781411, 794308, 806216, 817550, 828600, 839~
## $ btn_20_29  <int> 1241348, 1260904, 1280288, 1298803, 1316768, 1334989, 13539~
## $ btn_30_39  <int> 909953, 922765, 935638, 948321, 961484, 975801, 991166, 100~
## $ btn_40_49  <int> 661807, 667015, 672491, 678064, 684153, 691279, 699431, 708~
## $ btn_50_59  <int> 467170, 468881, 470898, 472969, 475117, 477664, 480665, 484~
## $ btn_60_69  <int> 271905, 273286, 274852, 276577, 278210, 279789, 281376, 282~
## $ btn_70_79  <int> 92691, 94358, 96026, 97705, 99298, 100839, 102385, 103932, ~
## $ btn_80_89  <int> 9499, 10155, 10721, 11254, 11793, 12342, 12890, 13428, 1395~
## $ btn_90_99  <int> 123, 118, 139, 166, 190, 210, 233, 255, 277, 307, 341, 372,~
## $ old_100    <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1,~
```

And now it looks like we have all the attributes in the right format. Let's proceed with the next table.

TABLE 'resources':

```r
head(recursos)
```

```
##        Entity Code Year Physicians..per.1.000.people.
## 1 Afghanistan  AFG 1960                         0.035
## 2 Afghanistan  AFG 1965                         0.063
## 3 Afghanistan  AFG 1970                         0.065
## 4 Afghanistan  AFG 1981                         0.077
## 5 Afghanistan  AFG 1986                         0.183
## 6 Afghanistan  AFG 1987                         0.179
```

This table respects the 4-column pattern we have seen in other tables. We have three columns common to the dataset and one that collects the object of study in this project, the ratio of medical personnel per 1000 people. Proceed with the cleaning and renaming:

```r
# Aplicamos 'rename()':
recursos_clean <- recursos %>% rename(phy_index = Physicians..per.1.000.people.)

# Aplicamos 'clean_names'
recursos_clean <- recursos_clean %>% clean_names()

# visualizamos el resultado
names(recursos_clean)
```

```
## [1] "entity"    "code"      "year"      "phy_index"
```

```r
# Aplicamos 'glimpse()' a la tabla 'recursos'

glimpse(recursos_clean)
```

```
## Rows: 4,686
## Columns: 4
## $ entity    <chr> "Afghanistan", "Afghanistan", "Afghanistan", "Afghanistan", ~
## $ code      <chr> "AFG", "AFG", "AFG", "AFG", "AFG", "AFG", "AFG", "AFG", "AFG~
## $ year      <int> 1960, 1965, 1970, 1981, 1986, 1987, 1989, 1990, 1993, 1997, ~
## $ phy_index <dbl> 0.0350, 0.0630, 0.0650, 0.0770, 0.1830, 0.1790, 0.1290, 0.10~
```

Format and the variables are correct. phy_index' is the variable that we will proceed to study in the next phases of this project.

TABLE 'income':

```r
#Exploramos la tabla renta

head(renta)
```

```
##        Entity Code Year GDP.per.capita.growth..annual...
## 1 Afghanistan  AFG 2003                         3.868380
## 2 Afghanistan  AFG 2004                        -2.875203
## 3 Afghanistan  AFG 2005                         7.207967
## 4 Afghanistan  AFG 2006                         2.253311
## 5 Afghanistan  AFG 2007                        11.022787
## 6 Afghanistan  AFG 2008                         1.594216
```

This table also respects the 4-column pattern we have seen in other tables. Have three columns common to the and one that collects the object of study in this project, the per capita income. We now proceed with the cleaning and renaming:

```r
# Aplicamos 'rename()':
renta_clean <- renta %>% rename(gdp_index = GDP.per.capita.growth..annual...)

# Aplicamos 'clean_names'
renta_clean <- renta_clean %>% clean_names()

# visualizamos el resultado
names(renta_clean)
```

```
## [1] "entity"    "code"      "year"      "gdp_index"
```

```r
# Aplicamos 'glimpse()' a la tabla 'renta'

glimpse(renta_clean)
```

```
## Rows: 10,463
## Columns: 4
## $ entity    <chr> "Afghanistan", "Afghanistan", "Afghanistan", "Afghanistan", ~
## $ code      <chr> "AFG", "AFG", "AFG", "AFG", "AFG", "AFG", "AFG", "AFG", "AFG~
## $ year      <int> 2003, 2004, 2005, 2006, 2007, 2008, 2009, 2010, 2011, 2012, ~
## $ gdp_index <dbl> 3.86838031, -2.87520313, 7.20796728, 2.25331068, 11.02278709~
```

We observe that the column type and format are appropriate for the variable of interest. We proceed to the next table:

TABLE 'seguridad':

```
#Exploramos la tabla seguridad

head(seguridad)
```

```
##    Entity Code Year
## 1 Algeria  DZA 2016
## 2 Algeria  DZA 2017
## 3 Algeria  DZA 2018
## 4 Algeria  DZA 2019
## 5 Algeria  DZA 2020
## 6  Angola  AGO 2018
##   X8.8.2...Level.of.national.compliance.with.labour.rights..freedom.of.association.and.collective.ba
## 1
## 2
## 3
## 4
## 5
## 6
```

We observe the same pattern as mentioned earlier. Three common variables and one variable for the study, the index of compliance with labor regulations, which is represented by a column with an excessively long name. We proceed with renaming and cleaning:

```
# Aplicamos 'rename()':
seguridad_clean <- seguridad %>% rename(ilo_index = X8.8.2...Level.of.national.compliance.with.labour.r

# Aplicamos 'clean_names'
seguridad_clean <- seguridad_clean %>% clean_names()

# visualizamos el resultado
names(seguridad_clean)
```

```
## [1] "entity"    "code"      "year"      "ilo_index"
```

```
# Aplicamos 'glimpse()' a la tabla 'seguridad'

glimpse(seguridad_clean)
```

```
## Rows: 834
## Columns: 4
## $ entity    <chr> "Algeria", "Algeria", "Algeria", "Algeria", "Algeria", "Ango~
## $ code      <chr> "DZA", "DZA", "DZA", "DZA", "DZA", "AGO", "AGO", "AGO", "ATG~
## $ year      <int> 2016, 2017, 2018, 2019, 2020, 2018, 2019, 2020, 2015, 2016, ~
## $ ilo_index <dbl> 1.90, 2.11, 2.30, 2.48, 2.48, 2.27, 2.27, 2.27, 0.72, 0.72, ~
```

Not observing any problems in the table, we proceed to verify the quality of the data.

Checking data quality

At this point, we are going to look for infinite or NA values, as well as other issues such as blank records, and review other aspects of data quality that may be pending:

TABLE “comp_renta_clean”

Searching for records with ‘NA’ or blank values:

```
# Buscamos problemas en la tabla:
print("NA")
```

```
## [1] "NA"
```

```
colSums(is.na(comp_renta_clean))
```

```
##    entity      code      year imf_index 
##         0         0         0         0
```

```
print("Blancos")
```

```
## [1] "Blancos"
```

```
colSums(comp_renta_clean=="")
```

```
##    entity      code      year imf_index 
##         0       262         0         0
```

We did not find any infinite or blank values except in the ‘code’ column. We will continue with the next table.

TABLE “comp_segur_clean”

Searching for records with ‘NA’ or blank values:

```
# Buscamos problemas en la tabla:
print("NA")
```

```
## [1] "NA"
```

```
colSums(is.na(comp_segur_clean))
```

```
##     entity  indicator    subject    measure  frequency       time  bci_index 
##          0          0          0          0          0          0          0 
## flag_codes 
##      22927
```

```
print("Blancos")
```

```
## [1] "Blancos"
```

```r
colSums(comp_segur_clean=="")
```

```
##     entity  indicator    subject    measure  frequency       time  bci_index
##          0          0          0          0          0          0          0
## flag_codes
##         NA
```

flag_codes’ registers problems, but it is a column that we note to remove in the next phase:

TABLE “infra_clean”.

We look for records with ‘NA’ or blank values:

```r
# Buscamos problemas en la tabla:
print("NA")
```

```
## [1] "NA"
```

```r
colSums(is.na(infra_clean))
```

```
##    entity      code      year uhc_index
##         0         0         0         0
```

```r
print("Blancos")
```

```
## [1] "Blancos"
```

```r
colSums(infra_clean=="")
```

```
##    entity      code      year uhc_index
##         0        78         0         0
```

We detected that in the column ‘code’ there are blank records, so we take note of them to work with them in the next phase, where we will undertake the cleaning.

TABLE “population_clean”.

We look for records with ‘NA’ or blank values:

```r
# Buscamos problemas en la tabla(I):
print("NA")
```

```
## [1] "NA"
```

```r
colSums(is.na(poblacion_clean))
```

```
##     entity       year population        u_1        u_5       u_15       u_25
##          0          0        336          0          0          0        126
##  btn_15_64     old_15     old_18       at_1    btn_1_4    btn_5_9  btn_10_14
##        172        193        166          0          0          0          0
##  btn_15_19  btn_20_29  btn_30_39  btn_40_49  btn_50_59  btn_60_69  btn_70_79
##          0          0          0          0          0          0          0
##  btn_80_89  btn_90_99    old_100
##          0          0          0
```

We have infinite values in some columns. We take note to work on them in the next phase.

Let's proceed to determine if there are any records with blank values, although we already know that the function won't sum correctly if there are 'NAs':

```
# Buscamos problemas en la tabla (II)
print("Blancos")
```

```
## [1] "Blancos"
```

```
colSums(poblacion_clean=="")
```

```
##     entity       year population        u_1        u_5       u_15       u_25
##          0          0         NA          0          0          0         NA
##  btn_15_64     old_15     old_18       at_1    btn_1_4    btn_5_9  btn_10_14
##         NA         NA         NA          0          0          0          0
##  btn_15_19  btn_20_29  btn_30_39  btn_40_49  btn_50_59  btn_60_69  btn_70_79
##          0          0          0          0          0          0          0
##  btn_80_89  btn_90_99    old_100
##          0          0          0
```

We observe that there are columns with 'NA' issues. Some variables, presumably, can be ignored as they do not belong to the age range of interest, but the 'NA' values in the 'population' column could be relevant. We take note to address these issues in the next phase.

Let's proceed to search for records with 'NA' or blank values in the "recursos_clean" table:

We have looked for records with 'NA' or blank values in the "recursos_clean" table and found that there are no such records.

```
# Buscamos problemas en la tabla:
print("NA")
```

```
## [1] "NA"
```

```
colSums(is.na(recursos_clean))
```

```
##    entity      code      year phy_index
##         0         0         0         0
```

```
print("Blancos")
```

```
## [1] "Blancos"
```

```
colSums(recursos_clean=="")
```

```
##    entity      code      year phy_index
##         0        53         0         0
```

We note that it presents the same problem as previously detected: the 'code' column has fifty-three blank records. We will address these problems in the next phase.

TABLE "renta_clean".

We look for records with 'NA' or blank values:

```
# Buscamos problemas en la tabla:
print("NA")
```

```
## [1] "NA"
```

```
colSums(is.na(renta_clean))
```

```
##    entity      code      year gdp_index
##         0         0         0         0
```

```
print("Blancos")
```

```
## [1] "Blancos"
```

```
colSums(renta_clean=="")
```

```
##    entity      code      year gdp_index
##         0       784         0         0
```

Once again, we note that there are 784 blank values in the ‘code’ column. We repeat the process of annotating to correct in the next phase.

TABLE “security_clean

We look for records with ‘NA’ or blank values:

```
# Buscamos problemas en la tabla:
print("NA")
```

```
## [1] "NA"
```

```
colSums(is.na(seguridad_clean))
```

```
##    entity      code      year ilo_index
##         0         0         0         0
```

```
print("Blancos")
```

```
## [1] "Blancos"
```

```
colSums(seguridad_clean=="")
```

```
##    entity      code      year ilo_index
##         0        60         0         0
```

The results are in line with expectations. We proceed to the next phase where we will address various issues and build the final unified dataset for applying different models.

Phase 3: Data Preparation

The first issue to tackle is the date format and temporal dimension. The different tables display data from different years. We have set the limit based on the lowest period recorded in the various tables, which in this case is 2015 from the ‘seguridad’ table. This will not prevent the occurrence of ‘NA’ values and other undesired effects when merging tables.

Rectifying the date format

The ‘comp_segur_clean’ table has a date format that is incompatible with the tables we are working with. We proceed to rectify it:

```r
# Rectificamos el formato de fecha
comp_segur_clean$time <- as.integer(substr(comp_segur_clean$time, 1, 4))

# Renombramos la columna
comp_segur_clean <- comp_segur_clean %>% rename(year = time)

# visualizamos los nombres de las columnas
head(comp_segur_clean)
```

```
##   entity indicator  subject   measure frequency year bci_index flag_codes
## 1    ZAF       BCI AMPLITUD LTRENDIDX         M 1974  102.2216         NA
## 2    ZAF       BCI AMPLITUD LTRENDIDX         M 1974  102.0966         NA
## 3    ZAF       BCI AMPLITUD LTRENDIDX         M 1974  101.9844         NA
## 4    ZAF       BCI AMPLITUD LTRENDIDX         M 1974  101.9051         NA
## 5    ZAF       BCI AMPLITUD LTRENDIDX         M 1974  101.8678         NA
## 6    ZAF       BCI AMPLITUD LTRENDIDX         M 1974  101.8198         NA
```

```r
# Visualizamos los nombres de variables
names(comp_segur_clean)
```

```
## [1] "entity"     "indicator"  "subject"    "measure"    "frequency"
## [6] "year"       "bci_index"  "flag_codes"
```

Reduction of the time dimension

```r
# Reducción de los conjuntos de datos a registros desde 2015 usando 'filter()'

comp_renta <- comp_renta_clean %>% filter(year >= 2015 & year <= 2020)
comp_seguridad <- comp_segur_clean %>% filter(year >= 2015 & year <= 2020)
infra <- infra_clean %>% filter(year >= 2015 & year <= 2020)
poblacion <- poblacion_clean %>% filter(year >= 2015 & year <= 2020)
recursos <- recursos_clean %>% filter(year >= 2015 & year <= 2020)
renta <- renta_clean %>% filter(year >= 2015 & year <= 2020)
seguridad <- seguridad_clean %>% filter(year >= 2015 & year <= 2020)

# Observamos las tablas con glimpse():
print("infra")
```

```
## [1] "infra"
```

```r
glimpse(infra)
```

```
## Rows: 624
## Columns: 4
## $ entity    <chr> "Afghanistan", "Afghanistan", "Afghanistan", "Africa", "Afri~
## $ code      <chr> "AFG", "AFG", "AFG", "", "", "", "ALB", "ALB", "ALB", "DZA",~
## $ year      <int> 2015, 2017, 2019, 2015, 2017, 2019, 2015, 2017, 2019, 2015, ~
## $ uhc_index <dbl> 32.19185, 35.78093, 37.32812, 45.35265, 46.84041, 48.70660, ~
```

```r
print("poblacion")
```

```
## [1] "poblacion"
```

```r
glimpse(poblacion)
```

```
## Rows: 1,524
## Columns: 24
## $ entity     <chr> "Afghanistan", "Afghanistan", "Afghanistan", "Afghanistan",~
## $ year       <int> 2015, 2016, 2017, 2018, 2019, 2020, 2015, 2016, 2017, 2018,~
## $ population <int> 33753500, 34636212, 35643420, 36686788, 37769496, 38972236,~
## $ u_1        <int> 1238201, 1256664, 1268641, 1290266, 1313684, 1338671, 40085~
## $ u_5        <int> 5747862, 5894636, 6028737, 6147355, 6262390, 6375097, 18929~
## $ u_15       <int> 15456437, 15766737, 16081817, 16402093, 16728620, 17072924,~
## $ u_25       <int> 22708012, 23239372, 23807538, 24377892, 24950588, 25563700,~
## $ btn_15_64  <int> 17485012, 18038552, 18706224, 19401608, 20127700, 20957368,~
## $ old_15     <int> 18296928, 18869402, 19561562, 20284670, 21040860, 21899280,~
## $ old_18     <int> 15818155, 16325871, 16945448, 17596500, 18284280, 19076408,~
## $ at_1       <int> 1187761, 1216903, 1238498, 1252219, 1274592, 1299717, 38746~
## $ btn_1_4    <int> 4509661, 4637972, 4760096, 4857089, 4948706, 5036426, 14920~
## $ btn_5_9    <int> 5083265, 5142552, 5212487, 5334330, 5487694, 5647052, 16465~
## $ btn_10_14  <int> 4625310, 4729549, 4840593, 4920408, 4978536, 5050775, 14154~
## $ btn_15_19  <int> 3998778, 4102738, 4222210, 4343114, 4461340, 4584023, 12264~
## $ btn_20_29  <int> 5732466, 5948136, 6206481, 6468908, 6738758, 7049695, 20427~
## $ btn_30_39  <int> 3529267, 3617735, 3741917, 3881454, 4039411, 4238417, 15003~
## $ btn_40_49  <int> 2350849, 2428714, 2517748, 2608343, 2701018, 2804197, 10039~
## $ btn_50_59  <int> 1413659, 1464517, 1523369, 1586312, 1653460, 1726474, 66087~
## $ btn_60_69  <int> 814056, 833677, 856795, 883719, 914688, 949111, 39015176, 4~
## $ btn_70_79  <int> 371848, 385801, 401322, 416827, 431683, 442845, 17785904, 1~
## $ btn_80_89  <int> 81378, 83944, 87484, 91481, 95602, 99060, 4936880, 5078636,~
## $ btn_90_99  <int> 4628, 4140, 4236, 4513, 4901, 5458, 434246, 455671, 479911,~
## $ old_100    <int> 133, 72, 40, 23, 14, 32, 9835, 10498, 11278, 12040, 12594, ~
```

```r
print("recursos")
```

```
## [1] "recursos"
```

```r
glimpse(recursos)
```

```
## Rows: 469
## Columns: 4
## $ entity    <chr> "Afghanistan", "Afghanistan", "Albania", "Albania", "Albania~
## $ code      <chr> "AFG", "AFG", "ALB", "ALB", "ALB", "DZA", "DZA", "AND", "AGO~
## $ year      <int> 2015, 2016, 2016, 2018, 2019, 2016, 2018, 2015, 2017, 2017, ~
## $ phy_index <dbl> 0.2850, 0.2782, 1.2164, 2.1584, 1.6471, 1.8325, 1.7193, 3.33~
```

```
print("renta")
```

```
## [1] "renta"
```

```
glimpse(renta)
```

```
## Rows: 1,312
## Columns: 4
## $ entity    <chr> "Afghanistan", "Afghanistan", "Afghanistan", "Afghanistan", ~
## $ code      <chr> "AFG", "AFG", "AFG", "AFG", "AFG", "AFG", "ALB", "ALB", "ALB~
## $ year      <int> 2015, 2016, 2017, 2018, 2019, 2020, 2015, 2016, 2017, 2018, ~
## $ gdp_index <dbl> -1.62285721, -0.54141617, 0.06476419, -1.19490039, 1.5356366~
```

```
print("seguridad")
```

```
## [1] "seguridad"
```

```
glimpse(seguridad)
```

```
## Rows: 834
## Columns: 4
## $ entity    <chr> "Algeria", "Algeria", "Algeria", "Algeria", "Algeria", "Ango~
## $ code      <chr> "DZA", "DZA", "DZA", "DZA", "DZA", "AGO", "AGO", "AGO", "ATG~
## $ year      <int> 2016, 2017, 2018, 2019, 2020, 2018, 2019, 2020, 2015, 2016, ~
## $ ilo_index <dbl> 1.90, 2.11, 2.30, 2.48, 2.48, 2.27, 2.27, 2.27, 0.72, 0.72, ~
```

```
# Eliminamos las tablas 'clean'
rm(comp_renta_clean, comp_segur_clean, infra_clean, poblacion_clean, recursos_clean, renta_clean, segur
```

Everything seems to be in order, the time dimension in the new variables has been reduced.

Joining the datasets

We perform a full join using the dplyr library, i.e. we join the tables based on a key, preserving all rows on both 'sides'. If a key is not present it will be completed with 'NA'.

