Rapport_DataChallenge_Naji_Mohammed

NAJI Mohammed

2024-12-15

1. Introduction

La pandémie de COVID-19 a eu un impact mondial sans précédent, affectant la santé publique, les systèmes économiques et sociaux. Ce rapport présente une analyse des données COVID-19 pour les pays de l'Union Européenne, en se basant sur les rapports quotidiens publiés par **Johns Hopkins University (JHU CSSE)**.

Objectifs du projet :

- Visualiser l'impact du COVID-19 sur les pays de l'Union Européenne à travers des graphiques interactifs.
- Suivre l'évolution des décès totaux dans le temps.
- Comparer les cas confirmés avec les décès pour chaque pays.
- Fournir un résumé interactif des données pour faciliter l'exploration.

Les résultats obtenus sont basés sur des données mises à jour et des visualisations pertinentes, couvrant les années 2020 à 2023.

---

2. Téléchargement des données (Nom du script = download_data_covid.R)

Le téléchargement des données utilisées dans ce projet a été automatisé à l'aide d'un script en R. Ce script récupère les rapports quotidiens disponibles dans le dépôt GitHub officiel de **Johns Hopkins University (JHU CSSE)**.

Objectifs du script

- Accéder aux fichiers CSV contenant les rapports quotidiens.
- Télécharger ces fichiers dans un répertoire local pour l'analyse.

Explications techniques

1. Utilisation des bibliothèques `httr` et `jsonlite` pour accéder à l'API GitHub.
2. Création d'un répertoire local (`data/covid`) pour stocker les fichiers téléchargés.
3. Téléchargement de chaque fichier CSV depuis l'API GitHub.
4. Gestion des erreurs pour assurer que le processus continue même si un fichier échoue.

Voici le code utilisé pour ce téléchargement :

```r
# Chargement des packages nécessaires
if (!require(httr)) install.packages("httr")
if (!require(jsonlite)) install.packages("jsonlite")
library(httr)
library(jsonlite)

# Fonction de téléchargement des rapports quotidiens COVID-19
download_covid_reports <- function(base_path = "data/covid", start_date = "2020-01-22") {
  # Création du répertoire si inexistant
  if (!dir.exists(base_path)) {
    dir.create(base_path, recursive = TRUE)
  }

  # Point d'accès à l'API GitHub pour le contenu du dépôt
  api_url <- "https://api.github.com/repos/CSSEGISandData/COVID-19/contents/csse_covid_19_data/csse_cov

  # Récupération des contenus du dépôt avec gestion des erreurs
  message("Récupération du contenu du dépôt...")
  response <- GET(api_url)
  if (status_code(response) != 200) {
    stop(sprintf("Erreur d'accès à l'API GitHub. Code : %d\nRéponse : %s",
                 status_code(response),
                 rawToChar(response$content)))
  }

  # Analyse de la réponse JSON
  files <- fromJSON(rawToChar(response$content))

  # Filtrage des fichiers CSV
  csv_files <- files[grep("\\.csv$", files$name), ]
  message(sprintf("Fichiers CSV trouvés : %d", nrow(csv_files)))

  # Téléchargement de chaque fichier CSV
  successful_downloads <- 0
  failed_downloads <- 0

  for (i in 1:nrow(csv_files)) {
    file_name <- csv_files$name[i]
    download_url <- csv_files$download_url[i]

    # Chemin local
    local_path <- file.path(base_path, file_name)

    # Téléchargement
    tryCatch({
```

```
      message(sprintf("[%d/%d] Téléchargement de %s...", i, nrow(csv_files), file_name))
      download.file(download_url, local_path, mode = "wb", quiet = TRUE)
      successful_downloads <- successful_downloads + 1
      Sys.sleep(0.5)  # Pause pour éviter la surcharge
    }, error = function(e) {
      warning(sprintf("Échec du téléchargement pour %s : %s", file_name, e$message))
      failed_downloads <- failed_downloads + 1
    })
  }

  # Résumé du téléchargement
  message("\nRésumé du téléchargement :")
  message(sprintf("Fichiers totaux trouvés : %d", nrow(csv_files)))
  message(sprintf("Téléchargés avec succès : %d", successful_downloads))
  message(sprintf("Échecs : %d", failed_downloads))
}

# Exemple d'utilisation
download_path <- "covid_data"
download_covid_reports(download_path)
```

