ANA 515 Assignment 2 - Loading, Saving and Describing data

Srilekha Gurrapu

2023-06-15

Section 1: Description of the data

The dataset I am using is the COVID-19 dataset, which provides comprehensive information about the COVID-19 pandemic across multiple countries. The data measures various aspects of the pandemic, including the number of cases, deaths, and recoveries. The dataset was collected from reliable sources such as national health agencies and international organizations tracking the pandemic.

The data is saved in a CSV (Comma-Separated Values) format. It is a flat file that uses commas as the delimiter to separate the values in each row. The dataset is organized in a tabular format, where each row represents a specific observation (e.g., a specific country and date), and each column represents a variable or attribute related to that observation (e.g., cases, deaths, recoveries). The CSV format is a widely used and easily accessible format for storing structured data. It can be opened and processed using various tools and programming languages, including R, Python, and spreadsheet software.

To read this data into R, I will use the **read.csv** function, which is a base R function specifically designed for reading CSV files. This function automatically handles the parsing of the CSV format, including the detection of the delimiter and the conversion of the data into appropriate data types. The resulting data will be stored in a dataframe, which is a common data structure in R for handling tabular data.

Section 2: Reading the data into R

In this code, I used the **read_csv()** function to read the data from the CSV file and assign it to the **data** dataframe object. The **read_csv()** function from the readr package is used to read the data from the zip file. The **read_csv()** function is designed to read CSV files and is a part of the readr package.

The **unzip()** function is used to extract the file from the zip archive, and the extracted data is directly passed to the **read_csv()** function to read it as a CSV file.

```
# Reading the data into R
library(readr)

## Warning: package 'readr' was built under R version 4.2.3

# Specify the path to the zip file
zip_file <- "E:/Data Analytics - Spring 2023/ANA 515 - Fundamentals of Data Storage/Week 3/Covid 19 dataset.zip"
```

```
# Read the data from the zip file using read_csv()
data <- read_csv(unzip(zip_file))

## Rows: 35156 Columns: 10
##         Column specification

## Delimiter: ","
## chr   (2): Country/Region, WHO Region
## dbl   (7): Confirmed, Deaths, Recovered, Active, New cases, New deaths, New
r...
## date  (1): Date
##
##       Use `spec()` to retrieve the full column specification for this data.
##      Specify the column types or set `show_col_types = FALSE` to quiet this
message.
```

Section 3: Cleaning the data

In the code chunk, I performed some basic data cleaning operations on the dataset.

First, I renamed the columns of the dataframe using the **colnames** function. I provided a vector of new column names that correspond to the desired names for each column.

Next, I converted the "Date" column to the "Date" format using the **as.Date** function. This ensures that the date values are treated as dates in R, allowing for easier manipulation and analysis.

Finally, I subsetted the dataset to keep only the "Country", "Date", "Total_Cases", and "Total_Deaths" columns using indexing with square brackets ([]). This creates a new dataframe called **data_subset** that contains the selected columns.

To verify the changes and check the cleaned dataset, I used the **head** function to display the first few rows of the **data_subset** dataframe.

```
# Cleaning the data
# Rename columns
colnames(data) <- c("Date", "Country/Region", "Confirmed", "Deaths",
"Recovered", "Active", "New Cases", "New deaths", "New recovered", "WHO
Region")

library(dplyr)

## Warning: package 'dplyr' was built under R version 4.2.3

##
## Attaching package: 'dplyr'

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

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

# Perform column renaming
data <- data %>%
  rename(Country = `Country/Region`, Total_Cases = Confirmed, Total_Deaths =
Deaths, Total_Recoveries = Recovered, Active_Cases = Active)

# Convert Date to Date format
data$Date <- as.Date(data$Date)

# Subset the dataset to keep at least 4 columns
data_subset <- data[, c("Date", "Country", "Total_Cases", "Total_Deaths")]

# View the cleaned dataset
head(data_subset)

## # A tibble: 6 × 4
##   Date       Country             Total_Cases Total_Deaths
##   <date>     <chr>                     <dbl>        <dbl>
## 1 2020-01-22 Afghanistan                   0            0
## 2 2020-01-22 Albania                       0            0
## 3 2020-01-22 Algeria                       0            0
## 4 2020-01-22 Andorra                       0            0
## 5 2020-01-22 Angola                        0            0
## 6 2020-01-22 Antigua and Barbuda           0            0
```

Section 4: Characteristics of the data.