```
# Unificamos las tablas con un 'full join':
full_pra <- full_join(comp_renta, infra, by = c("entity", "code", "year")) %>%
  full_join(poblacion, by = c("entity", "year")) %>%
  full_join(recursos, by = c("entity", "code", "year")) %>%
  full_join(renta, by = c("entity", "code", "year")) %>%
  full_join(seguridad, by = c("entity", "code", "year")) %>%
  full_join(comp_seguridad, by = c("entity", "year"))

# Visualizamos la nueva tabla:
# glimpse(full_pra)
# Comento para que la salida derive en un pdf de muchas paginas innesesariamente
```

Continue the cleaning process by reducing the segments or demographics that are not relevant to this study:

Column and variable cleaning.

```r
# reducimos columnas irrelevantes

full_pra <- select(full_pra, -c(frequency, flag_codes, measure, subject, indicator))
```

```r
# Visualizamos nuevamente la tabla para revisar los cambios

glimpse(full_pra)
```

```
## Rows: 5,627
## Columns: 31
## $ entity     <chr> "Afghanistan", "Afghanistan", "Afghanistan", "Albania", "Al~
## $ code       <chr> "AFG", "AFG", "AFG", "ALB", "ALB", "ALB", "ALB", "ALB", "AG~
## $ year       <int> 2015, 2016, 2017, 2015, 2016, 2017, 2018, 2019, 2015, 2016,~
## $ imf_index  <dbl> 37.02257, 43.92276, 39.22011, 24.44618, 23.95673, 23.86760,~
## $ uhc_index  <dbl> 32.19185, NA, 35.78093, 61.65859, NA, 61.72466, NA, 62.1624~
## $ population <int> 33753500, 34636212, 35643420, 2882482, 2881064, 2879361, 28~
## $ u_1        <int> 1238201, 1256664, 1268641, 33567, 32138, 30844, 29625, 2877~
## $ u_5        <int> 5747862, 5894636, 6028737, 167618, 165643, 162006, 156677, ~
## $ u_15       <int> 15456437, 15766737, 16081817, 523652, 512651, 502251, 49148~
## $ u_25       <int> 22708012, 23239372, 23807538, 1019225, 993922, 968245, 9418~
## $ btn_15_64  <int> 17485012, 18038552, 18706224, 1968730, 1965888, 1962446, 19~
## $ old_15     <int> 18296928, 18869402, 19561562, 2358651, 2368228, 2376917, 23~
## $ old_18     <int> 15818155, 16325871, 16945448, 2218868, 2234305, 2249333, 22~
## $ at_1       <int> 1187761, 1216903, 1238498, 34071, 32964, 31430, 30142, 2889~
## $ btn_1_4    <int> 4509661, 4637972, 4760096, 134051, 133505, 131162, 127052, ~
## $ btn_5_9    <int> 5083265, 5142552, 5212487, 160753, 159404, 159817, 160919, ~
## $ btn_10_14  <int> 4625310, 4729549, 4840593, 195281, 187604, 180428, 173888, ~
## $ btn_15_19  <int> 3998778, 4102738, 4222210, 239277, 228578, 218700, 210667, ~
## $ btn_20_29  <int> 5732466, 5948136, 6206481, 477068, 478945, 477974, 474297, ~
## $ btn_30_39  <int> 3529267, 3617735, 3741917, 340003, 344363, 350407, 357749, ~
## $ btn_40_49  <int> 2350849, 2428714, 2517748, 360196, 355067, 349087, 342580, ~
## $ btn_50_59  <int> 1413659, 1464517, 1523369, 394606, 396270, 397303, 397162, ~
## $ btn_60_69  <int> 814056, 833677, 856795, 284099, 294822, 305532, 316169, 326~
## $ btn_70_79  <int> 371848, 385801, 401322, 185620, 188767, 192299, 196515, 201~
## $ btn_80_89  <int> 81378, 83944, 87484, 68581, 71884, 75685, 79826, 84182, 779~
## $ btn_90_99  <int> 4628, 4140, 4236, 9201, 9532, 9930, 10365, 10829, 4901, 523~
## $ old_100    <int> 133, 72, 40, 179, 185, 193, 205, 218, 182, 184, 187, 187, 1~
## $ phy_index  <dbl> 0.2850, 0.2782, NA, NA, 1.2164, NA, 2.1584, 1.6471, NA, NA,~
## $ gdp_index  <dbl> -1.62285721, -0.54141617, 0.06476419, 2.51682711, 3.4802932~
## $ ilo_index  <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, 2.27, 2.27, 2.1~
## $ bci_index  <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
```

Demographics

The dataset contains numerous attributes where the population has been segmented into age groups. While we are interested in the number of individuals who are within 15-20 years of retirement - as they are potential clients for the company - we will keep all the columns to train different supervised models.

Country Codes and Entities

We will create variables with 3-letter codes according to ISO 3166-1 alpha-3 standard “ISO - ISO 3166 — Country Codes” (n.d.) order to modify the empty fields and those with infinite values.

```
# Codigos UE
eu_countries <- c("AUT", "BEL", "BGR", "HRV", "CYP", "CZE", "DNK", "EST", "FIN", "FRA", "DEU", "GRC", "
# Resto de Europa (no UE)
other_europe_countries <- c("ALB", "AND", "BIH", "BLR", "CHE", "GEO", "ISL", "LIE", "MDA", "MKD", "MNE"
# Asia
asia_countries <- c("AFG", "ARM", "AZE", "BHR", "BGD", "BTN", "BRN", "KHM", "CHN", "CYP", "GEO", "IND",
# América
america_countries <- c("ARG", "BHS", "BRB", "BLZ", "BOL", "BRA", "CHL", "COL", "CRI", "CUB", "DOM", "EC
# Oceanía
oceania_countries <- c("AUS", "FJI", "KIR", "MHL", "FSM", "NRU", "NZL", "PLW", "PNG", "WSM", "SLB", "TO
# Africa
africa_countries <- c("DZA", "AGO", "BEN", "BWA", "BFA", "BDI", "CPV", "CMR", "CAF", "TCD", "COM", "COG

# Siglas de regiones y organizaciones y otros lugare peculiares y y pintorescos
world_organizations <- c("EUR", "NAM", "SAM", "ANT", "OCE", "G-7", "ASM", "ATG", "AIA", "ABW", "BMU", "


# Combinamos todas las listas de códigos de países en una
all_countries <- c(eu_countries, other_europe_countries, asia_countries, america_countries, oceania_cou

# Creamos un vector con nombres de paises en ingles
all_country_nameI <- c("Austria", "Belgium", "Bulgaria", "Croatia", "Cyprus", "Czech Republic", "Denmar
```

```
all_country_name2 <- c("South-east Asia", "Upper-middle-income", "Low income", "Lower middle income", "

# fusionamos
all_country_names <- c(all_country_nameI, all_country_name2)

# Eliminamos lo que no necesitamos:
rm(africa_countries, america_countries, asia_countries, eu_countries, oceania_countries, other_europe_c
```

Next, we will search for ‘NA’ values in the ‘code’ column and assign them the ISO code based on the name or, if the name listed in the ‘entity’ column is already a code, we will assign them their corresponding name using the pairs in ‘all_country_names’ and ‘all_countries’.

```
# si 'code' es un NA, entonces en el vector de "codigos de pais" que casa con el par nombre en "nombres
full_pra$code <- ifelse(is.na(full_pra$code), all_countries[match(full_pra$entity, all_country_names)],

# Este caso es para aquellos registros que tienen un codigo de pais en 'entity' y tambien y a la vez, u
full_pra$entity <- ifelse(is.na(full_pra$code) & full_pra$entity %in% all_countries, all_country_names[

# Reemplaza "" en 'code' con los códigos correspondientes de all_countries.
full_pra <- full_pra %>% mutate(code = if_else(code == "", all_countries[match(entity, all_country_name
```

And we check to locate the errors, we represent them in a table to manage them better (printing is omitted in the pdf).

And at this point, we can confirm that we don’t have any issues with infinite or blank values in the first two columns.

Let's check the 'year' column for infinite or blank values:

```
# Buscamos filas donde 'year' es NA
na_rows <- is.na(full_pra$year)

# Buscamos las filas donde 'year' es 'blank'
blank_rows <- (full_pra$year == "")

# Vemos las filas donde 'year' es NA
full_pra[na_rows,]
```

```
##  [1] entity     code       year       imf_index  uhc_index  population
##  [7] u_1        u_5        u_15       u_25       btn_15_64  old_15
## [13] old_18     at_1       btn_1_4    btn_5_9    btn_10_14  btn_15_19
## [19] btn_20_29  btn_30_39  btn_40_49  btn_50_59  btn_60_69  btn_70_79
## [25] btn_80_89  btn_90_99  old_100    phy_index  gdp_index  ilo_index
## [31] bci_index
## <0 rows> (or 0-length row.names)
```

```
# vemos las filas donde 'year' es blank
full_pra[na_rows, ]
```

```
##  [1] entity     code       year       imf_index  uhc_index  population
##  [7] u_1        u_5        u_15       u_25       btn_15_64  old_15
## [13] old_18     at_1       btn_1_4    btn_5_9    btn_10_14  btn_15_19
## [19] btn_20_29  btn_30_39  btn_40_49  btn_50_59  btn_60_69  btn_70_79
## [25] btn_80_89  btn_90_99  old_100    phy_index  gdp_index  ilo_index
## [31] bci_index
## <0 rows> (or 0-length row.names)
```

We also know that the date column does not contain any errors. This is important because the combination of information and other data operations will rely on this column and the code column as primary keys.

The 'entity' column has duplicate values that we will unify using the key 'code' and 'year'. But first, we will try to select the shortest entity name among the duplicates with the same code and year.

```
# Agrupamos por 'year' y 'code':
full_pra <- full_pra %>%
  group_by(year, code) %>%
  mutate(
    entity = entity[which.min(nchar(entity))] # Ponemos los nombres mas cortos
  ) %>%
  ungroup()

# visualizamos la columna y los registros unicos evitando duplicados y los nombres es algo que se repite
# unique(full_pra$entity)
# omitimos la impresion
```

We observe that we have 282 names of countries, regions, organisations or, in short, entities.

The records of the same year and identification code are duplicates, we are going to merge them, saving the records that coincide by calculating the average:

```
# Identificamos las columnas numéricas y usamos este vector en aggregate para calcular la media de las
numeric_cols <- sapply(full_pra, is.numeric) # Crea un vector con todas las columnas numéricas

# Aplicamos aggregate:

full_pra_no_duplicados <- aggregate(full_pra[, numeric_cols], by=list( full_pra$entity, full_pra$code,

# Visualizamos la nueva tabla:

glimpse(full_pra_no_duplicados)
```

```
## Rows: 1,637
## Columns: 32
## $ Group.1    <chr> "Aruba", "Europe and Central Asia", "Afghanistan", "Africa"~
## $ Group.2    <chr> "ABW", "ACS", "AFG", "AFR", "AGO", "AIA", "ALB", "AMR", "AN~
## $ Group.3    <int> 2015, 2015, 2015, 2015, 2015, 2015, 2015, 2015, 2015, 2015,~
## $ year       <dbl> 2015, 2015, 2015, 2015, 2015, 2015, 2015, 2015, 2015, 2015,~
## $ imf_index  <dbl> NaN, 36.127605, 37.022568, NaN, 20.818832, NaN, 24.446177, ~
## $ uhc_index  <dbl> NaN, NaN, 32.19185, 45.35265, 36.67979, NaN, 61.65859, 76.1~
## $ population <dbl> 104269, NaN, 33753500, 1201108000, 28127724, 14554, 2882482~
## $ u_1        <dbl> 1139, NaN, 1238201, 40085576, 1117282, 152, 33567, NaN, 565~
## $ u_5        <dbl> 6026, NaN, 5747862, 189293000, 5158468, 925, 167618, NaN, 3~
## $ u_15       <dbl> 19599, NaN, 15456437, 495490940, 12871117, 2959, 523652, Na~
## $ u_25       <dbl> 32827, NaN, 22708012, 726413440, 18146508, 4789, 1019225, N~
## $ btn_15_64  <dbl> 72312, NaN, 17485012, 665997900, 14540242, 10451, 1968730, ~
## $ old_15     <dbl> 84669, NaN, 18296928, 705607200, 15256426, 11595, 2358651, ~
## $ old_18     <dbl> 80750, NaN, 15818155, 630187140, 13509109, 11055, 2218868, ~
## $ at_1       <dbl> 1170, NaN, 1187761, 38746430, 1067063, 173, 34071, NaN, 611~
## $ btn_1_4    <dbl> 4887, NaN, 4509661, 149207420, 4041186, 773, 134051, NaN, 2~
## $ btn_5_9    <dbl> 6677, NaN, 5083265, 164655650, 4267204, 1071, 160753, NaN, ~
## $ btn_10_14  <dbl> 6896, NaN, 4625310, 141542290, 3445445, 963, 195281, NaN, 3~
## $ btn_15_19  <dbl> 6533, NaN, 3998778, 122642700, 2823571, 892, 239277, NaN, 3~
## $ btn_20_29  <dbl> 12811, NaN, 5732466, 204278270, 4588283, 2153, 477068, NaN,~
## $ btn_30_39  <dbl> 13038, NaN, 3529267, 150031490, 3198090, 2576, 340003, NaN,~
## $ btn_40_49  <dbl> 15979, NaN, 2350849, 100395150, 2108238, 2341, 360196, NaN,~
## $ btn_50_59  <dbl> 17286, NaN, 1413659, 66087336, 1390927, 1845, 394606, NaN, ~
## $ btn_60_69  <dbl> 11532, NaN, 814056, 39015176, 727173, 1083, 284099, NaN, 65~
## $ btn_70_79  <dbl> 5707, NaN, 371848, 17785904, 337324, 507, 185620, NaN, 3672~
## $ btn_80_89  <dbl> 1667, NaN, 81378, 4936880, 77919, 175, 68581, NaN, 1890, 10~
## $ btn_90_99  <dbl> 116, NaN, 4628, 434246, 4901, 23, 9201, NaN, 519, 1400, 206~
## $ old_100    <dbl> 1, NaN, 133, 9835, 182, 0, 179, NaN, 27, 19, 6723, 67, 1874~
## $ phy_index  <dbl> NaN, NaN, 0.2850, NaN, NaN, NaN, NaN, NaN, 3.3333, 2.2111, ~
## $ gdp_index  <dbl> 5.1296573, 1.6336858, -1.6228572, NaN, -2.4687374, NaN, 2.5~
## $ ilo_index  <dbl> NaN, NaN, NaN, NaN, NaN, NaN, NaN, NaN, NaN, NaN, 2.19, 1.2~
## $ bci_index  <dbl> NaN, NaN, NaN, NaN, NaN, NaN, NaN, NaN, NaN, NaN, NaN, NaN,~
```

We note that we already have unique records with unique names, reducing the number of instances, to 1637 objects in total. The new table has a problem with the variable names, the type of columns and the 'NaN' (Not a number) values resulting from dividing NA in the aggregate function, as the function has calculated the averages from a division of zeros, returning infinity. We will address all the problems below and leave the 'NaN' problem for a next step when we work with each variable:

```r
# Renombramos Group.1 a entity y cremos un nuevo df
pra1 <- full_pra_no_duplicados %>% rename(entity = Group.1)

# Renombramos Group.2 a code
pra1 <- pra1 %>%  rename(code = Group.2)

# Eliminamos la columna Group.3 y dejamos la de 'year'
pra1 <- subset(pra1, select = -Group.3)

# Cambiamos el tipo de variables de punto flotante a integrales usando un 'for' en "year", "population"

puntos_flotantes <- c("year", "population", "u_1", "u_5", "u_15", "u_25", "btn_15_64", "old_15", "old_1

# Bucle for para cambiar cada columna
for (col in puntos_flotantes) {
  if (col %in% names(pra1)) {
    pra1[[col]] <- as.integer(pra1[[col]])
  } else {
    warning(paste("La columna", col, "no se encuentra en el dataframe."))
  }
}

# Necesitamos cambiar el orden de las columnas para poder entender los datos mejor de un vistazo
orden <- c("entity", "code", "year", "gdp_index", "imf_index", "uhc_index", "bci_index", "ilo_index", "