3. Le code principal (Nom du Script = datachallenge.R)

Nettoyage et préparation des données

Pour cette analyse, nous nous sommes concentrés sur les pays membres de l'Union Européenne (UE). Les étapes suivantes ont été réalisées : - Les données brutes ont été combinées depuis les fichiers CSV téléchargés. - Les colonnes clés ont été sélectionnées et nettoyées. - Les métriques principales ont été calculées, telles que les décès totaux et les cas confirmés.

Voici le code utilisé pour cette étape :

```
library(dplyr)
library(readr)

folder_path <- "C:/Users/moha_/Documents/covid_data"
# Lister toutes les archives CSV dans le dossier
file_list <- list.files(path = folder_path, pattern = "\\.csv$", full.names = TRUE)
# Fonction pour lire toutes les données et forcer les colonnes à être des caractères comme type
read_as_character <- function(file) {
  read_csv(file, col_types = cols(.default = "c"))  # Forcer toutes les colonnes comme type 'character'
}

# Lire et combiner les archives
covid_data <- file_list %>%
  lapply(read_as_character) %>%  # Lire les archives comme liste de data frames
  bind_rows()                    # Combinaison en un seul data frame

# Conversion de la variable 'Last_Update' en format datetime
covid_data <- covid_data %>%
  mutate(`Last Update` = as.POSIXct(Last_Update, format = "%Y-%m-%d %H:%M:%S", tz = "UTC") %>%
           coalesce(as.POSIXct(Last_Update, format = "%m/%d/%Y %H:%M", tz = "UTC")))
```

Analyse exploratoire des données

Après avoir combiné et nettoyé les données, plusieurs analyses exploratoires ont été effectuées pour comprendre leur structure et identifier des incohérences éventuelles. Nous avons également filtré les données pour ne conserver que les pays membres de l'Union Européenne.

```
# pour connaitre a quoi il rassemble: (3.708.419 lignes, 21 colonnes)
dim(covid_data)
str(covid_data)
tail(covid_data)
head(covid_data)
colnames(covid_data)

# regarder des valeures uniques dans des col clés
unique(covid_data$Country_Region) #noms des tous les pays dispo
unique(covid_data$Last_Update) #toutes les dates dispo, 1000 dates differentes

# comparaison des colonnes semblantes, les plus pertinentes
identical(covid_data$Last_Update, covid_data$`Last Update`)  # ¿Son idénticas?
# car elle sont pas indentiques on filtre pour voir c'est quoi les diffs
covid_data %>%
  filter(Last_Update != `Last Update`) %>%
  select(Last_Update, `Last Update`) %>%
  head()

#management des valeurs manquantes
sum(is.na(covid_data$Last_Update))
unique(covid_data$Last_Update)

covid_data$`Last Update`
str(covid_data$`Last Update`)
sum(is.na(covid_data$`Last Update`))
#on voit que 'Last Update' est indentique à Last_Update, donc on elimine un

# Creation d'un nouveau data frame pour les pays de l'europe
european_countries <- c(
  "Austria", "Belgium", "Bulgaria", "Croatia", "Cyprus", "Czechia", "Denmark",
  "Estonia", "Finland", "France", "Germany", "Greece", "Hungary", "Ireland",
  "Italy", "Latvia", "Lithuania", "Luxembourg", "Malta", "Netherlands",
  "Poland", "Portugal", "Romania", "Slovakia", "Slovenia", "Spain", "Sweden"
)
population_data <- data.frame(
  Country_Region = eu_countries,
  Population = c(
    8917205, 11555997, 6948445, 4047200, 1214558, 10701777, 5889575, 1331057,
    5533366, 67564251, 83900471, 10300434, 9655361, 5021529, 58112786, 1886198,
    2793474, 645397, 514100, 17533405, 38268000, 10196709, 19132604, 5460102,
    2100126, 47329981, 10549379
  )
)

# filtrer les donnees par pays europeens
european_data <- covid_data %>%
  filter(Country_Region %in% european_countries) %>%
```

```r
  select(
    Country_Region,
    Last_Update,
    Confirmed,
    Deaths,
    Lat,
    Long_
  )
european_data <- european_data %>%
  left_join(population_data, by = "Country_Region")

european_data <- european_data %>%
  mutate(
    Deaths = as.numeric(Deaths),
    Confirmed = as.numeric(Confirmed),
    Lat = as.numeric(Lat),
    Long_ =as.numeric(Long_)
  )
#quelques tests et traitements pour comprendre la BD:
min(european_data$Last_Update, na.rm = TRUE)
max(european_data$Last_Update, na.rm = TRUE)

# filtrer pour avoir la derniere mise a jour par pays
latest_data <- european_data %>%
  group_by(Country_Region) %>%
  filter(Last_Update == max(Last_Update, na.rm = TRUE)) %>%
  ungroup()

# sum des morts totales a la derniere dates enrregistres, par pays
latest_by_country <- latest_data %>%
  group_by(Country_Region, Last_Update) %>%
  summarise(
    Total_Deaths = sum(Deaths, na.rm = TRUE),
    Population = max(Population, na.rm = TRUE)
  ) %>%
  ungroup()
# calculer les coordonees moyennes par pays
coordinates_by_country <- european_data %>%
  group_by(Country_Region) %>%
  summarise(
    Avg_Lat = mean(Lat, na.rm = TRUE),
    Avg_Long = mean(Long_, na.rm = TRUE)
  ) %>%
  ungroup()
# on rajoute les coordonees dans le df
latest_by_country <- latest_by_country %>%
  left_join(coordinates_by_country, by = "Country_Region")
```