In the code, **num_rows** and **num_cols** variables are assigned the number of rows and columns in the dataframe **data**. The **column_descriptions** variable contains the descriptions for each column. The **paste()** function to insert the column names and descriptions. The resulting table will have two columns: “Column Name” and “Description”.

created a table using the **kable** function from the **knitr** package in R Markdown. This code will create a table using the **kable** function and format it as Markdown. The table will have two columns: “Column Name” and “Description”. The “column_table” data frame contains the column names and descriptions, which are then passed to the **kable** function for formatting.

```
# Number of rows and columns
num_rows <- nrow(data)
num_cols <- ncol(data)

library(knitr)
```

```
## Warning: package 'knitr' was built under R version 4.2.3
```

```
# Define the column names and descriptions
column_names <- c("Column Name", "Description")
column_descriptions <- c(
  "Date", "The date of the recorded data for COVID-19 cases.",
  "Country", "The name of the country where the COVID-19 cases were reported.",
  "Total_Cases", "The total number of confirmed COVID-19 cases in a specific country or region.",
  "Total_Deaths", "The total number of deaths caused by COVID-19 in a specific country or region.",
  "Total_Recoveries", "The total number of individuals who have recovered from COVID-19 in a specific country or region.",
  "Active_Cases", "The number of active COVID-19 cases in a specific country or region.",
  "New cases", "The number of new COVID-19 cases reported on a specific date.",
  "New deaths", "The number of new deaths due to COVID-19 reported on a specific date.",
  "New recovered", "The number of new recoveries from COVID-19 reported on a specific date.",
  "WHO Region", "The World Health Organization (WHO) region to which the country or region belongs."
)

# Create a data frame with the column names and descriptions
column_table <- data.frame(matrix(column_descriptions, ncol = 2, byrow = TRUE))
colnames(column_table) <- column_names

# Print the table using kable
kable(column_table, format = "markdown")
```

Column Name	Description
Date	The date of the recorded data for COVID-19 cases.
Country	The name of the country where the COVID-19 cases were reported.
Total_Cases	The total number of confirmed COVID-19 cases in a specific country or region.
Total_Deaths	The total number of deaths caused by COVID-19 in a specific country or region.
Total_Recoveries	The total number of individuals who have recovered from COVID-19 in a specific country or region.

Column Name	Description
Active_Cases	The number of active COVID-19 cases in a specific country or region.
New cases	The number of new COVID-19 cases reported on a specific date.
New deaths	The number of new deaths due to COVID-19 reported on a specific date.
New recovered	The number of new recoveries from COVID-19 reported on a specific date.
WHO Region	The World Health Organization (WHO) region to which the country or region belongs.

#Inline code "This data set has 35156 rows and, 10, columns., The names of the columns and a brief description of each are in the table above"

Section 5: Summary statistics.

In this code, we first define the "selected_columns" variable to contain the names of the three columns you want to analyze. We then create a subset of the dataframe "data" by selecting only those columns using "subset_data <- data[selected_columns]".

Next, we use the **apply()** function to calculate the minimum, maximum, and mean values for each column in the **subset_data** dataframe. The **na.rm = TRUE** argument is used to exclude missing values from the calculations.

We also use the **colSums()** and **is.na()** functions to count the number of missing values in each column.

Finally, we create a new dataframe **summary_data** to store the summary statistics, with columns for the column names, minimum values, maximum values, mean values, and number of missing values.

The **summary_data** dataframe is then printed to display the summary statistics in a tabular format.

```
# Select three columns
selected_columns <- c("Total_Cases", "Total_Deaths", "Total_Recoveries")

# Subset the data to include only the selected columns
subset_data <- data[selected_columns]

# Calculate summary statistics
min_values <- apply(subset_data, 2, min, na.rm = TRUE)
max_values <- apply(subset_data, 2, max, na.rm = TRUE)
mean_values <- apply(subset_data, 2, mean, na.rm = TRUE)
num_missing <- colSums(is.na(subset_data))

# Create a summary dataframe
summary_data <- data.frame(Column = selected_columns, Minimum = min_values,
```

```
Maximum = max_values, Mean = mean_values, Missing_Values = num_missing)




# Print the summary dataframe
summary_data
```

```
##                                 Column  Minimum  Maximum         Mean  Missing_Values
## Total_Cases                Total_Cases        0  4290259  23566.631                   0
## Total_Deaths              Total_Deaths        0   148011   1234.068                   0
## Total_Recoveries  Total_Recoveries            0  1846641  11048.135                   0
```