# Reordenar las columnas
pra1 <- pra1[, orden]



# visualizamos el dataframe:
glimpse(pra1)
```

```
## Rows: 1,637
## Columns: 31
## $ entity      <chr> "Aruba", "Europe and Central Asia", "Afghanistan", "Africa"~
## $ code        <chr> "ABW", "ACS", "AFG", "AFR", "AGO", "AIA", "ALB", "AMR", "AN~
## $ year        <int> 2015, 2015, 2015, 2015, 2015, 2015, 2015, 2015, 2015, 2015,~
## $ gdp_index   <dbl> 5.1296573, 1.6336858, -1.6228572, NaN, -2.4687374, NaN, 2.5~
## $ imf_index   <dbl> NaN, 36.127605, 37.022568, NaN, 20.818832, NaN, 24.446177, ~
## $ uhc_index   <dbl> NaN, NaN, 32.19185, 45.35265, 36.67979, NaN, 61.65859, 76.1~
## $ bci_index   <dbl> NaN, NaN, NaN, NaN, NaN, NaN, NaN, NaN, NaN, NaN, NaN, NaN,~
## $ ilo_index   <dbl> NaN, NaN, NaN, NaN, NaN, NaN, NaN, NaN, NaN, NaN, 2.19, 1.2~
## $ phy_index   <dbl> NaN, NaN, 0.2850, NaN, NaN, NaN, NaN, NaN, 3.3333, 2.2111, ~
## $ u_1         <int> 1139, NA, 1238201, 40085576, 1117282, 152, 33567, NA, 565, ~
## $ u_5         <int> 6026, NA, 5747862, 189293000, 5158468, 925, 167618, NA, 335~
## $ u_15        <int> 19599, NA, 15456437, 495490940, 12871117, 2959, 523652, NA,~
## $ u_25        <int> 32827, NA, 22708012, 726413440, 18146508, 4789, 1019225, NA~
## $ btn_15_64   <int> 72312, NA, 17485012, 665997900, 14540242, 10451, 1968730, N~
## $ old_15      <int> 84669, NA, 18296928, 705607200, 15256426, 11595, 2358651, N~
## $ old_18      <int> 80750, NA, 15818155, 630187140, 13509109, 11055, 2218868, N~
## $ at_1        <int> 1170, NA, 1187761, 38746430, 1067063, 173, 34071, NA, 611, ~
## $ btn_1_4     <int> 4887, NA, 4509661, 149207420, 4041186, 773, 134051, NA, 278~
## $ btn_5_9    <int> 6677, NA, 5083265, 164655650, 4267204, 1071, 160753, NA, 39~
## $ btn_10_14  <int> 6896, NA, 4625310, 141542290, 3445445, 963, 195281, NA, 384~
## $ btn_15_19  <int> 6533, NA, 3998778, 122642700, 2823571, 892, 239277, NA, 387~
## $ btn_20_29  <int> 12811, NA, 5732466, 204278270, 4588283, 2153, 477068, NA, 7~
## $ btn_30_39  <int> 13038, NA, 3529267, 150031490, 3198090, 2576, 340003, NA, 1~
## $ btn_40_49  <int> 15979, NA, 2350849, 100395150, 2108238, 2341, 360196, NA, 1~
## $ btn_50_59  <int> 17286, NA, 1413659, 66087336, 1390927, 1845, 394606, NA, 10~
## $ btn_60_69  <int> 11532, NA, 814056, 39015176, 727173, 1083, 284099, NA, 6516~
## $ btn_70_79  <int> 5707, NA, 371848, 17785904, 337324, 507, 185620, NA, 3672, ~
## $ btn_80_89  <int> 1667, NA, 81378, 4936880, 77919, 175, 68581, NA, 1890, 1018~
## $ btn_90_99  <int> 116, NA, 4628, 434246, 4901, 23, 9201, NA, 519, 1400, 20624~
## $ old_100    <int> 1, NA, 133, 9835, 182, 0, 179, NA, 27, 19, 6723, 67, 187429~
## $ population <int> 104269, NA, 33753500, 1201108000, 28127724, 14554, 2882482,~
```

```
# Visualizamos las variables
names(pra1)
```

```
##  [1] "entity"     "code"       "year"       "gdp_index"  "imf_index"
##  [6] "uhc_index"  "bci_index"  "ilo_index"  "phy_index"  "u_1"
## [11] "u_5"        "u_15"       "u_25"       "btn_15_64"  "old_15"
## [16] "old_18"     "at_1"       "btn_1_4"    "btn_5_9"    "btn_10_14"
## [21] "btn_15_19"  "btn_20_29"  "btn_30_39"  "btn_40_49"  "btn_50_59"
## [26] "btn_60_69"  "btn_70_79"  "btn_80_89"  "btn_90_99"  "old_100"
## [31] "population"
```

We now have a table “pra1” to work on, with all its records unified. Let’s clean up the working environment:

```
# Creamos una lista de todas las variables
all_vars <- ls()

# Creamos una lista de las variables a mantener
keep_vars <- "pra1"

# Creamos una lista de las variables a borrar con setdiff
delete_vars <- setdiff(all_vars, keep_vars)

# Borrar lo que ya no necesitamos:
rm(list = delete_vars)
rm(all_vars)
rm(keep_vars)
rm(delete_vars)

# Copia de seguridad del dataframe:
write.csv(pra1, file= "pra1.csv")
```

Creating regions for each country

Once we have ensured the integrity of the table and before we start searching for ‘NA’ and blank values on the dataframe, for example on variables such as ‘gdp’ and the different populations, we need to create regions.

```
# Buscamos filas donde 'year' es NA
na_rows <- is.na(pra1$gdp_index)

# Buscamos las filas donde 'year' es 'blank'
blank_rows <- (pra1$gdp_index == "")

# Vemos las filas donde 'year' es NA
pra1[na_rows,]

# vemos las filas donde 'year' es blank
pra1[na_rows, ]
```

We observed numerous NaN (Not a Number).

So far, we have kept different data relating to geographical subdivisions containing numerical values referring to the different numerical indices that we have given to collect as variables that we will use to impute missing data. We are going to create geographical subdivisions with the data:

```
# Codigos UE
eu_countries <- unique(c("AUT", "BEL", "BGR", "HRV", "CYP", "CZE", "DNK", "EST", "FIN", "FRA", "DEU", "
# Resto de Europa (no UE)
other_europe_countries <- unique(c("ALB", "AND", "BIH", "BLR", "CHE", "GEO", "ISL", "LIE", "MDA", "MKD"
# Asia
asia_countries <- unique(c("AFG", "ARM", "AZE", "BHR", "BGD", "BTN", "BRN", "KHM", "CHN", "IND", "IDN",

# América del Norte
north_america_countries <- unique(c("USA", "CAN", "MEX", "PRI", "ASM", "ATG", "BMU", "TCA", "MNP"))

# América Central
central_america_countries <- unique(c("BLZ", "CRI", "SLV", "GTM", "HND", "NIC", "PAN"))

# América del Sur
south_america_countries <- unique(c("ARG", "BOL", "BRA", "CHL", "COL", "ECU", "GUY", "PRY", "PER", "SUR

# Caribe
caribbean_countries <- unique(c("BHS", "BRB", "CUB", "DOM", "GRD", "JAM", "KNA", "LCA", "VCT", "TTO", "


# Oceanía
oceania_countries <- unique(c("AUS", "FJI", "KIR", "MHL", "FSM", "NRU", "NZL", "PLW", "PNG", "WSM", "SL

# Africa
africa_countries <- unique(c("DZA", "AGO", "BEN", "BWA", "BFA", "BDI", "CPV", "CMR", "CAF", "TCD", "COM


# Siglas de regiones y organizaciones y otros lugare peculiares y y pintorescos
world_organizations <- unique(c("EUR", "NAM", "SAM", "ANT", "OCE", "G-7", "ASM", "ATG", "AIA", "BMU", "


# Juntamos todas las listas de códigos de países en una sola
all_countries <- c(north_america_countries, central_america_countries, south_america_countries, caribbe

# Eliminamos de 'world_organizations' los códigos que ya están en 'all_countries'
world_organizations <- world_organizations[!world_organizations %in% all_countries]
```

We now need to classify the registers geographically or according to whether they are organisations, divisions or entities:

```r
# Aseguramos que los códigos son todos mayúsculas en el dataframe:
pra1$code <- toupper(pra1$code)

# Añadimos la nueva columna 'region' al dataframe 'pra1':
pra1 <- pra1 %>%
  mutate(region = case_when(
    code %in% eu_countries ~ "UE",
    code %in% other_europe_countries ~ "Europa no UE",
    code %in% asia_countries ~ "Asia",
    code %in% north_america_countries ~ "América del Norte",
    code %in% central_america_countries ~ "América Central",
    code %in% south_america_countries ~ "América del Sur",
    code %in% caribbean_countries ~ "Caribe",
    code %in% oceania_countries ~ "Oceanía",
    code %in% africa_countries ~ "África",
    code %in% world_organizations ~ "Organizaciones y otros",
    TRUE ~ "No clasificado"
  ))


# Aseguramos el formato de los nombres de variables:
pra1 <- pra1 %>% clean_names()

# Observamos la tabla
glimpse(pra1)
```

```
## Rows: 1,637
## Columns: 32
## $ entity     <chr> "Aruba", "Europe and Central Asia", "Afghanistan", "Africa"~
## $ code       <chr> "ABW", "ACS", "AFG", "AFR", "AGO", "AIA", "ALB", "AMR", "AN~
## $ year       <int> 2015, 2015, 2015, 2015, 2015, 2015, 2015, 2015, 2015, 2015,~
## $ gdp_index  <dbl> 5.1296573, 1.6336858, -1.6228572, NaN, -2.4687374, NaN, 2.5~
## $ imf_index  <dbl> NaN, 36.127605, 37.022568, NaN, 20.818832, NaN, 24.446177, ~
## $ uhc_index  <dbl> NaN, NaN, 32.19185, 45.35265, 36.67979, NaN, 61.65859, 76.1~
## $ bci_index  <dbl> NaN, NaN, NaN, NaN, NaN, NaN, NaN, NaN, NaN, NaN, NaN, NaN,~
## $ ilo_index  <dbl> NaN, NaN, NaN, NaN, NaN, NaN, NaN, NaN, NaN, NaN, 2.19, 1.2~
## $ phy_index  <dbl> NaN, NaN, 0.2850, NaN, NaN, NaN, NaN, NaN, 3.3333, 2.2111, ~
## $ u_1        <int> 1139, NA, 1238201, 40085576, 1117282, 152, 33567, NA, 565, ~
## $ u_5        <int> 6026, NA, 5747862, 189293000, 5158468, 925, 167618, NA, 335~
## $ u_15       <int> 19599, NA, 15456437, 495490940, 12871117, 2959, 523652, NA,~
## $ u_25       <int> 32827, NA, 22708012, 726413440, 18146508, 4789, 1019225, NA~
## $ btn_15_64  <int> 72312, NA, 17485012, 665997900, 14540242, 10451, 1968730, N~
## $ old_15     <int> 84669, NA, 18296928, 705607200, 15256426, 11595, 2358651, N~
## $ old_18     <int> 80750, NA, 15818155, 630187140, 13509109, 11055, 2218868, N~
## $ at_1       <int> 1170, NA, 1187761, 38746430, 1067063, 173, 34071, NA, 611, ~
## $ btn_1_4    <int> 4887, NA, 4509661, 149207420, 4041186, 773, 134051, NA, 278~
## $ btn_5_9    <int> 6677, NA, 5083265, 164655650, 4267204, 1071, 160753, NA, 39~
## $ btn_10_14  <int> 6896, NA, 4625310, 141542290, 3445445, 963, 195281, NA, 384~
## $ btn_15_19  <int> 6533, NA, 3998778, 122642700, 2823571, 892, 239277, NA, 387~
## $ btn_20_29  <int> 12811, NA, 5732466, 204278270, 4588283, 2153, 477068, NA, 7~
## $ btn_30_39  <int> 13038, NA, 3529267, 150031490, 3198090, 2576, 340003, NA, 1~
## $ btn_40_49  <int> 15979, NA, 2350849, 100395150, 2108238, 2341, 360196, NA, 1~
## $ btn_50_59  <int> 17286, NA, 1413659, 66087336, 1390927, 1845, 394606, NA, 10~
## $ btn_60_69  <int> 11532, NA, 814056, 39015176, 727173, 1083, 284099, NA, 6516~
## $ btn_70_79  <int> 5707, NA, 371848, 17785904, 337324, 507, 185620, NA, 3672, ~
## $ btn_80_89  <int> 1667, NA, 81378, 4936880, 77919, 175, 68581, NA, 1890, 1018~
## $ btn_90_99  <int> 116, NA, 4628, 434246, 4901, 23, 9201, NA, 519, 1400, 20624~
## $ old_100    <int> 1, NA, 133, 9835, 182, 0, 179, NA, 27, 19, 6723, 67, 187429~
## $ population <int> 104269, NA, 33753500, 1201108000, 28127724, 14554, 2882482,~
## $ region     <chr> "Caribe", "Organizaciones y otros", "Asia", "Organizaciones~
```

```
# Analizamos la nueva columna 'region' en busca de blancos y NA
na_rows <- is.na(pra1$region)

# Buscamos blank
blank_rows <- (pra1$region == "")

# Vemos las filas donde 'year' es NA
pra1[na_rows,]
```

```
##  [1] entity     code       year       gdp_index  imf_index  uhc_index 
##  [7] bci_index  ilo_index  phy_index  u_1        u_5        u_15      
## [13] u_25       btn_15_64  old_15     old_18     at_1       btn_1_4   
## [19] btn_5_9    btn_10_14  btn_15_19  btn_20_29  btn_30_39  btn_40_49 
## [25] btn_50_59  btn_60_69  btn_70_79  btn_80_89  btn_90_99  old_100   
## [31] population region    
## <0 rows> (or 0-length row.names)
```

```
# vemos las filas donde 'year' es blank
pra1[na_rows, ]
```

```
##  [1] entity     code       year       gdp_index  imf_index  uhc_index 
##  [7] bci_index  ilo_index  phy_index  u_1        u_5        u_15      
## [13] u_25       btn_15_64  old_15     old_18     at_1       btn_1_4   
## [19] btn_5_9    btn_10_14  btn_15_19  btn_20_29  btn_30_39  btn_40_49 
## [25] btn_50_59  btn_60_69  btn_70_79  btn_80_89  btn_90_99  old_100   
## [31] population region    
## <0 rows> (or 0-length row.names)
```

'region' is a new categorical variable. Its format is correct <chr> and that there is no 'NA' or 'blank'.

Tidy up the dataframe to better visualise the data and detect errors:

```
# Ordenamos el dataframe segun la region, el codigo y la entidad
pra1_ordenado <- pra1 %>%
  arrange(region, code, entity)

head(pra1_ordenado)
```

```
##   entity code year   gdp_index imf_index uhc_index bci_index ilo_index
## 1 Belize  BLZ 2015   0.4879972  26.26154  64.14784       NaN      0.53
## 2 Belize  BLZ 2016  -2.0610666  26.62480       NaN       NaN      0.53
## 3 Belize  BLZ 2017  -0.1847950  27.24283  64.91026       NaN      0.53
## 4 Belize  BLZ 2018   0.9461128       NaN       NaN       NaN      0.53
## 5 Belize  BLZ 2019  -0.1350662       NaN  67.16825       NaN      0.53
## 6 Belize  BLZ 2020 -15.5827417       NaN       NaN       NaN      0.53
##   phy_index  u_1   u_5   u_15   u_25 btn_15_64 old_15 old_18 at_1 btn_1_4
## 1       NaN 7708 37541 114345 188185    230150 245538 222378 7644   29833
## 2       NaN 7743 38042 114272 189431    237024 253054 229557 7704   30299
## 3    1.1229 7800 38445 114269 190504    243732 260431 236793 7739   30645
## 4    1.0780 7896 38809 114409 191488    250266 267664 244045 7797   30913
## 5       NaN 7615 38777 114282 192038    256676 274817 251284 7892   31162
## 6       NaN 7138 38192 113644 191753    262301 281281 257858 7608   31054
##   btn_5_9 btn_10_14 btn_15_19 btn_20_29 btn_30_39 btn_40_49 btn_50_59 btn_60_69
## 1   37788     39016     38018     67852     51375     38610     26066     13971
## 2   37339     38891     38616     69636     53139     39860     27128     14741
## 3   37104     38720     39090     71273     54993     41112     28178     15534
## 4   37076     38524     39356     72837     56930     42337     29232     16362
## 5   37226     38279     39426     74369     58927     43520     30329     17229
## 6   37611     37841     39298     75539     60769     44630     31417     18121
##   btn_70_79 btn_80_89 btn_90_99 old_100 population          region
## 1      6679      2686       281       1     359884 América Central
## 2      6869      2750       315       1     367327 América Central
## 3      7094      2813       344       2     374702 América Central
## 4      7363      2873       374       3     382076 América Central
## 5      7676      2934       407       4     389103 América Central
## 6      8054      3006       447       6     394931 América Central
```

Proceeding to search for codes longer than three characters:

```
# seleccionamos, si existen, registros cuyo código sea mayor de 3 caracteres:

error_code <- subset(pra1_ordenado, nchar(code) > 3)