Visualisations

Carte interactive

La carte ci-dessous montre les décès totaux liés au COVID-19 dans les pays de l'Union Européenne.

```r
# avec latest_by_country on cree le premier graphique - LA CARTE DE L'UE

library(plotly)

choropleth_map <- plot_ly(
  data = latest_by_country,
  type = 'choropleth',
  locations = ~Country_Region,  #col avec les noms des pays
  locationmode = 'country names',
  z = ~Total_Deaths, # ce qu'on veut représenter
  text = ~paste(
    "<b>Pays :</b> ", Country_Region, "<br>",
    "<b>Nombre total de morts :</b> ", Total_Deaths, "<br>",
    "<b>Population :</b> ", Population
  ),  # info qui s'affiche quand on passe le curseur
  colorscale = 'Reds',  #echelle des couleurs
  colorbar = list(title = "Morts totales")
) %>%
  layout(
    title = "Carte des décès par COVID-19 dans l'Union Européenne",
    geo = list(
      projection = list(type = 'natural earth'),
      showframe = FALSE,  # sans marc
      showcoastlines = TRUE
    )
  )

#print du graph:
choropleth_map
```

Courbe épidémique

Le graphique suivant présente l'évolution temporelle des décès liés au COVID-19 dans les pays européens.

```r
#pour s'assurer d'inclure la premiere date de 2020:
selected_dates <- european_data %>%
  mutate(Year = format(Last_Update, "%Y"),  #extraction annee
         Month = format(Last_Update, "%m")) %>%  #extraction mois
  filter(
    (Month %in% c("02", "10") & Year != "2022") |  #filtrage fevrier et octobre,sauf 2022
      (Year == "2022" & Month == "03")  #pour 2022 prendre moi de mars
  ) %>%
  group_by(Year, Month) %>%
  summarise(
    Date = max(Last_Update, na.rm = TRUE)  #selectioner la dernier date valable
  ) %>%
  ungroup() %>%
  arrange(Date)

#inclure la premier date de 2022
selected_dates <- selected_dates %>%
  add_row(Year = "2020", Month = "03", Date = min(european_data$Last_Update, na.rm = TRUE))
```

```r
#filtrer european_data (df) par les dates selectionnes (de notre interet):
european_data_selected <- european_data %>%
  filter(Last_Update %in% selected_dates$Date) %>%
  mutate(Date = as.Date(Last_Update))

#somme des morts totales par date:
deaths_over_time <- european_data_selected %>%
  group_by(Date) %>%
  summarise(Total_Deaths = sum(Deaths, na.rm = TRUE)) %>%
  arrange(Date)

#création du graph intéractif avec plotly
courbe_epidemique <- plot_ly(
  data = deaths_over_time,
  x = ~Date,
  y = ~Total_Deaths,
  type = 'scatter',
  mode = 'lines+markers',
  marker = list(size = 10, color = 'brown'),
  line = list(color = 'orange', width = 2),
  text = ~paste(
    "Date: ", Date, "<br>",
    "Morts totales: ", Total_Deaths
  ),  #info qui apparait quand on passe le curseur
  hoverinfo = "text"  #pour montrer cette info
) %>%
  layout(
    title = "Évolution des décès totales par COVID-19 à l'UE",
    xaxis = list(title = "Date"),
    yaxis = list(title = "Nombre total de morts"),
    hovermode = "closest"
  )

#print du graph:
courbe_epidemique
```