# Imprimimos los resultados
print(error_code)
```

```
##      entity     code year gdp_index imf_index uhc_index bci_index ilo_index
## 745  Kosovo OWID_KOS 2015  7.371867       NaN       NaN       NaN       NaN
## 746  Kosovo OWID_KOS 2016  6.203622       NaN       NaN       NaN       NaN
## 747  Kosovo OWID_KOS 2017  4.038709       NaN       NaN       NaN       NaN
## 748  Kosovo OWID_KOS 2018  3.056665       NaN       NaN       NaN       NaN
## 749  Kosovo OWID_KOS 2019  5.237433       NaN       NaN       NaN       NaN
## 750  Kosovo OWID_KOS 2020 -5.406638       NaN       NaN       NaN       NaN
## 1096   OECD     OECD 2015       NaN       NaN       NaN 100.05208       NaN
## 1097   OECD     OECD 2016       NaN       NaN       NaN 100.08704       NaN
## 1098   OECD     OECD 2017       NaN       NaN       NaN 101.10919       NaN
## 1099   OECD     OECD 2018       NaN       NaN       NaN 101.22548       NaN
## 1100   OECD     OECD 2019       NaN       NaN       NaN  99.87191       NaN
## 1101   OECD     OECD 2020       NaN       NaN       NaN  98.69282       NaN
## 1102  World OWID_WRL 2015  1.963134  27.57807  63.57609       NaN      5.33
## 1103  World OWID_WRL 2016  1.636977  27.41954       NaN       NaN      5.33
## 1104  World OWID_WRL 2017  2.217117  27.34953  65.49816       NaN      5.31
## 1105  World OWID_WRL 2018  2.138745  27.21464       NaN       NaN      5.29
## 1106  World OWID_WRL 2019  1.519110  27.92516  67.48169       NaN      5.11
## 1107  World OWID_WRL 2020 -4.290962       NaN       NaN       NaN      5.01
##      phy_index       u_1       u_5       u_15 u_25 btn_15_64 old_15 old_18
## 745        NaN        NA        NA         NA   NA        NA     NA     NA
## 746        NaN        NA        NA         NA   NA        NA     NA     NA
## 747        NaN        NA        NA         NA   NA        NA     NA     NA
## 748        NaN        NA        NA         NA   NA        NA     NA     NA
## 749        NaN        NA        NA         NA   NA        NA     NA     NA
## 750        NaN        NA        NA         NA   NA        NA     NA     NA
## 1096       NaN        NA        NA         NA   NA        NA     NA     NA
## 1097       NaN        NA        NA         NA   NA        NA     NA     NA
## 1098       NaN        NA        NA         NA   NA        NA     NA     NA
## 1099       NaN        NA        NA         NA   NA        NA     NA     NA
## 1100       NaN        NA        NA         NA   NA        NA     NA     NA
## 1101       NaN        NA        NA         NA   NA        NA     NA     NA
## 1102       NaN 139309630 687534000 1964306700   NA        NA     NA     NA
## 1103       NaN 139262160 689597950 1977518000   NA        NA     NA     NA
## 1104  1.756681 139420590 690360700 1990610400   NA        NA     NA     NA
## 1105       NaN 137690030 688660400 2001436500   NA        NA     NA     NA
## 1106       NaN 135471330 684872600 2009205500   NA        NA     NA     NA
## 1107       NaN        NA        NA         NA   NA        NA     NA     NA
##           at_1   btn_1_4   btn_5_9 btn_10_14 btn_15_19  btn_20_29  btn_30_39
## 745         NA        NA        NA        NA        NA         NA         NA
## 746         NA        NA        NA        NA        NA         NA         NA
## 747         NA        NA        NA        NA        NA         NA         NA
## 748         NA        NA        NA        NA        NA         NA         NA
## 749         NA        NA        NA        NA        NA         NA         NA
## 750         NA        NA        NA        NA        NA         NA         NA
## 1096        NA        NA        NA        NA        NA         NA         NA
## 1097        NA        NA        NA        NA        NA         NA         NA
## 1098        NA        NA        NA        NA        NA         NA         NA
## 1099        NA        NA        NA        NA        NA         NA         NA
## 1100        NA        NA        NA        NA        NA         NA         NA
## 1101        NA        NA        NA        NA        NA         NA         NA
## 1102 138354700 548224400 654825150 621947460 602199500 1210493200 1054263200
## 1103 138167890 550335740 661842400 626077600 604712060 1210461200 1072298300
## 1104 138183950 550940160 668653200 631596500 607785800 1207877900 1092995100
## 1105 138376850 550970400 674324740 638451460 611398340 1204281300 1114043900
## 1106 136690500 549401300 678417150 645915800 615536000 1200389900 1133424100
## 1107        NA        NA        NA        NA        NA         NA         NA
##      btn_40_49 btn_50_59 btn_60_69 btn_70_79 btn_80_89 btn_90_99 old_100
## 745         NA        NA        NA        NA        NA        NA      NA
## 746         NA        NA        NA        NA        NA        NA      NA
## 747         NA        NA        NA        NA        NA        NA      NA
## 748         NA        NA        NA        NA        NA        NA      NA
## 749         NA        NA        NA        NA        NA        NA      NA
## 750         NA        NA        NA        NA        NA        NA      NA
## 1096        NA        NA        NA        NA        NA        NA      NA
## 1097        NA        NA        NA        NA        NA        NA      NA
## 1098        NA        NA        NA        NA        NA        NA      NA
## 1099        NA        NA        NA        NA        NA        NA      NA
## 1100        NA        NA        NA        NA        NA        NA      NA
## 1101        NA        NA        NA        NA        NA        NA      NA
## 1102 940925440 741697700 511844960 271247680 112493050  16703625  422479
## 1103 947469700 758017000 531722300 277227970 115949520  17657084  441166
## 1104 953971900 772621950 549785660 285532670 119566930  18615136  459150
## 1105 958684500 788847800 566538940 295304100 123199544  19578904  475921
## 1106 963090600 809066200 579960400 306498750 126700504  20575482  503572
## 1107        NA        NA        NA        NA        NA        NA      NA
##      population                  region
## 745          NA            Europa no UE
## 746          NA            Europa no UE
## 747          NA            Europa no UE
## 748          NA            Europa no UE
## 749          NA            Europa no UE
## 750          NA            Europa no UE
## 1096         NA Organizaciones y otros
## 1097         NA Organizaciones y otros
## 1098         NA Organizaciones y otros
## 1099         NA Organizaciones y otros
## 1100         NA Organizaciones y otros
## 1101         NA Organizaciones y otros
## 1102         NA Organizaciones y otros
## 1103         NA Organizaciones y otros
## 1104         NA Organizaciones y otros
## 1105         NA Organizaciones y otros
## 1106         NA Organizaciones y otros
## 1107         NA Organizaciones y otros
```

Seven records have been merged, reducing the total number of objects from 1637 to 1630. We have noticed that the codes and entities are not fully unified yet, as the codes do not match. We will rectify this, taking into account that we have instances of ‘kosovo’, ‘OCDE’, and ‘World’.

```
# Rectificamos los códigos de Kosovo a XKX:
pra1_ordenado$code <- sub("OWID_KOS", "XKX", pra1_ordenado$code)

# Rectificamos los códigos de OECDE a OCD:
pra1_ordenado$code <- sub("OECDE", "OCD", pra1_ordenado$code)

# Rectificamos los códigos de OWID_WRL a WRL:
pra1_ordenado$code <- sub("OWID_WRL", "WRL", pra1_ordenado$code)

# Repetimos, seleccionamos, si existen, registros cuyo código sea mayor de 3 caracteres y ver si los ca
error_code <- subset(pra1_ordenado, nchar(code) > 3)

# Imprimimos los resultados actualizados
print(error_code)
```

```
##      entity code year gdp_index imf_index uhc_index bci_index ilo_index
## 1096   OECD OECD 2015       NaN       NaN       NaN 100.05208       NaN
## 1097   OECD OECD 2016       NaN       NaN       NaN 100.08704       NaN
## 1098   OECD OECD 2017       NaN       NaN       NaN 101.10919       NaN
## 1099   OECD OECD 2018       NaN       NaN       NaN 101.22548       NaN
## 1100   OECD OECD 2019       NaN       NaN       NaN  99.87191       NaN
## 1101   OECD OECD 2020       NaN       NaN       NaN  98.69282       NaN
##      phy_index u_1 u_5 u_15 u_25 btn_15_64 old_15 old_18 at_1 btn_1_4 btn_5_9
## 1096       NaN  NA  NA   NA   NA        NA     NA     NA   NA      NA      NA
## 1097       NaN  NA  NA   NA   NA        NA     NA     NA   NA      NA      NA
## 1098       NaN  NA  NA   NA   NA        NA     NA     NA   NA      NA      NA
## 1099       NaN  NA  NA   NA   NA        NA     NA     NA   NA      NA      NA
## 1100       NaN  NA  NA   NA   NA        NA     NA     NA   NA      NA      NA
## 1101       NaN  NA  NA   NA   NA        NA     NA     NA   NA      NA      NA
##      btn_10_14 btn_15_19 btn_20_29 btn_30_39 btn_40_49 btn_50_59 btn_60_69
## 1096        NA        NA        NA        NA        NA        NA        NA
## 1097        NA        NA        NA        NA        NA        NA        NA
## 1098        NA        NA        NA        NA        NA        NA        NA
## 1099        NA        NA        NA        NA        NA        NA        NA
## 1100        NA        NA        NA        NA        NA        NA        NA
## 1101        NA        NA        NA        NA        NA        NA        NA
##      btn_70_79 btn_80_89 btn_90_99 old_100 population                 region
## 1096        NA        NA        NA      NA         NA Organizaciones y otros
## 1097        NA        NA        NA      NA         NA Organizaciones y otros
## 1098        NA        NA        NA      NA         NA Organizaciones y otros
## 1099        NA        NA        NA      NA         NA Organizaciones y otros
## 1100        NA        NA        NA      NA         NA Organizaciones y otros
## 1101        NA        NA        NA      NA         NA Organizaciones y otros
```

```
# Salvamos una copia del dataframe:
write.csv(pra1_ordenado, file= "pra1_ordenado.csv")
```

Now there are no codes with inappropriate formats, but there are still several duplicate entities in the dataset (based on entity-year criteria). Therefore, we will once again use aggregate() to recombine the information in the table.

```
# Identificamos las columnas numéricas y usamos este vector en aggregate para calcular la media de las 

numeric_cols <- sapply(pra1_ordenado, is.numeric) # Crea un vector con todas las columnas numéricas

# Aplicamos aggregate:

pra1 <- aggregate(pra1_ordenado[, numeric_cols], by=list( pra1_ordenado$entity, pra1_ordenado$code, pra

# Visualizamos la nueva tabla:

glimpse(pra1)
```

```
## Rows: 1,630
## Columns: 32
## $ Group.1    <chr> "Aruba", "Europe and Central Asia", "Afghanistan", "Africa"~
## $ Group.2    <chr> "ABW", "ACS", "AFG", "AFR", "AGO", "AIA", "ALB", "AMR", "AN~
## $ Group.3    <int> 2015, 2015, 2015, 2015, 2015, 2015, 2015, 2015, 2015, 2015,~
## $ year       <dbl> 2015, 2015, 2015, 2015, 2015, 2015, 2015, 2015, 2015, 2015,~
## $ gdp_index  <dbl> 5.1296573, 1.6336858, -1.6228572, NaN, -2.4687374, NaN, 2.5~
## $ imf_index  <dbl> NaN, 36.127605, 37.022568, NaN, 20.818832, NaN, 24.446177, ~
## $ uhc_index  <dbl> NaN, NaN, 32.19185, 45.35265, 36.67979, NaN, 61.65859, 76.1~
## $ bci_index  <dbl> NaN, NaN, NaN, NaN, NaN, NaN, NaN, NaN, NaN, NaN, NaN, NaN,~
## $ ilo_index  <dbl> NaN, NaN, NaN, NaN, NaN, NaN, NaN, NaN, NaN, NaN, 2.19, 1.2~
## $ phy_index  <dbl> NaN, NaN, 0.2850, NaN, NaN, NaN, NaN, NaN, 3.3333, 2.2111, ~
## $ u_1        <dbl> 1139, NaN, 1238201, 40085576, 1117282, 152, 33567, NaN, 565~
## $ u_5        <dbl> 6026, NaN, 5747862, 189293000, 5158468, 925, 167618, NaN, 3~
## $ u_15       <dbl> 19599, NaN, 15456437, 495490940, 12871117, 2959, 523652, Na~
## $ u_25       <dbl> 32827, NaN, 22708012, 726413440, 18146508, 4789, 1019225, N~
## $ btn_15_64  <dbl> 72312, NaN, 17485012, 665997900, 14540242, 10451, 1968730, ~
## $ old_15     <dbl> 84669, NaN, 18296928, 705607200, 15256426, 11595, 2358651, ~
## $ old_18     <dbl> 80750, NaN, 15818155, 630187140, 13509109, 11055, 2218868, ~
## $ at_1       <dbl> 1170, NaN, 1187761, 38746430, 1067063, 173, 34071, NaN, 611~
## $ btn_1_4    <dbl> 4887, NaN, 4509661, 149207420, 4041186, 773, 134051, NaN, 2~
## $ btn_5_9    <dbl> 6677, NaN, 5083265, 164655650, 4267204, 1071, 160753, NaN, ~
## $ btn_10_14  <dbl> 6896, NaN, 4625310, 141542290, 3445445, 963, 195281, NaN, 3~
## $ btn_15_19  <dbl> 6533, NaN, 3998778, 122642700, 2823571, 892, 239277, NaN, 3~
## $ btn_20_29  <dbl> 12811, NaN, 5732466, 204278270, 4588283, 2153, 477068, NaN,~
## $ btn_30_39  <dbl> 13038, NaN, 3529267, 150031490, 3198090, 2576, 340003, NaN,~
## $ btn_40_49  <dbl> 15979, NaN, 2350849, 100395150, 2108238, 2341, 360196, NaN,~
## $ btn_50_59  <dbl> 17286, NaN, 1413659, 66087336, 1390927, 1845, 394606, NaN, ~
## $ btn_60_69  <dbl> 11532, NaN, 814056, 39015176, 727173, 1083, 284099, NaN, 65~
## $ btn_70_79  <dbl> 5707, NaN, 371848, 17785904, 337324, 507, 185620, NaN, 3672~
## $ btn_80_89  <dbl> 1667, NaN, 81378, 4936880, 77919, 175, 68581, NaN, 1890, 10~
## $ btn_90_99  <dbl> 116, NaN, 4628, 434246, 4901, 23, 9201, NaN, 519, 1400, 206~
## $ old_100    <dbl> 1, NaN, 133, 9835, 182, 0, 179, NaN, 27, 19, 6723, 67, 1874~
## $ population <dbl> 104269, NaN, 33753500, 1201108000, 28127724, 14554, 2882482~
```

```
# copia de seguridad:
pra1_c <- pra1
```

As before, have unwanted changes in column type and variable names. We need to reuse the strategy and code used previously:

```
# Renombramos Group.1 a entity y cremos un nuevo df
pra1 <- pra1 %>% rename(entity = Group.1)

# Renombramos Group.2 a code
pra1 <- pra1 %>%  rename(code = Group.2)

# Eliminamos la columna Group.3 y dejamos la de 'year'
pra1 <- subset(pra1, select = -Group.3)

# Cambiamos el tipo de variables de punto flotante a integrales usando un 'for' en "year", "population"

puntos_flotantes <- c("year", "population", "u_1", "u_5", "u_15", "u_25", "btn_15_64", "old_15", "old_1

# Bucle for para cambiar cada columna
for (col in puntos_flotantes) {
  if (col %in% names(pra1)) {
    pra1[[col]] <- as.integer(pra1[[col]])
  } else {
    warning(paste("La columna", col, "no se encuentra en el dataframe."))
  }
}

##### Recreamos region ######################
# Aseguramos que los códigos son todos mayúsculas en el dataframe:
pra1$code <- toupper(pra1$code)

# Añadimos la nueva columna 'region' al dataframe 'pra1':
pra1 <- pra1 %>%
  mutate(region = case_when(
```

```r
    code %in% eu_countries ~ "UE",
    code %in% other_europe_countries ~ "Europa no UE",
    code %in% asia_countries ~ "Asia",
    code %in% north_america_countries ~ "América del Norte",
    code %in% central_america_countries ~ "América Central",
    code %in% south_america_countries ~ "América del Sur",
    code %in% caribbean_countries ~ "Caribe",
    code %in% oceania_countries ~ "Oceanía",
    code %in% africa_countries ~ "África",
    code %in% world_organizations ~ "Organizaciones y otros",
    TRUE ~ "No clasificado"
  ))

# Ordenamos el dataframe segun la region, el codigo y la entidad
pra1 <- pra1 %>%
  arrange(region, code, entity)

# Aplicamos 'clean_names'
pra1 <- pra1 %>% clean_names()


# visualizamos el dataframe:
glimpse(pra1)
```

```
## Rows: 1,630
## Columns: 32
## $ entity     <chr> "Belize", "Belize", "Belize", "Belize", "Belize", "Belize",~
## $ code       <chr> "BLZ", "BLZ", "BLZ", "BLZ", "BLZ", "BLZ", "CRI", "CRI", "CR~
## $ year       <int> 2015, 2016, 2017, 2018, 2019, 2020, 2015, 2016, 2017, 2018,~
## $ gdp_index  <dbl> 0.4879972, -2.0610666, -0.1847950, 0.9461128, -0.1350662, -~
## $ imf_index  <dbl> 26.26154, 26.62480, 27.24283, NaN, NaN, NaN, 28.55489, 27.1~
## $ uhc_index  <dbl> 64.14784, NaN, 64.91026, NaN, 67.16825, NaN, 73.84337, NaN,~
## $ bci_index  <dbl> NaN, NaN, NaN, NaN, NaN, NaN, NaN, NaN, NaN, NaN, NaN, NaN,~
## $ ilo_index  <dbl> 0.53, 0.53, 0.53, 0.53, 0.53, 0.53, 1.95, 1.57, 1.40, 1.40,~
## $ phy_index  <dbl> NaN, NaN, 1.1229, 1.0780, NaN, NaN, 1.3794, 1.3759, 1.3386,~
## $ u_1        <int> 7708, 7743, 7800, 7896, 7615, 7138, 71058, 70078, 69325, 68~
## $ u_5        <int> 37541, 38042, 38445, 38809, 38777, 38192, 361427, 358095, 3~
## $ u_15       <int> 114345, 114272, 114269, 114409, 114282, 113644, 1105736, 10~
## $ u_25       <int> 188185, 189431, 190504, 191488, 192038, 191753, 1922674, 19~
## $ btn_15_64  <int> 230150, 237024, 243732, 250266, 256676, 262301, 3363121, 34~
## $ old_15     <int> 245538, 253054, 260431, 267664, 274817, 281281, 3789380, 38~
## $ old_18     <int> 222378, 229557, 236793, 244045, 251284, 257858, 3552173, 36~
## $ at_1       <int> 7644, 7704, 7739, 7797, 7892, 7608, 71613, 71022, 70045, 69~
## $ btn_1_4    <int> 29833, 30299, 30645, 30913, 31162, 31054, 290369, 288017, 2~
## $ btn_5_9    <int> 37788, 37339, 37104, 37076, 37226, 37611, 370950, 371458, 3~
## $ btn_10_14  <int> 39016, 38891, 38720, 38524, 38279, 37841, 373359, 368701, 3~
## $ btn_15_19  <int> 38018, 38616, 39090, 39356, 39426, 39298, 399150, 395856, 3~
## $ btn_20_29  <int> 67852, 69636, 71273, 72837, 74369, 75539, 845215, 842528, 8~
## $ btn_30_39  <int> 51375, 53139, 54993, 56930, 58927, 60769, 754747, 773227, 7~
## $ btn_40_49  <int> 38610, 39860, 41112, 42337, 43520, 44630, 624526, 628731, 6~
## $ btn_50_59  <int> 26066, 27128, 28178, 29232, 30329, 31417, 545622, 560170, 5~
## $ btn_60_69  <int> 13971, 14741, 15534, 16362, 17229, 18121, 345186, 359612, 3~
## $ btn_70_79  <int> 6679, 6869, 7094, 7363, 7676, 8054, 189960, 197894, 206475,~
## $ btn_80_89  <int> 2686, 2750, 2813, 2873, 2934, 3006, 76042, 79300, 82749, 86~
## $ btn_90_99  <int> 281, 315, 344, 374, 407, 447, 8932, 9502, 10124, 10762, 113~
## $ old_100    <int> 1, 1, 2, 3, 4, 6, 125, 131, 138, 147, 157, 167, 485, 559, 6~
## $ population <int> 359884, 367327, 374702, 382076, 389103, 394931, 4895241, 49~
## $ region     <chr> "América Central", "América Central", "América Central", "A~
```

```
# Visualizamos las variables
names(pra1)
```

```
##  [1] "entity"     "code"       "year"       "gdp_index"  "imf_index"
##  [6] "uhc_index"  "bci_index"  "ilo_index"  "phy_index"  "u_1"
## [11] "u_5"        "u_15"       "u_25"       "btn_15_64"  "old_15"
## [16] "old_18"     "at_1"       "btn_1_4"    "btn_5_9"    "btn_10_14"
## [21] "btn_15_19"  "btn_20_29"  "btn_30_39"  "btn_40_49"  "btn_50_59"
## [26] "btn_60_69"  "btn_70_79"  "btn_80_89"  "btn_90_99"  "old_100"
## [31] "population" "region"
```

```
# Limpieza
 rm(pra1_ordenado, puntos_flotantes, blank_rows, na_rows, numeric_cols)

# Copia del dataframe
 write.csv(pra1, file= "pra1.csv")
```

Imputing missing values in GDP column

As observed earlier, the GDP column has numerous NaN values. We will impute the missing values based on the available data in the variable. To begin with, we will impute the missing values by taking the average of the records from the year and the previous year of the missing entry. Let's start by installing two new libraries that we will need.

```
# Creamos una función para imputar con la media solo si los valores adyacentes no son 0
impute_with_mean <- function(x) {
  na_loc <- is.na(x)
  prev_val <- ifelse(is.na(lag(x)), 0, lag(x))
  next_val <- ifelse(is.na(lead(x)), 0, lead(x))
  x[na_loc & prev_val != 0 & next_val != 0] <- (prev_val + next_val)/2 # si los valores anterior/posteri
  return(x)
}

# Aplicamos la función a cada grupo de 'code'
pra1 <- pra1 %>%
  arrange(code, year) %>% # organizamos por code y year
  group_by(code) %>% # agrupamos por code
  mutate(gdp_index = impute_with_mean(gdp_index)) # imputa por la media si esta disponible segun los pa
```

The code has not covered all cases where there are 'NAs'. But we had retained data from different regions to be able to impute data. We will create regions for the 'gdp_index':

```
# Agregamos una nueva columna que indique la región de cada país
pra1 <- pra1 %>%
  mutate(region_gdp = case_when(
    code %in% eu_countries ~ "EU",
    code %in% other_europe_countries ~ "ACS",
    code %in% africa_countries ~ "LWC",
    (code %in% caribbean_countries | code %in% south_america_countries | code %in% central_america_coun
    code %in% asia_countries ~ "EAP",
    code %in% oceania_countries ~ "LWC",
    code %in% north_america_countries ~ "HIN",
    TRUE ~ NA_character_
  ))

# Filtramos los datos de las regiones
region_data_gdp <- pra1 %>%
  filter(code %in% c("EU", "ACS", "LWC", "LAC", "EAP", "HIN"))

# Unimos los datos de las regiones a los datos de los países
pra1 <- pra1 %>%
  left_join(region_data_gdp, by = c("region_gdp" = "code", "year"), suffix = c("", "_region"))

# Imputamos los valores faltantes de 'gdp_index' utilizando los valores de las regiones
pra1 <- pra1 %>%
  mutate(gdp_index = ifelse(is.na(gdp_index), gdp_index_region, gdp_index))

# renombramos columnas en pra1
pra1 <- pra1 %>%  rename(entity_global_gdp = entity_region)
pra1 <- pra1 %>%  rename(gdp_global = gdp_index_region)

# Eliminamos lo que no necesitamos en la tabla 'pra1':
pra1 <- select(pra1, -matches("_region"))

# Nos quedamos con lo que necesitamos en la tabla 'region_data_gdp':
region_data_gdp <- select(region_data_gdp, entity, code, year, gdp_index)

# visualizamos la tabla pra1
print(pra1)
```

```
## # A tibble: 1,630 x 35
## # Groups:   code [275]
##    entity          code   year gdp_index imf_index uhc_index bci_index ilo_index
##    <chr>           <chr> <int>     <dbl>     <dbl>     <dbl>     <dbl>     <dbl>
##  1 Aruba           ABW    2015     5.13      NaN         NaN       NaN       NaN
##  2 Aruba           ABW    2016     1.59      NaN         NaN       NaN       NaN
##  3 Aruba           ABW    2017     1.52      NaN         NaN       NaN       NaN
##  4 Aruba           ABW    2018     0.686     NaN         NaN       NaN       NaN
##  5 Aruba           ABW    2019    -0.130     NaN         NaN       NaN       NaN
##  6 Aruba           ABW    2020    -7.57      NaN         NaN       NaN       NaN
##  7 Europe and Cen~ ACS    2015     1.63       36.1       NaN       NaN       NaN
##  8 Europe and Cen~ ACS    2016     1.50       35.8       NaN       NaN       NaN
##  9 Europe and Cen~ ACS    2017     2.40       35.3       NaN       NaN       NaN
## 10 Europe and Cen~ ACS    2018     1.79       34.9       NaN       NaN       NaN
## # i 1,620 more rows
## # i 27 more variables: phy_index <dbl>, u_1 <int>, u_5 <int>, u_15 <int>,
## #   u_25 <int>, btn_15_64 <int>, old_15 <int>, old_18 <int>, at_1 <int>,
## #   btn_1_4 <int>, btn_5_9 <int>, btn_10_14 <int>, btn_15_19 <int>,
## #   btn_20_29 <int>, btn_30_39 <int>, btn_40_49 <int>, btn_50_59 <int>,
## #   btn_60_69 <int>, btn_70_79 <int>, btn_80_89 <int>, btn_90_99 <int>,
## #   old_100 <int>, population <int>, region <chr>, region_gdp <chr>, ...
```

```
# visualizamos las variables
names(pra1)
```

```
##  [1] "entity"            "code"              "year"
##  [4] "gdp_index"         "imf_index"         "uhc_index"
##  [7] "bci_index"         "ilo_index"         "phy_index"
## [10] "u_1"               "u_5"               "u_15"
## [13] "u_25"              "btn_15_64"         "old_15"
## [16] "old_18"            "at_1"              "btn_1_4"
## [19] "btn_5_9"           "btn_10_14"         "btn_15_19"
## [22] "btn_20_29"         "btn_30_39"         "btn_40_49"
## [25] "btn_50_59"         "btn_60_69"         "btn_70_79"
## [28] "btn_80_89"         "btn_90_99"         "old_100"
## [31] "population"        "region"            "region_gdp"
## [34] "entity_global_gdp" "gdp_global"
```

```
#visualizamos la tabla region_data_gdp
print(region_data_gdp)
```

```
## # A tibble: 36 x 4
## # Groups:   code [6]
##    entity                  code   year gdp_index
##    <chr>                   <chr> <int>     <dbl>
##  1 Europe and Central Asia ACS    2015      1.63
##  2 Europe and Central Asia ACS    2016      1.50
##  3 Europe and Central Asia ACS    2017      2.40
##  4 Europe and Central Asia ACS    2018      1.79
##  5 Europe and Central Asia ACS    2019      1.50
##  6 Europe and Central Asia ACS    2020     -5.82
##  7 East Asia and Pacific   EAP    2015      3.96
##  8 East Asia and Pacific   EAP    2016      3.93
##  9 East Asia and Pacific   EAP    2017      4.31
## 10 East Asia and Pacific   EAP    2018      4.09
## # i 26 more rows
```

We look at the changes. We have an additional table 'region_data_gdp' which contains the reference values for GDP or GDP. We check the imputation of values in the GDP index:

```
# Comprobamos que los países salvo 'organizaciones y otros' tiene todos los valores asignados en la colu

# Analizamos la nueva columna 'region' en busca de blancos y NA
na_rows <- is.na(pra1$gdp_index) & pra1$region != "Organizaciones y otros"

# Buscamos blank
blank_rows <- (pra1$gdp_index == "" & pra1$region != "Organizaciones y otros")

# Vemos las filas donde 'gdp_index' es NA
print("NAs")
```

```
## [1] "NAs"
```

```r
pra1[na_rows,]
```

```
## # A tibble: 0 x 35
## # Groups:   code [0]
## # i 35 variables: entity <chr>, code <chr>, year <int>, gdp_index <dbl>,
## #   imf_index <dbl>, uhc_index <dbl>, bci_index <dbl>, ilo_index <dbl>,
## #   phy_index <dbl>, u_1 <int>, u_5 <int>, u_15 <int>, u_25 <int>,
## #   btn_15_64 <int>, old_15 <int>, old_18 <int>, at_1 <int>, btn_1_4 <int>,
## #   btn_5_9 <int>, btn_10_14 <int>, btn_15_19 <int>, btn_20_29 <int>,
## #   btn_30_39 <int>, btn_40_49 <int>, btn_50_59 <int>, btn_60_69 <int>,
## #   btn_70_79 <int>, btn_80_89 <int>, btn_90_99 <int>, old_100 <int>, ...
```

```r
# vemos las filas donde 'gdp_index' es blank
print("Blancos")
```

```
## [1] "Blancos"
```

```r
pra1[blank_rows, ]
```

```
## # A tibble: 0 x 35
## # Groups:   code [0]
## # i 35 variables: entity <chr>, code <chr>, year <int>, gdp_index <dbl>,
## #   imf_index <dbl>, uhc_index <dbl>, bci_index <dbl>, ilo_index <dbl>,
## #   phy_index <dbl>, u_1 <int>, u_5 <int>, u_15 <int>, u_25 <int>,
## #   btn_15_64 <int>, old_15 <int>, old_18 <int>, at_1 <int>, btn_1_4 <int>,
## #   btn_5_9 <int>, btn_10_14 <int>, btn_15_19 <int>, btn_20_29 <int>,
## #   btn_30_39 <int>, btn_40_49 <int>, btn_50_59 <int>, btn_60_69 <int>,
## #   btn_70_79 <int>, btn_80_89 <int>, btn_90_99 <int>, old_100 <int>, ...
```

We observe that it returns records of the 'OECD' which, for the moment, we are interested in keeping. But because of the operations, we observe complete records with 'NA' values, which in this case should be cleaned up, but which we will also do later.

Imputing missing values in "imf_index" or government spending index

We will proceed in a similar manner as described before with the GDP. In this case, we are addressing the government spending index. We will impute the missing values with the mean values, taking into account that there may be few cases where the function can operate, as the IMF index is sparse in the dataset.

```r
# # Creamos una función para imputar con la media solo si los valores adyacentes no son 0, la habiamos l

impute_with_mean <- function(x) {
  na_loc <- is.na(x)
  prev_val <- ifelse(is.na(lag(x)), 0, lag(x))
  next_val <- ifelse(is.na(lead(x)), 0, lead(x))
  x[na_loc & prev_val != 0 & next_val != 0] <- (prev_val + next_val)/2 # si los valores anterior/posteri
  return(x)
}
```

```
# Aplicamos la función a cada grupo de 'code'
pra1 <- pra1 %>%
  arrange(code, year) %>% # organizamos por code y year
  group_by(code) %>% # agrupamos por code
  mutate(imf_index = impute_with_mean(imf_index)) # imputa por la media si esta disponible segun los pa

# visualizamos la tabla:
print(pra1)
```

```
## # A tibble: 1,630 x 35
## # Groups:   code [275]
##    entity          code   year gdp_index imf_index uhc_index bci_index ilo_index
##    <chr>           <chr> <int>     <dbl>     <dbl>     <dbl>     <dbl>     <dbl>
##  1 Aruba           ABW    2015     5.13      NaN         NaN       NaN       NaN
##  2 Aruba           ABW    2016     1.59      NaN         NaN       NaN       NaN
##  3 Aruba           ABW    2017     1.52      NaN         NaN       NaN       NaN
##  4 Aruba           ABW    2018     0.686     NaN         NaN       NaN       NaN
##  5 Aruba           ABW    2019    -0.130     NaN         NaN       NaN       NaN
##  6 Aruba           ABW    2020    -7.57      NaN         NaN       NaN       NaN
##  7 Europe and Cen~ ACS    2015     1.63       36.1       NaN       NaN       NaN
##  8 Europe and Cen~ ACS    2016     1.50       35.8       NaN       NaN       NaN
##  9 Europe and Cen~ ACS    2017     2.40       35.3       NaN       NaN       NaN
## 10 Europe and Cen~ ACS    2018     1.79       34.9       NaN       NaN       NaN
## # i 1,620 more rows
## # i 27 more variables: phy_index <dbl>, u_1 <int>, u_5 <int>, u_15 <int>,
## #   u_25 <int>, btn_15_64 <int>, old_15 <int>, old_18 <int>, at_1 <int>,
## #   btn_1_4 <int>, btn_5_9 <int>, btn_10_14 <int>, btn_15_19 <int>,
## #   btn_20_29 <int>, btn_30_39 <int>, btn_40_49 <int>, btn_50_59 <int>,
## #   btn_60_69 <int>, btn_70_79 <int>, btn_80_89 <int>, btn_90_99 <int>,
## #   old_100 <int>, population <int>, region <chr>, region_gdp <chr>, ...
```

We have encountered a warning that we will ignore for now, as it does not affect our current process. We will address the remaining cases of missing values in the ‘imf_index’ column by creating regions to impute the missing data.

```
# Copia del dataframe
write.csv(pra1, file= "pra1.csv")
pra1_2 <- pra1

# Agregamos una nueva columna que indique la región de cada país
pra1 <- pra1 %>%
  mutate(region_imf = case_when(
    code %in% eu_countries ~ "EU",
    code %in% other_europe_countries ~ "ACS",
    code %in% africa_countries ~ "LMC",
    (code %in% caribbean_countries | code %in% south_america_countries | code %in% central_america_coun
    code %in% asia_countries ~ "SAS",
    code %in% oceania_countries ~ "LMC",
    code %in% north_america_countries ~ "HIN",
    TRUE ~ NA_character_
  ))
```

```r
# Filtramos los datos de las regiones
region_data_imf <- pra1 %>%
  filter(code %in% c("EU", "ACS", "LMC", "LAC", "SAS", "HIN"))

# Imputamos los valores faltantes en los datos de las regiones, pues IMF esta incompleto
region_data_imf <- region_data_imf %>%
  group_by(code) %>%
  mutate(imf_index = ifelse(is.na(imf_index), mean(imf_index, na.rm = TRUE), imf_index))

# Unimos los datos de las regiones a los datos de los países
pra1 <- pra1 %>%
  left_join(region_data_imf, by = c("region_imf" = "code", "year"), suffix = c("", "_region"))

# Imputamos los valores faltantes de 'imf_index' utilizando los valores de las regiones
pra1 <- pra1 %>% mutate(imf_index = ifelse(is.na(imf_index), imf_index_region, imf_index))

# renombramos columnas en pra1
pra1 <- pra1 %>%  rename(entity_global_imf = entity_region)
pra1 <- pra1 %>%  rename(imf_global = imf_index_region)

# Eliminamos lo que no necesitamos en la tabla 'pra1':
pra1 <- select(pra1, -matches("_region"))

# Nos quedamos con lo que necesitamos en la tabla 'region_data_imf':
region_data_imf <- select(region_data_imf, entity, code, year, imf_index)

# visualizamos la tabla pra1
print(pra1)
```

```
## # A tibble: 1,630 x 38
## # Groups:   code [275]
##    entity          code   year gdp_index imf_index uhc_index bci_index ilo_index
##    <chr>           <chr> <int>     <dbl>     <dbl>     <dbl>     <dbl>     <dbl>
##  1 Aruba           ABW    2015     5.13       28.5       NaN       NaN       NaN
##  2 Aruba           ABW    2016     1.59       28.5       NaN       NaN       NaN
##  3 Aruba           ABW    2017     1.52       27.9       NaN       NaN       NaN
##  4 Aruba           ABW    2018     0.686      27.7       NaN       NaN       NaN
##  5 Aruba           ABW    2019    -0.130      27.5       NaN       NaN       NaN
##  6 Aruba           ABW    2020    -7.57       28.0       NaN       NaN       NaN
##  7 Europe and Cen~ ACS    2015     1.63       36.1       NaN       NaN       NaN
##  8 Europe and Cen~ ACS    2016     1.50       35.8       NaN       NaN       NaN
##  9 Europe and Cen~ ACS    2017     2.40       35.3       NaN       NaN       NaN
## 10 Europe and Cen~ ACS    2018     1.79       34.9       NaN       NaN       NaN
## # i 1,620 more rows
## # i 30 more variables: phy_index <dbl>, u_1 <int>, u_5 <int>, u_15 <int>,
## #   u_25 <int>, btn_15_64 <int>, old_15 <int>, old_18 <int>, at_1 <int>,
## #   btn_1_4 <int>, btn_5_9 <int>, btn_10_14 <int>, btn_15_19 <int>,
## #   btn_20_29 <int>, btn_30_39 <int>, btn_40_49 <int>, btn_50_59 <int>,
## #   btn_60_69 <int>, btn_70_79 <int>, btn_80_89 <int>, btn_90_99 <int>,
## #   old_100 <int>, population <int>, region <chr>, region_gdp <chr>, ...
```

```r
# visualizamos las variables
names(pra1)
```

```
##  [1] "entity"            "code"              "year"
##  [4] "gdp_index"         "imf_index"         "uhc_index"
##  [7] "bci_index"         "ilo_index"         "phy_index"
## [10] "u_1"               "u_5"               "u_15"
## [13] "u_25"              "btn_15_64"         "old_15"
## [16] "old_18"            "at_1"              "btn_1_4"
## [19] "btn_5_9"           "btn_10_14"         "btn_15_19"
## [22] "btn_20_29"         "btn_30_39"         "btn_40_49"
## [25] "btn_50_59"         "btn_60_69"         "btn_70_79"
## [28] "btn_80_89"         "btn_90_99"         "old_100"
## [31] "population"        "region"            "region_gdp"
## [34] "entity_global_gdp" "gdp_global"        "region_imf"
## [37] "entity_global_imf" "imf_global"
```

```
#visualizamos la tabla region_data_gdp
print(region_data_imf)
```

```
## # A tibble: 36 x 4
## # Groups:   code [6]
##    entity                  code   year imf_index
##    <chr>                   <chr> <int>     <dbl>
##  1 Europe and Central Asia ACS    2015      36.1
##  2 Europe and Central Asia ACS    2016      35.8
##  3 Europe and Central Asia ACS    2017      35.3
##  4 Europe and Central Asia ACS    2018      34.9
##  5 Europe and Central Asia ACS    2019      35.0
##  6 Europe and Central Asia ACS    2020      35.4
##  7 European Union          EU     2015      38.4
##  8 European Union          EU     2016      37.9
##  9 European Union          EU     2017      37.0
## 10 European Union          EU     2018      36.7
## # i 26 more rows
```

It looks like we are now up to the fifth column corrected. Let's proceed with the sixth. The 'UHC' index reflecting compliance and adherence to labour laws:

Imputing missing values in the variable 'UHC' or basic health coverage.

Essential health coverage or essential health services is key to the claims of this company, as the lack of a good health care system would call into question the establishment in a new market.

The 'UHC' index is quite scarce in the registers, only some regions have these values specified, so we will proceed to impute according to the regions and subsequently, we will impute the missing values, so we will reverse the order in comparison with previously imputed indices:

```
# Copia del dataframe
write.csv(pra1, file= "pra1.csv")
pra1_2 <- pra1


# Agregamos una nueva columna que indique la región de cada país y creacion de nueva columna
pra1 <- pra1 %>%
  mutate(region_uhc = case_when(
    code %in% eu_countries ~ "EUR",
    code %in% other_europe_countries ~ "WRL",
    code %in% africa_countries ~ "AFR",
    code %in% caribbean_countries ~ "LWC",
    code %in% central_america_countries ~ "LWC",
    code %in% south_america_countries ~ "LMC",
    code %in% asia_countries ~ "ASI",
    code %in% oceania_countries ~ "OCE",
    code %in% north_america_countries ~ "HIC",
    TRUE ~ "Imputado"
  ))

# Filtramos los datos de las regiones
region_data_uhc <- pra1 %>%
  filter(code %in% c("EUR", "WRL", "AFR", "LWC", "LMC", "ASI", "OCE", "HIC"))

# Imputamos los valores faltantes en los datos de las regiones, pues UHC esta incompleto
region_data_uhc <- region_data_uhc %>%
  group_by(code) %>%
  mutate(uhc_index = ifelse(is.na(uhc_index), mean(uhc_index, na.rm = TRUE), imf_index))

# Unimos los datos de las regiones a los datos de los países
pra1 <- pra1 %>%
  left_join(region_data_uhc, by = c("region_uhc" = "code", "year"), suffix = c("", "_region"))

# Imputamos los valores faltantes de 'uhc_index' utilizando los valores de las regiones
pra1 <- pra1 %>% mutate(uhc_index = ifelse(is.na(uhc_index), uhc_index_region, uhc_index))

# renombramos columnas en pra1
pra1 <- pra1 %>%  rename(entity_global_uhc = entity_region)
pra1 <- pra1 %>%  rename(uhc_global = uhc_index_region)

# Eliminamos lo que no necesitamos en la tabla 'pra1':
pra1 <- select(pra1, -matches("_region"))

# Nos quedamos con lo que necesitamos en la tabla 'region_data_uhc':
region_data_uhc <- select(region_data_uhc, entity, code, year, uhc_index)

# visualizamos la tabla pra1
print(pra1)
```

```
## # A tibble: 1,630 x 41
## # Groups:   code [275]
##    entity           code   year gdp_index imf_index uhc_index bci_index ilo_index
##    <chr>            <chr> <int>     <dbl>     <dbl>     <dbl>     <dbl>     <dbl>
##  1 Aruba            ABW    2015     5.13       28.5      NA         NaN       NaN
##  2 Aruba            ABW    2016     1.59       28.5      40.8       NaN       NaN
##  3 Aruba            ABW    2017     1.52       27.9      NA         NaN       NaN
##  4 Aruba            ABW    2018     0.686      27.7      40.8       NaN       NaN
##  5 Aruba            ABW    2019    -0.130      27.5      NA         NaN       NaN
##  6 Aruba            ABW    2020    -7.57       28.0      40.8       NaN       NaN
##  7 Europe and Cen~ ACS    2015     1.63       36.1      NA         NaN       NaN
##  8 Europe and Cen~ ACS    2016     1.50       35.8      NA         NaN       NaN
##  9 Europe and Cen~ ACS    2017     2.40       35.3      NA          NaN       NaN
## 10 Europe and Cen~ ACS    2018     1.79       34.9      NA          NaN       NaN
## # i 1,620 more rows
## # i 33 more variables: phy_index <dbl>, u_1 <int>, u_5 <int>, u_15 <int>,
## #   u_25 <int>, btn_15_64 <int>, old_15 <int>, old_18 <int>, at_1 <int>,
## #   btn_1_4 <int>, btn_5_9 <int>, btn_10_14 <int>, btn_15_19 <int>,
## #   btn_20_29 <int>, btn_30_39 <int>, btn_40_49 <int>, btn_50_59 <int>,
## #   btn_60_69 <int>, btn_70_79 <int>, btn_80_89 <int>, btn_90_99 <int>,
## #   old_100 <int>, population <int>, region <chr>, region_gdp <chr>, ...
```

```
# visualizamos las variables
names(pra1)
```

```
##  [1] "entity"            "code"              "year"
##  [4] "gdp_index"         "imf_index"         "uhc_index"
##  [7] "bci_index"         "ilo_index"         "phy_index"
## [10] "u_1"               "u_5"               "u_15"
## [13] "u_25"              "btn_15_64"         "old_15"
## [16] "old_18"            "at_1"              "btn_1_4"
## [19] "btn_5_9"           "btn_10_14"         "btn_15_19"
## [22] "btn_20_29"         "btn_30_39"         "btn_40_49"
## [25] "btn_50_59"         "btn_60_69"         "btn_70_79"
## [28] "btn_80_89"         "btn_90_99"         "old_100"
## [31] "population"        "region"            "region_gdp"
## [34] "entity_global_gdp" "gdp_global"        "region_imf"
## [37] "entity_global_imf" "imf_global"        "region_uhc"
## [40] "entity_global_uhc" "uhc_global"
```

```
#visualizamos la tabla region_data_gdp
print(region_data_uhc)
```

```
## # A tibble: 48 x 4
## # Groups:   code [8]
##    entity      code   year uhc_index
##    <chr>       <chr> <int>     <dbl>
##  1 Africa      AFR    2015      NA
##  2 Africa (UN) AFR    2016      47.0
##  3 Africa      AFR    2017      NA
##  4 Africa (UN) AFR    2018      47.0
##  5 Africa      AFR    2019      NA
##  6 Africa (UN) AFR    2020      47.0
##  7 Asia        ASI    2015      NA
##  8 Asia (UN)   ASI    2016      65.9
##  9 Asia        ASI    2017      NA
## 10 Asia (UN)   ASI    2018      65.9
## # i 38 more rows
```

The operations, in general, seem to have been correct but 'NAs' remain in several columns. Let us impute values to the mean:

```
# Modificamos la función para imputar con la media solo si los valores adyacentes no son NA, y luego co
```

```
impute_with_mean <- function(x) {
  # Consideramos los valores exactamente 0 como NA
  x[x == 0] <- NA

  na_loc <- is.na(x)
  prev_val <- lag(x)
  next_val <- lead(x)
  x[na_loc] <- (prev_val + next_val)/2 # si los valores anterior/posterior son no NA

  # Si aún quedan NA, imputamos con la media de todos los valores no NA
  if(any(is.na(x))){
    x[is.na(x)] <- mean(x, na.rm = TRUE)
  }
  return(x)
}

# Aplicamos la función a cada grupo de 'code'
pra1 <- pra1 %>%
  arrange(code, year) %>% # organizamos por code y year
  group_by(code) %>% # agrupamos por code
  mutate(uhc_index = impute_with_mean(uhc_index)) # imputa por la media si esta disponible segun los pa

# Aplicamos la función a cada grupo de 'code' en region_data_uhc
region_data_uhc <- region_data_uhc %>%
  arrange(code, year) %>% # organizamos por code y year
  group_by(code) %>% # agrupamos por code
  mutate(uhc_index = impute_with_mean(uhc_index)) # imputa por la media si esta disponible segun los pa


# visualizamos la tabla:
print(pra1)
```

```
## # A tibble: 1,630 x 41
## # Groups:   code [275]
##    entity          code   year gdp_index imf_index uhc_index bci_index ilo_index
##    <chr>           <chr> <int>     <dbl>     <dbl>     <dbl>     <dbl>     <dbl>
##  1 Aruba           ABW    2015     5.13       28.5      40.8       NaN       NaN
##  2 Aruba           ABW    2016     1.59       28.5      40.8       NaN       NaN
##  3 Aruba           ABW    2017     1.52       27.9      40.8       NaN       NaN
##  4 Aruba           ABW    2018     0.686      27.7      40.8       NaN       NaN
##  5 Aruba           ABW    2019    -0.130      27.5      40.8       NaN       NaN
##  6 Aruba           ABW    2020    -7.57       28.0      40.8       NaN       NaN
##  7 Europe and Cen~ ACS    2015     1.63       36.1     NaN         NaN       NaN
##  8 Europe and Cen~ ACS    2016     1.50       35.8     NaN         NaN       NaN
##  9 Europe and Cen~ ACS    2017     2.40       35.3     NaN         NaN       NaN
## 10 Europe and Cen~ ACS    2018     1.79       34.9     NaN         NaN       NaN
## # i 1,620 more rows
## # i 33 more variables: phy_index <dbl>, u_1 <int>, u_5 <int>, u_15 <int>,
## #   u_25 <int>, btn_15_64 <int>, old_15 <int>, old_18 <int>, at_1 <int>,
## #   btn_1_4 <int>, btn_5_9 <int>, btn_10_14 <int>, btn_15_19 <int>,
## #   btn_20_29 <int>, btn_30_39 <int>, btn_40_49 <int>, btn_50_59 <int>,
## #   btn_60_69 <int>, btn_70_79 <int>, btn_80_89 <int>, btn_90_99 <int>,
## #   old_100 <int>, population <int>, region <chr>, region_gdp <chr>, ...
```

Have had quite a few imputations due to averaging. Some columns derived from 'uhc_index' such as 'uhc_global' are not going to be needed, so we did some clean-up:

```
# Limpiamos las columnas innecesarias en este caso de UHC

pra1 <- subset(pra1, select = -uhc_global )
```

We have had quite a few imputations for the average. We proceed with the next column.

Imputing missing values in the column 'bci_index' or business confidence index.

The business confidence index can be an indicator of whether the right conditions exist to invest in a country.

```
# Aplicamos la función a cada grupo de 'code'
pra1 <- pra1 %>%
  arrange(code, year) %>% # organizamos por code y year
  group_by(code) %>% # agrupamos por code
  mutate(bci_index = impute_with_mean(bci_index)) # imputa por la media si esta disponible segun los pa

# visualizamos la tabla:
print(pra1)
```

```
## # A tibble: 1,630 x 40
## # Groups:   code [275]
##    entity          code   year gdp_index imf_index uhc_index bci_index ilo_index
##    <chr>           <chr> <int>     <dbl>     <dbl>     <dbl>     <dbl>     <dbl>
##  1 Aruba           ABW    2015     5.13       28.5      40.8       NaN       NaN
##  2 Aruba           ABW    2016     1.59       28.5      40.8       NaN       NaN
##  3 Aruba           ABW    2017     1.52       27.9      40.8       NaN       NaN
##  4 Aruba           ABW    2018     0.686      27.7      40.8       NaN       NaN
##  5 Aruba           ABW    2019    -0.130      27.5      40.8       NaN       NaN
##  6 Aruba           ABW    2020    -7.57       28.0      40.8       NaN       NaN
##  7 Europe and Cen~ ACS    2015     1.63       36.1     NaN         NaN       NaN
##  8 Europe and Cen~ ACS    2016     1.50       35.8     NaN         NaN       NaN
##  9 Europe and Cen~ ACS    2017     2.40       35.3     NaN         NaN       NaN
## 10 Europe and Cen~ ACS    2018     1.79       34.9     NaN         NaN       NaN
## # i 1,620 more rows
## # i 32 more variables: phy_index <dbl>, u_1 <int>, u_5 <int>, u_15 <int>,
## #   u_25 <int>, btn_15_64 <int>, old_15 <int>, old_18 <int>, at_1 <int>,
## #   btn_1_4 <int>, btn_5_9 <int>, btn_10_14 <int>, btn_15_19 <int>,
## #   btn_20_29 <int>, btn_30_39 <int>, btn_40_49 <int>, btn_50_59 <int>,
## #   btn_60_69 <int>, btn_70_79 <int>, btn_80_89 <int>, btn_90_99 <int>,
## #   old_100 <int>, population <int>, region <chr>, region_gdp <chr>, ...
```

The business confidence index is a very scarce commodity as can be seen, we will complete the index by turning once again to the broader territorial and supra-organisations.

```
# Copia del dataframe
write.csv(pra1, file= "pra1.csv")
pra1_2 <- pra1

# Creamos un nuevo grupo de paises de la OCDE:
```

```
ocde_countries <- unique(c("AUS", "AUT", "BEL", "CAN", "CHL", "COL", "CZE", "DNK", "EST", "FIN", "FRA",

# Agregamos una nueva columna que indique la región de cada país
pra1 <- pra1 %>%
  mutate(region_bci = case_when(
    code %in% eu_countries ~ "E20",
    code %in% ocde_countries ~ "OCD",
    TRUE ~ "No OCDE"
  ))

# Filtramos los datos de las regiones
region_data_bci <- pra1 %>%
  filter(code %in% c("E20", "OCD"))

# Imputamos los valores faltantes en los datos de las regiones, pues BCI esta incompleto
region_data_bci <- region_data_bci %>%
  group_by(code) %>%
  mutate(bci_index = ifelse(is.na(bci_index), mean(bci_index, na.rm = TRUE), bci_index))

# Unimos los datos de las regiones a los datos de los países
pra1 <- pra1 %>%
  left_join(region_data_bci, by = c("region_bci" = "code", "year"), suffix = c("", "_region"))

# Imputamos los valores faltantes de 'bci_index' utilizando los valores de las regiones
pra1 <- pra1 %>% mutate(bci_index = ifelse(is.na(bci_index), bci_index_region, bci_index))

# Asigna 45.0 a los países que no están en ocde_countries o eu_countries
pra1 <- pra1 %>% mutate(bci_index = ifelse(is.na(bci_index), 45.0, bci_index))

# renombramos columnas en pra1
pra1 <- pra1 %>%  rename(entity_global_bci = entity_region)
pra1 <- pra1 %>%  rename(bci_global = bci_index_region)

# Eliminamos lo que no necesitamos en la tabla 'pra1':
pra1 <- select(pra1, -matches("_region"))

# Nos quedamos con lo que necesitamos en la tabla 'region_data_bci':
region_data_bci <- select(region_data_bci, entity, code, year, bci_index)

# visualizamos la tabla pra1
print(pra1)
```

```
## # A tibble: 1,630 x 43
## # Groups:   code [275]
##    entity          code   year gdp_index imf_index uhc_index bci_index ilo_index
##    <chr>           <chr> <int>     <dbl>     <dbl>     <dbl>     <dbl>     <dbl>
##  1 Aruba           ABW    2015     5.13       28.5      40.8        45       NaN
##  2 Aruba           ABW    2016     1.59       28.5      40.8        45       NaN
##  3 Aruba           ABW    2017     1.52       27.9      40.8        45       NaN
##  4 Aruba           ABW    2018     0.686      27.7      40.8        45       NaN
##  5 Aruba           ABW    2019    -0.130      27.5      40.8        45       NaN
##  6 Aruba           ABW    2020    -7.57       28.0      40.8        45       NaN
##  7 Europe and Cen~ ACS    2015     1.63       36.1     NaN          45       NaN
##  8 Europe and Cen~ ACS    2016     1.50       35.8     NaN          45       NaN
##  9 Europe and Cen~ ACS    2017     2.40       35.3     NaN          45       NaN
## 10 Europe and Cen~ ACS    2018     1.79       34.9     NaN          45       NaN
## # i 1,620 more rows
## # i 35 more variables: phy_index <dbl>, u_1 <int>, u_5 <int>, u_15 <int>,
## #   u_25 <int>, btn_15_64 <int>, old_15 <int>, old_18 <int>, at_1 <int>,
## #   btn_1_4 <int>, btn_5_9 <int>, btn_10_14 <int>, btn_15_19 <int>,
## #   btn_20_29 <int>, btn_30_39 <int>, btn_40_49 <int>, btn_50_59 <int>,
## #   btn_60_69 <int>, btn_70_79 <int>, btn_80_89 <int>, btn_90_99 <int>,
## #   old_100 <int>, population <int>, region <chr>, region_gdp <chr>, ...
```

```
# visualizamos las variables
names(pra1)
```

```
##  [1] "entity"            "code"              "year"
##  [4] "gdp_index"         "imf_index"         "uhc_index"
##  [7] "bci_index"         "ilo_index"         "phy_index"
## [10] "u_1"               "u_5"               "u_15"
## [13] "u_25"              "btn_15_64"         "old_15"
## [16] "old_18"            "at_1"              "btn_1_4"
## [19] "btn_5_9"           "btn_10_14"         "btn_15_19"
## [22] "btn_20_29"         "btn_30_39"         "btn_40_49"
## [25] "btn_50_59"         "btn_60_69"         "btn_70_79"
## [28] "btn_80_89"         "btn_90_99"         "old_100"
## [31] "population"        "region"            "region_gdp"
## [34] "entity_global_gdp" "gdp_global"        "region_imf"
## [37] "entity_global_imf" "imf_global"        "region_uhc"
## [40] "entity_global_uhc" "region_bci"        "entity_global_bci"
## [43] "bci_global"
```

```
#visualizamos la tabla region_data_bci
print(region_data_bci)
```

```
## # A tibble: 6 x 4
## # Groups:   code [1]
##   entity    code   year bci_index
##   <chr>     <chr> <int>     <dbl>
## 1 EU27_2020 E20    2015     101.
## 2 EU27_2020 E20    2016     101.
## 3 EU27_2020 E20    2017     102.
## 4 EU27_2020 E20    2018     102.
## 5 EU27_2020 E20    2019     100.
## 6 EU27_2020 E20    2020      98.2
```

Countries for which there is no BCI index would be considered by default under a ‘negative outlook’ according to the BCI index itself, as nothing is as negative as a lack of business information.

We proceed to remove unnecessary columns for the BCI index:

```
# Limpiamos las columnas innecesarias de la tabla pra1

pra1 <- subset(pra1, select = -bci_global)

pra1 <- subset(pra1, select = -entity_global_bci)
```

Proceeding with the next variable.

Imputing missing values in the 'ilo_index' column regarding respect for labor rights.

The index of respect for labor rights and implementation of labor laws can be a good indicator of the quality of employees' working life, their level of commitment to their performance, and their sense of belonging to the group. Therefore, it is important to consider it when selecting a country where the company can expand its operations in a sustainable manner.

```
# Creamos una función para imputar con la media solo si los valores adyacentes no son 0
impute_with_mean <- function(x) {
  na_loc <- is.na(x)
  prev_val <- ifelse(is.na(lag(x)), 0, lag(x))
  next_val <- ifelse(is.na(lead(x)), 0, lead(x))
  x[na_loc & prev_val != 0 & next_val != 0] <- (prev_val + next_val)/2 # si los valores anterior/posteri
  return(x)
}

# Aplicamos la función a cada grupo de 'code'
pra1 <- pra1 %>%
  arrange(code, year) %>% # organizamos por code y year
  group_by(code) %>% # agrupamos por code
  mutate(ilo_index = impute_with_mean(ilo_index)) # imputa por la media si esta disponible segun los par

# Analizamos la nueva columna 'region' en busca de blancos y NA
na_rows <- is.na(pra1$ilo_index) & pra1$region != "Organizaciones y otros"

# Buscamos blank
blank_rows <- (pra1$ilo_index == "" & pra1$region != "Organizaciones y otros")

# Vemos las filas donde 'ilo_index' es NA
print("NAs")
```

```
## [1] "NAs"
```

```
pra1[na_rows,]
```

```
## # A tibble: 638 x 41
## # Groups:   code [112]
##    entity      code   year gdp_index imf_index uhc_index bci_index ilo_index
##    <chr>       <chr> <int>     <dbl>     <dbl>     <dbl>     <dbl>     <dbl>
##  1 Aruba       ABW    2015    5.13        28.5      40.8        45       NaN
##  2 Aruba       ABW    2016    1.59        28.5      40.8        45       NaN
##  3 Aruba       ABW    2017    1.52        27.9      40.8        45       NaN
##  4 Aruba       ABW    2018    0.686       27.7      40.8        45       NaN
##  5 Aruba       ABW    2019   -0.130       27.5      40.8        45       NaN
##  6 Aruba       ABW    2020   -7.57        28.0      40.8        45       NaN
##  7 Afghanistan AFG    2015   -1.62        37.0      32.2        45       NaN
##  8 Afghanistan AFG    2016   -0.541       43.9      65.9        45       NaN
##  9 Afghanistan AFG    2017    0.0648      39.2      35.8        45       NaN
## 10 Afghanistan AFG    2018   -1.19        15.8      65.9        45       NaN
## # i 628 more rows
## # i 33 more variables: phy_index <dbl>, u_1 <int>, u_5 <int>, u_15 <int>,
## #   u_25 <int>, btn_15_64 <int>, old_15 <int>, old_18 <int>, at_1 <int>,
## #   btn_1_4 <int>, btn_5_9 <int>, btn_10_14 <int>, btn_15_19 <int>,
## #   btn_20_29 <int>, btn_30_39 <int>, btn_40_49 <int>, btn_50_59 <int>,
## #   btn_60_69 <int>, btn_70_79 <int>, btn_80_89 <int>, btn_90_99 <int>,
## #   old_100 <int>, population <int>, region <chr>, region_gdp <chr>, ...
```

```
# vemos las filas donde 'ilo_index' es blank
print("Blancos")
```

```
## [1] "Blancos"
```

```
pra1[blank_rows, ]
```

```
## # A tibble: 0 x 41
## # Groups:   code [0]
## # i 41 variables: entity <chr>, code <chr>, year <int>, gdp_index <dbl>,
## #   imf_index <dbl>, uhc_index <dbl>, bci_index <dbl>, ilo_index <dbl>,
## #   phy_index <dbl>, u_1 <int>, u_5 <int>, u_15 <int>, u_25 <int>,
## #   btn_15_64 <int>, old_15 <int>, old_18 <int>, at_1 <int>, btn_1_4 <int>,
## #   btn_5_9 <int>, btn_10_14 <int>, btn_15_19 <int>, btn_20_29 <int>,
## #   btn_30_39 <int>, btn_40_49 <int>, btn_50_59 <int>, btn_60_69 <int>,
## #   btn_70_79 <int>, btn_80_89 <int>, btn_90_99 <int>, old_100 <int>, ...
```

We still have quite a few countries for which we have yet to impute a value to ‘ilo_index’:

```
# Agregamos una nueva columna que indique la región de cada país
pra1 <- pra1 %>%
  mutate(region_ilo = case_when(
    code %in% eu_countries ~ "ENA",
    code %in% other_europe_countries ~ "EUR",
    code %in% africa_countries ~ "NAF",
    code %in% caribbean_countries ~ "SID",
    code %in% south_america_countries ~ "LAC",
    code %in% central_america_countries ~ "LDV",
    code %in% asia_countries ~ "CSA",
    code %in% oceania_countries ~ "OCE",
    code %in% north_america_countries ~ "ENA",
    TRUE ~ "Otros"
  ))

# Filtramos los datos de las regiones
region_data_ilo <- pra1 %>%
  filter(code %in% c("ENA", "EUR", "NAF", "SID", "LAC", "LDV", "CSA", "OCE", "ENA"))

# Unimos los datos de las regiones a los datos de los países
pra1 <- pra1 %>%
  left_join(region_data_ilo, by = c("region_ilo" = "code", "year"), suffix = c("", "_region"))

# Imputamos los valores faltantes de 'ilo_index' utilizando los valores de las regiones
pra1 <- pra1 %>%
```

```r
  mutate(ilo_index = ifelse(is.na(ilo_index), ilo_index_region, ilo_index))

# renombramos columnas en pra1
pra1 <- pra1 %>%  rename(entity_global_ilo = entity_region)
pra1 <- pra1 %>%  rename(ilo_global = ilo_index_region)

# Eliminamos lo que no necesitamos en la tabla 'pra1':
pra1 <- select(pra1, -matches("_region"))

# Nos quedamos con lo que necesitamos en la tabla 'region_data_gdp':
region_data_ilo <- select(region_data_ilo, entity, code, year, ilo_index)

# visualizamos la tabla pra1
print(pra1)
```

```
## # A tibble: 1,630 x 44
## # Groups:   code [275]
##    entity          code   year gdp_index imf_index uhc_index bci_index ilo_index
##    <chr>           <chr> <int>     <dbl>     <dbl>     <dbl>     <dbl>     <dbl>
##  1 Aruba           ABW    2015     5.13       28.5      40.8        45      3.74
##  2 Aruba           ABW    2016     1.59       28.5      40.8        45      3.67
##  3 Aruba           ABW    2017     1.52       27.9      40.8        45      3.63
##  4 Aruba           ABW    2018     0.686      27.7      40.8        45      3.59
##  5 Aruba           ABW    2019    -0.130      27.5      40.8        45      3.62
##  6 Aruba           ABW    2020    -7.57       28.0      40.8        45      3.55
##  7 Europe and Cen~ ACS    2015     1.63       36.1     NaN          45     NA
##  8 Europe and Cen~ ACS    2016     1.50       35.8     NaN          45     NA
##  9 Europe and Cen~ ACS    2017     2.40       35.3     NaN          45     NA
## 10 Europe and Cen~ ACS    2018     1.79       34.9     NaN          45     NA
## # i 1,620 more rows
## # i 36 more variables: phy_index <dbl>, u_1 <int>, u_5 <int>, u_15 <int>,
## #   u_25 <int>, btn_15_64 <int>, old_15 <int>, old_18 <int>, at_1 <int>,
## #   btn_1_4 <int>, btn_5_9 <int>, btn_10_14 <int>, btn_15_19 <int>,
## #   btn_20_29 <int>, btn_30_39 <int>, btn_40_49 <int>, btn_50_59 <int>,
## #   btn_60_69 <int>, btn_70_79 <int>, btn_80_89 <int>, btn_90_99 <int>,
## #   old_100 <int>, population <int>, region <chr>, region_gdp <chr>, ...
```

```r
# visualizamos las variables
names(pra1)
```

```
##  [1] "entity"            "code"              "year"
##  [4] "gdp_index"         "imf_index"         "uhc_index"
##  [7] "bci_index"         "ilo_index"         "phy_index"
## [10] "u_1"               "u_5"               "u_15"
## [13] "u_25"              "btn_15_64"         "old_15"
## [16] "old_18"            "at_1"              "btn_1_4"
## [19] "btn_5_9"           "btn_10_14"         "btn_15_19"
## [22] "btn_20_29"         "btn_30_39"         "btn_40_49"
## [25] "btn_50_59"         "btn_60_69"         "btn_70_79"
## [28] "btn_80_89"         "btn_90_99"         "old_100"
## [31] "population"        "region"            "region_gdp"
## [34] "entity_global_gdp" "gdp_global"        "region_imf"
## [37] "entity_global_imf" "imf_global"        "region_uhc"
## [40] "entity_global_uhc" "region_bci"        "region_ilo"
## [43] "entity_global_ilo" "ilo_global"
```

```
#visualizamos la tabla region_data_ilo
print(region_data_ilo)
```

```
## # A tibble: 48 x 4
## # Groups:   code [8]
##    entity                           code   year ilo_index
##    <chr>                            <chr> <int>     <dbl>
##  1 Central and Southern Asia (UN)   CSA    2015      5.18
##  2 Central and Southern Asia (UN)   CSA    2016      5.5
##  3 Central and Southern Asia (UN)   CSA    2017      5.41
##  4 Central and Southern Asia (UN)   CSA    2018      5.31
##  5 Central and Southern Asia (UN)   CSA    2019      4.67
##  6 Central and Southern Asia (UN)   CSA    2020      4.35
##  7 Europe and Northern America (UN) ENA    2015      1.62
##  8 Europe and Northern America (UN) ENA    2016      1.61
##  9 Europe and Northern America (UN) ENA    2017      1.63
## 10 Europe and Northern America (UN) ENA    2018      1.62
## # i 38 more rows
```

Pending final checks, we will consider it valid and proceed with the next and final variable that requires imputations:

Imputing values in the 'phy_index' index of healthcare personnel density per thousand inhabitants.

The 'phy_index' index refers to the density of healthcare personnel in the population per thousand inhabitants. We will proceed with imputations using the mean of the missing values:

```
# Aplicamos la función previamente creada para aplicar a cada grupo de 'code'
pra1 <- pra1 %>%
  arrange(code, year) %>% # organizamos por code y year
  group_by(code) %>% # agrupamos por code
  mutate(phy_index = impute_with_mean(phy_index)) # imputa por la media si esta disponible segun los pa

# Analizamos la nueva columna 'region' en busca de blancos y NA
na_rows <- is.na(pra1$phy_index) & pra1$region != "Organizaciones y otros"

# Buscamos blank
blank_rows <- (pra1$phy_index == "" & pra1$region != "Organizaciones y otros")

# Vemos las filas donde 'phy_index' es NA
print("NAs")
```

```
## [1] "NAs"
```

```
pra1[na_rows,]
```

```
## # A tibble: 926 x 44
## # Groups:   code [236]
##    entity      code   year gdp_index imf_index uhc_index bci_index ilo_index
##    <chr>       <chr> <int>     <dbl>     <dbl>     <dbl>     <dbl>     <dbl>
##  1 Aruba       ABW    2015    5.13        28.5      40.8        45      3.74
##  2 Aruba       ABW    2016    1.59        28.5      40.8        45      3.67
##  3 Aruba       ABW    2017    1.52        27.9      40.8        45      3.63
##  4 Aruba       ABW    2018    0.686       27.7      40.8        45      3.59
##  5 Aruba       ABW    2019   -0.130       27.5      40.8        45      3.62
##  6 Aruba       ABW    2020   -7.57        28.0      40.8        45      3.55
##  7 Afghanistan AFG    2017    0.0648      39.2      35.8        45      5.41
##  8 Afghanistan AFG    2018   -1.19        15.8      65.9        45      5.31
##  9 Afghanistan AFG    2019    1.54        15.4      37.3        45      4.67
## 10 Afghanistan AFG    2020   -4.58        15.4      65.9        45      4.35
## # i 916 more rows
## # i 36 more variables: phy_index <dbl>, u_1 <int>, u_5 <int>, u_15 <int>,
## #   u_25 <int>, btn_15_64 <int>, old_15 <int>, old_18 <int>, at_1 <int>,
## #   btn_1_4 <int>, btn_5_9 <int>, btn_10_14 <int>, btn_15_19 <int>,
## #   btn_20_29 <int>, btn_30_39 <int>, btn_40_49 <int>, btn_50_59 <int>,
## #   btn_60_69 <int>, btn_70_79 <int>, btn_80_89 <int>, btn_90_99 <int>,
## #   old_100 <int>, population <int>, region <chr>, region_gdp <chr>, ...
```

```r
# vemos las filas donde 'phy_index' es blank
print("Blancos")
```

```
## [1] "Blancos"
```

```r
pra1[blank_rows, ]
```

```
## # A tibble: 0 x 44
## # Groups:   code [0]
## # i 44 variables: entity <chr>, code <chr>, year <int>, gdp_index <dbl>,
## #   imf_index <dbl>, uhc_index <dbl>, bci_index <dbl>, ilo_index <dbl>,
## #   phy_index <dbl>, u_1 <int>, u_5 <int>, u_15 <int>, u_25 <int>,
## #   btn_15_64 <int>, old_15 <int>, old_18 <int>, at_1 <int>, btn_1_4 <int>,
## #   btn_5_9 <int>, btn_10_14 <int>, btn_15_19 <int>, btn_20_29 <int>,
## #   btn_30_39 <int>, btn_40_49 <int>, btn_50_59 <int>, btn_60_69 <int>,
## #   btn_70_79 <int>, btn_80_89 <int>, btn_90_99 <int>, old_100 <int>, ...
```

The problem arises because in this case we only have data for a single year. Therefore, using the function, we will fill in the missing values. We will have to consider later the limitations of this index.

```r
library(tidyverse)

# Propagamos el ultimo valor que no sea NA hacia adelante para cada grupo de 'code'
pra1 <- pra1 %>%
  arrange(code, year) %>% # Organizamos por code y year
  group_by(code) %>% # Agrupamos por code
  fill(phy_index, .direction = "down") # Propagamos el valor hacia adelante

# Propagamos el ultimo valor  hacia atras para cada grupo de 'code'
pra1 <- pra1 %>%
  arrange(code, year) %>% # Organizamos por code y year
  group_by(code) %>% # Agrupamos por code
  fill(phy_index, .direction = "up") # Propagamos el valor hacia atras

# Visualizamos la tabla
print(pra1)
```

```
## # A tibble: 1,630 x 44
## # Groups:   code [275]
##    entity          code   year gdp_index imf_index uhc_index bci_index ilo_index
##    <chr>           <chr> <int>     <dbl>     <dbl>     <dbl>     <dbl>     <dbl>
##  1 Aruba           ABW    2015     5.13       28.5      40.8        45      3.74
##  2 Aruba           ABW    2016     1.59       28.5      40.8        45      3.67
##  3 Aruba           ABW    2017     1.52       27.9      40.8        45      3.63
##  4 Aruba           ABW    2018     0.686      27.7      40.8        45      3.59
##  5 Aruba           ABW    2019    -0.130      27.5      40.8        45      3.62
##  6 Aruba           ABW    2020    -7.57       28.0      40.8        45      3.55
##  7 Europe and Cen~ ACS    2015     1.63       36.1     NaN          45     NA
##  8 Europe and Cen~ ACS    2016     1.50       35.8     NaN          45     NA
##  9 Europe and Cen~ ACS    2017     2.40       35.3     NaN          45     NA
## 10 Europe and Cen~ ACS    2018     1.79       34.9     NaN          45     NA
## # i 1,620 more rows
## # i 36 more variables: phy_index <dbl>, u_1 <int>, u_5 <int>, u_15 <int>,
## #   u_25 <int>, btn_15_64 <int>, old_15 <int>, old_18 <int>, at_1 <int>,
## #   btn_1_4 <int>, btn_5_9 <int>, btn_10_14 <int>, btn_15_19 <int>,
## #   btn_20_29 <int>, btn_30_39 <int>, btn_40_49 <int>, btn_50_59 <int>,
## #   btn_60_69 <int>, btn_70_79 <int>, btn_80_89 <int>, btn_90_99 <int>,
## #   old_100 <int>, population <int>, region <chr>, region_gdp <chr>, ...
```

The changes have been made correctly. Now we will impute the data according to the regions of each country:

```
# Agregamos una nueva columna que indique la región de cada país
pra1 <- pra1 %>%
  mutate(region_phy = case_when(
    code %in% eu_countries ~ "EU",
    code %in% other_europe_countries ~ "ACS",
    code %in% africa_countries ~ "LMC",
    code %in% caribbean_countries ~ "LWC",
    code %in% south_america_countries ~ "LMC",
    code %in% central_america_countries ~ "LWC",
    code %in% asia_countries ~ "EAP",
    code %in% oceania_countries ~ "U-I",
    code %in% north_america_countries ~ "HIN",
    TRUE ~ "Otros"
  ))

# Filtramos los datos de las regiones
region_data_phy <- pra1 %>%
  filter(code %in% c("EU", "ACS", "LMC", "LWC", "EAP", "U-I", "HIN"))

# Unimos los datos de las regiones a los datos de los países
pra1 <- pra1 %>%
  left_join(region_data_phy, by = c("region_phy" = "code", "year"), suffix = c("", "_region"))
```

```r
# Imputamos los valores faltantes de 'phy_index' utilizando los valores de las regiones
pra1 <- pra1 %>%
  mutate(phy_index = ifelse(is.na(phy_index), phy_index_region, phy_index))

# renombramos columnas en pra1
pra1 <- pra1 %>%  rename(entity_global_phy = entity_region)
pra1 <- pra1 %>%  rename(phy_global = phy_index_region)

# Eliminamos lo que no necesitamos en la tabla 'pra1':
pra1 <- select(pra1, -matches("_region"))

# Nos quedamos con lo que necesitamos en la tabla 'region_data_gdp':
region_data_phy <- select(region_data_phy, entity, code, year, phy_index)

# visualizamos la tabla pra1
print(pra1)
```

```
## # A tibble: 1,630 x 47
## # Groups:   code [275]
##    entity           code   year gdp_index imf_index uhc_index bci_index ilo_index
##    <chr>            <chr> <int>     <dbl>     <dbl>     <dbl>     <dbl>     <dbl>
##  1 Aruba            ABW    2015     5.13       28.5      40.8        45      3.74
##  2 Aruba            ABW    2016     1.59       28.5      40.8        45      3.67
##  3 Aruba            ABW    2017     1.52       27.9      40.8        45      3.63
##  4 Aruba            ABW    2018     0.686      27.7      40.8        45      3.59
##  5 Aruba            ABW    2019    -0.130      27.5      40.8        45      3.62
##  6 Aruba            ABW    2020    -7.57       28.0      40.8        45      3.55
##  7 Europe and Cen~ ACS    2015     1.63       36.1     NaN          45     NA
##  8 Europe and Cen~ ACS    2016     1.50       35.8     NaN          45     NA
##  9 Europe and Cen~ ACS    2017     2.40       35.3     NaN          45     NA
## 10 Europe and Cen~ ACS    2018     1.79       34.9     NaN          45     NA
## # i 1,620 more rows
## # i 39 more variables: phy_index <dbl>, u_1 <int>, u_5 <int>, u_15 <int>,
## #   u_25 <int>, btn_15_64 <int>, old_15 <int>, old_18 <int>, at_1 <int>,
## #   btn_1_4 <int>, btn_5_9 <int>, btn_10_14 <int>, btn_15_19 <int>,
## #   btn_20_29 <int>, btn_30_39 <int>, btn_40_49 <int>, btn_50_59 <int>,
## #   btn_60_69 <int>, btn_70_79 <int>, btn_80_89 <int>, btn_90_99 <int>,
## #   old_100 <int>, population <int>, region <chr>, region_gdp <chr>, ...
```

```r
# visualizamos las variables
names(pra1)
```

```
##  [1] "entity"            "code"              "year"
##  [4] "gdp_index"         "imf_index"         "uhc_index"
##  [7] "bci_index"         "ilo_index"         "phy_index"
## [10] "u_1"               "u_5"               "u_15"
## [13] "u_25"              "btn_15_64"         "old_15"
## [16] "old_18"            "at_1"              "btn_1_4"
## [19] "btn_5_9"           "btn_10_14"         "btn_15_19"
## [22] "btn_20_29"         "btn_30_39"         "btn_40_49"
## [25] "btn_50_59"         "btn_60_69"         "btn_70_79"
## [28] "btn_80_89"         "btn_90_99"         "old_100"
## [31] "population"        "region"            "region_gdp"
## [34] "entity_global_gdp" "gdp_global"        "region_imf"
## [37] "entity_global_imf" "imf_global"        "region_uhc"
## [40] "entity_global_uhc" "region_bci"        "region_ilo"
## [43] "entity_global_ilo" "ilo_global"        "region_phy"
## [46] "entity_global_phy" "phy_global"
```

```
#visualizamos la tabla region_data_ilo
print(region_data_phy)
```

```
## # A tibble: 42 x 4
## # Groups:   code [7]
##    entity                  code   year phy_index
##    <chr>                   <chr> <int>     <dbl>
##  1 Europe and Central Asia ACS    2015      4.31
##  2 Europe and Central Asia ACS    2016      4.31
##  3 Europe and Central Asia ACS    2017      4.31
##  4 Europe and Central Asia ACS    2018      4.31
##  5 Europe and Central Asia ACS    2019      4.31
##  6 Europe and Central Asia ACS    2020      4.31
##  7 East Asia and Pacific   EAP    2015      1.68
##  8 East Asia and Pacific   EAP    2016      1.68
##  9 East Asia and Pacific   EAP    2017      1.68
## 10 East Asia and Pacific   EAP    2018      1.68
## # i 32 more rows
```

It looks like the changes have been effective, but before we dive into further testing, let's remove the global records from the dataframe that we no longer need.

Cleaning up global records

```
# Filtramos los datos que no queremos
pra1 <- pra1 %>%
  filter(region != "Organizaciones y otros", region != "No clasificado")
```

Spreading population

Some records for some countries in some years and in some specific cases lack information, so we are going to propagate the data:

```
# Propagamos el último valor no-NA hacia adelante para cada grupo de 'code' en múltiples columnas
pra1 <- pra1 %>%
  arrange(code, year) %>% # Organizamos por code y year
  group_by(code) %>% # Agrupamos por code
  fill(u_1, u_5, u_15, u_25, btn_15_64, old_15, old_18, at_1, btn_1_4, btn_5_9, btn_10_14, btn_15_19, b

# Propagamos el último valor no-NA hacia atrás para cada grupo de 'code' en múltiples columnas
pra1 <- pra1 %>%
  arrange(code, year) %>% # Organizamos por code y year
  group_by(code) %>% # Agrupamos por code
  fill(u_1, u_5, u_15, u_25, btn_15_64, old_15, old_18, at_1, btn_1_4, btn_5_9, btn_10_14, btn_15_19, b
```

Creating new binary variables.

```
# Códigos ISO de los países miembros de la OPEP
opep_countries <- c("DZA", "AGO", "GNQ", "GAB", "IRN", "IRQ", "KWT", "LBY", "NGA", "SAU", "ARE", "VEN")
pra1$opep <- ifelse(pra1$code %in% opep_countries, 1, 0)


# Nueva columna 'kyoto' y asigno 1 a todos los países excepto Andorra, Canadá, Sudán del Sur y Estados
pra1$kyoto <- ifelse(pra1$code %in% c("AND", "CAN", "SSD", "USA"), 0, 1)

# Creo una nueva columna OCDE, vector anteriormente agregado
pra1 <- pra1 %>%
  mutate(ocde = ifelse(code %in% ocde_countries, 1, 0))

# Paises en conflicto
conflict_countries <- c("AFG", "IRQ", "SYR", "YEM", "SDN", "SSD", "SOM", "COD", "UKR", "LBY", "RUS")

# Creo una nueva columna
pra1 <- pra1 %>%
  mutate(war = ifelse(code %in% conflict_countries, 1, 0))
```

Final checks

Checking again for infinite and blank values and other possible problems:

```
# Encuentra filas con al menos un NA
na_rows <- rowSums(is.na(pra1)) > 0

# Encuentra filas con al menos un blanco
blank_rows <- rowSums(pra1 == "") > 0

# Imprime las filas con al menos un NA
print("NAs")
```

```
## [1] "NAs"
```

```
print(pra1[na_rows, ])
```

```
## # A tibble: 6 x 51
## # Groups:   code [1]
##   entity code   year gdp_index imf_index uhc_index bci_index ilo_index phy_index
##   <chr>  <chr> <int>     <dbl>     <dbl>     <dbl>     <dbl>     <dbl>     <dbl>
## 1 Sub-S~ SSF    2015   0.106        17.0      47.0        45      2.2      0.209
## 2 Sub-S~ SSF    2016  -1.52         15.9      47.0        45      2.25     0.209
## 3 Sub-S~ SSF    2017  -0.254        16.5      47.0        45      2.14     0.209
## 4 Sub-S~ SSF    2018   0.00732      16.5      47.0        45      2.16     0.209
## 5 Sub-S~ SSF    2019  -0.0768       16.5      47.0        45      2.16     0.209
## 6 Sub-S~ SSF    2020  -4.52         16.5      47.0        45      2.18     0.209
## # i 42 more variables: u_1 <int>, u_5 <int>, u_15 <int>, u_25 <int>,
## #   btn_15_64 <int>, old_15 <int>, old_18 <int>, at_1 <int>, btn_1_4 <int>,
## #   btn_5_9 <int>, btn_10_14 <int>, btn_15_19 <int>, btn_20_29 <int>,
## #   btn_30_39 <int>, btn_40_49 <int>, btn_50_59 <int>, btn_60_69 <int>,
## #   btn_70_79 <int>, btn_80_89 <int>, btn_90_99 <int>, old_100 <int>,
## #   population <int>, region <chr>, region_gdp <chr>, entity_global_gdp <chr>,
## #   gdp_global <dbl>, region_imf <chr>, entity_global_imf <chr>, ...
```

```
# Imprime las filas con al menos un blanco
print("Blancos")
```

```
## [1] "Blancos"
```

```
print(pra1[blank_rows, ])
```

```
## # A tibble: 6 x 51
## # Groups:   code [1]
##   entity code   year gdp_index imf_index uhc_index bci_index ilo_index phy_index
##   <chr>  <chr> <int>     <dbl>     <dbl>     <dbl>     <dbl>     <dbl>     <dbl>
## 1 <NA>   <NA>     NA        NA        NA        NA        NA        NA        NA
## 2 <NA>   <NA>     NA        NA        NA        NA        NA        NA        NA
## 3 <NA>   <NA>     NA        NA        NA        NA        NA        NA        NA
## 4 <NA>   <NA>     NA        NA        NA        NA        NA        NA        NA
## 5 <NA>   <NA>     NA        NA        NA        NA        NA        NA        NA
## 6 <NA>   <NA>     NA        NA        NA        NA        NA        NA        NA
## # i 42 more variables: u_1 <int>, u_5 <int>, u_15 <int>, u_25 <int>,
## #   btn_15_64 <int>, old_15 <int>, old_18 <int>, at_1 <int>, btn_1_4 <int>,
## #   btn_5_9 <int>, btn_10_14 <int>, btn_15_19 <int>, btn_20_29 <int>,
## #   btn_30_39 <int>, btn_40_49 <int>, btn_50_59 <int>, btn_60_69 <int>,
## #   btn_70_79 <int>, btn_80_89 <int>, btn_90_99 <int>, old_100 <int>,
## #   population <int>, region <chr>, region_gdp <chr>, entity_global_gdp <chr>,
## #   gdp_global <dbl>, region_imf <chr>, entity_global_imf <chr>, ...
```

Note that we have several complete rows that are NA:

```
# Limpiamos NA
# Copia de seguridad
pra1_2 <- pra1


# Eliminamos las filas que contienen NA
pra1 <- na.omit(pra1)

# Encuentra filas con al menos un NA
na_rows <- rowSums(is.na(pra1)) > 0

# Encuentra filas con al menos un blanco
blank_rows <- rowSums(pra1 == "") > 0

# Imprime las filas con al menos un NA
print("NAs")
```

```
## [1] "NAs"
```

```r
print(pra1[na_rows, ])
```

```
## # A tibble: 0 x 51
## # Groups:   code [0]
## # i 51 variables: entity <chr>, code <chr>, year <int>, gdp_index <dbl>,
## #   imf_index <dbl>, uhc_index <dbl>, bci_index <dbl>, ilo_index <dbl>,
## #   phy_index <dbl>, u_1 <int>, u_5 <int>, u_15 <int>, u_25 <int>,
## #   btn_15_64 <int>, old_15 <int>, old_18 <int>, at_1 <int>, btn_1_4 <int>,
## #   btn_5_9 <int>, btn_10_14 <int>, btn_15_19 <int>, btn_20_29 <int>,
## #   btn_30_39 <int>, btn_40_49 <int>, btn_50_59 <int>, btn_60_69 <int>,
## #   btn_70_79 <int>, btn_80_89 <int>, btn_90_99 <int>, old_100 <int>, ...
```

```r
# Imprime las filas con al menos un blanco
print("Blancos")
```

```
## [1] "Blancos"
```

```r
print(pra1[blank_rows, ])
```

```
## # A tibble: 0 x 51
## # Groups:   code [0]
## # i 51 variables: entity <chr>, code <chr>, year <int>, gdp_index <dbl>,
## #   imf_index <dbl>, uhc_index <dbl>, bci_index <dbl>, ilo_index <dbl>,
## #   phy_index <dbl>, u_1 <int>, u_5 <int>, u_15 <int>, u_25 <int>,
## #   btn_15_64 <int>, old_15 <int>, old_18 <int>, at_1 <int>, btn_1_4 <int>,
## #   btn_5_9 <int>, btn_10_14 <int>, btn_15_19 <int>, btn_20_29 <int>,
## #   btn_30_39 <int>, btn_40_49 <int>, btn_50_59 <int>, btn_60_69 <int>,
## #   btn_70_79 <int>, btn_80_89 <int>, btn_90_99 <int>, old_100 <int>, ...
```

```r
#Visualizamos la tabla
glimpse(pra1)
```

```
## Rows: 1,408
## Columns: 51
## Groups: code [235]
## $ entity          <chr> "Aruba", "Aruba", "Aruba", "Aruba", "Aruba", "Aruba"~
## $ code            <chr> "ABW", "ABW", "ABW", "ABW", "ABW", "ABW", "AFG", "AF~
## $ year            <int> 2015, 2016, 2017, 2018, 2019, 2020, 2015, 2016, 2017~
## $ gdp_index       <dbl> 5.12965727, 1.58786881, 1.51982141, 0.68640423, -0.1~
## $ imf_index       <dbl> 28.51431, 28.45803, 27.92962, 27.65600, 27.46546, 28~
## $ uhc_index       <dbl> 40.83718, 40.83718, 40.83718, 40.83718, 40.83718, 40~
## $ bci_index       <dbl> 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, ~
## $ ilo_index       <dbl> 3.74, 3.67, 3.63, 3.59, 3.62, 3.55, 5.18, 5.50, 5.41~
## $ phy_index       <dbl> 0.3214649, 0.3214649, 0.3214649, 0.3214649, 0.321464~
## $ u_1             <int> 1139, 1113, 1085, 1055, 1020, 927, 1238201, 1256664,~
## $ u_5             <int> 6026, 5919, 5792, 5664, 5525, 5288, 5747862, 5894636~
## $ u_15            <int> 19599, 19480, 19334, 19150, 18912, 18494, 15456437, ~
## $ u_25            <int> 32827, 32411, 31974, 31568, 31226, 30935, 22708012, ~
## $ btn_15_64       <int> 72312, 72397, 72421, 72402, 72349, 72176, 17485012, ~
## $ old_15          <int> 84669, 85409, 86119, 86829, 87545, 88102, 18296928, ~
## $ old_18          <int> 80750, 81518, 82173, 82780, 83380, 83857, 15818155, ~
## $ at_1              <int> 1170, 1146, 1120, 1093, 1063, 1023, 1187761, 1216903~
## $ btn_1_4           <int> 4887, 4806, 4707, 4609, 4505, 4361, 4509661, 4637972~
## $ btn_5_9           <int> 6677, 6625, 6571, 6501, 6421, 6301, 5083265, 5142552~
## $ btn_10_14         <int> 6896, 6936, 6971, 6985, 6966, 6905, 4625310, 4729549~
## $ btn_15_19         <int> 6533, 6389, 6373, 6468, 6654, 6859, 3998778, 4102738~
## $ btn_20_29         <int> 12811, 12851, 12711, 12419, 12019, 11749, 5732466, 5~
## $ btn_30_39         <int> 13038, 13061, 13150, 13291, 13464, 13506, 3529267, 3~
## $ btn_40_49         <int> 15979, 15688, 15433, 15224, 15049, 14846, 2350849, 2~
## $ btn_50_59         <int> 17286, 17443, 17502, 17487, 17410, 17260, 1413659, 1~
## $ btn_60_69         <int> 11532, 12134, 12730, 13309, 13871, 14349, 814056, 83~
## $ btn_70_79         <int> 5707, 5933, 6187, 6483, 6808, 7152, 371848, 385801, ~
## $ btn_80_89         <int> 1667, 1791, 1910, 2016, 2119, 2213, 81378, 83944, 87~
## $ btn_90_99         <int> 116, 119, 123, 132, 151, 168, 4628, 4140, 4236, 4513~
## $ old_100           <int> 1, 1, 1, 1, 1, 1, 133, 72, 40, 23, 14, 32, 182, 184,~
## $ population        <int> 104269, 104890, 105454, 105980, 106458, 106597, 3375~
## $ region            <chr> "Caribe", "Caribe", "Caribe", "Caribe", "Caribe", "C~
## $ region_gdp        <chr> "LAC", "LAC", "LAC", "LAC", "LAC", "LAC", "EAP", "EA~
## $ entity_global_gdp <chr> "Latin America and Caribbean", "Latin America and Ca~
## $ gdp_global        <dbl> -0.4736689, -1.1364956, 0.9009982, 0.6864042, -0.129~
## $ region_imf        <chr> "LAC", "LAC", "LAC", "LAC", "LAC", "LAC", "SAS", "SA~
## $ entity_global_imf <chr> "Latin America and Caribbean", "Latin America and Ca~
## $ imf_global        <dbl> 28.51431, 28.45803, 27.92962, 27.65600, 27.46546, 28~
## $ region_uhc        <chr> "LWC", "LWC", "LWC", "LWC", "LWC", "LWC", "ASI", "AS~
## $ entity_global_uhc <chr> "Low-income", "Low income", "Low-income", "Low incom~
## $ region_bci        <chr> "No OCDE", "No OCDE", "No OCDE", "No OCDE", "No OCDE~
## $ region_ilo        <chr> "SID", "SID", "SID", "SID", "SID", "SID", "CSA", "CS~
## $ entity_global_ilo <chr> "Small island developing states (SIDS)", "Small isla~
## $ ilo_global        <dbl> 3.74, 3.67, 3.63, 3.59, 3.62, 3.55, 5.18, 5.50, 5.41~
## $ region_phy        <chr> "LWC", "LWC", "LWC", "LWC", "LWC", "LWC", "EAP", "EA~
## $ entity_global_phy <chr> "Low-income", "Low income", "Low-income", "Low incom~
## $ phy_global        <dbl> 0.3214649, 0.3214649, 0.3214649, 0.3214649, 0.321464~
## $ opep              <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1~
## $ kyoto             <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1~
## $ ocde              <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0~
## $ war               <dbl> 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0~
```

Having a dataframe containing 1408 rows or records, 51 columns and 235 groups or countries, without 'NA' or blanks, so the first part of this project is finished.

BIBLIOGRAPHY

"Data at WHO." n.d. https://www.who.int/data.

Firke, Sam, Bill Denney, Chris Haid, Ryan Knight, Malte Grosser, and Jonathan Zadra. 2023. *Janitor: Simple Tools for Examining and Cleaning Dirty Data.* https://cran.r-project.org/web/packages/janitor/index.html.

"Global Change Data Lab." n.d. https://global-change-data-lab.org/.

"Global Development." n.d. https://www.oxfordmartin.ox.ac.uk/global-development/.

Hennig, Christian. 2015. "What Are the True Clusters?" *Pattern Recognition Letters* 64: 53–62.

"ISO - ISO 3166 — Country Codes." n.d. https://www.iso.org/iso-3166-country-codes.html.

"Leading Indicators - Business Confidence Index (BCI) - OECD Data." n.d. http://data.oecd.org/leadind/business-confidence-index-bci.htm.

"Measuring Progress Towards the Sustainable Development Goals - SDG Tracker." n.d. https://sdg-tracker.org/.

Miluska.Jara. n.d. "Objetivos y metas de desarrollo sostenible." https://www.un.org/sustainabledevelopment/es/sustainable-development-goals/.

Ortiz-Ospina, Esteban, and Max Roser. 2016. "Government Spending." *Our World in Data*, October. https://ourworldindata.org/government-spending.

"Our World in Data." n.d. https://ourworldindata.org.

Roser, Max. 2013. "Economic Growth." *Our World in Data*, November. https://ourworldindata.org/economic-growth.

"Services Trade Restrictiveness Index | OECD Statistics on International Trade in Services | OECD iLibrary." n.d. https://www.oecd-ilibrary.org/trade/data/oecd-statistics-on-international-trade-in-services/services-trade-restrictiveness-index_g2g55848-en.

"Statista - El portal de estadísticas." n.d. https://es.statista.com/.

"The Dataverse Project - Dataverse.org." 2023. https://dataverse.org/home.

United, Nations. n.d. "United Nations | Peace, Dignity and Equality on a Healthy Planet." https://www.un.org/en/.

Wickham, Hadley, and RStudio. 2023. *Tidyverse: Easily Install and Load the 'Tidyverse'*. https://cran.r-project.org/web/packages/tidyverse/index.html.