Grahpique à barres empilées

Le graphique qui suit représente la comparaison des cas confirmés et des décès totau par pays.

```r
#au moment de la derniere mise à jour
#filtrer pour obtenir la dernier mise à jour par pays
latest_data_confirmed <- european_data %>%
  group_by(Country_Region) %>%
  filter(Last_Update == max(Last_Update, na.rm = TRUE)) %>%
  ungroup()

#somme des cas confirmes par pays
confirmed_by_country <- latest_data_confirmed %>%
  group_by(Country_Region, Last_Update) %>%
  summarise(
    Total_Confirmed = sum(Confirmed, na.rm = TRUE)  # Sumar los casos confirmados
  ) %>%
```

```r
  ungroup()

#création d'une copie de latest_by_country
latest_by_country_with_confirmed <- latest_by_country %>%
  left_join(confirmed_by_country %>% select(Country_Region, Total_Confirmed),
            by = "Country_Region")

library(plotly)
#preparation des donnes pour le grpahique interactif
stacked_bar_interactive <- latest_by_country_with_confirmed %>%
  gather(key = "Metric", value = "Count", Total_Deaths, Total_Confirmed)

graph_barres_empilées <- plot_ly(
  data = stacked_bar_interactive,
  x = ~Count,
  y = ~reorder(Country_Region, Count),
  type = 'bar',
  orientation = 'h',
  color = ~Metric,  #differencier par métrique cas/morts
  colors = c("Total_Deaths" = "brown", "Total_Confirmed" = "orange"),
  text = ~paste(
    "Pays: ", Country_Region, "<br>",
    "Métrique: ", ifelse(Metric == "Total_Deaths", "Morts totales", "Cas confirmés"), "<br>",
    "Nombre: ", Count
  ),
  hoverinfo = "text"  #montrer le texte personnalisée quand on passe le curseur
) %>%
  layout(
    title = "Cas confirmés et morts par pays (UE)",
    xaxis = list(title = "Nombre"),
    yaxis = list(title = "Pays"),
    barmode = 'stack',
    legend = list(title = list(text = "Métrique")),
    margin = list(l = 100)
  )

#print du graphique
graph_barres_empilées
```

Table intéractive

Résumé interactif des cas confirmés, des décès, de la population, ainsi que des coordonnées moyennes (latitude et longitude) des pays de l'Union Européenne.

```r
#instaler et charger le package DT si il y est pas
if (!require(DT)) install.packages("DT")
library(DT)

#preparer les donnees pour la table:
interactive_table <- latest_by_country_with_confirmed %>%
  select(
    Country_Region,
    Population,
```

```
    Total_Confirmed,
    Total_Deaths,
    Avg_Lat, Avg_Long
  ) %>%
  arrange(desc(Total_Confirmed))

#création de la rable:
datatable(
  interactive_table,
  options = list(
    pageLength = 10,          #montrer 10 lignes par page
    autoWidth = TRUE,
    dom = 'Bfrtip',           #barre de recherche
    buttons = c('copy', 'csv', 'excel', 'pdf', 'print')  #exportation des données
  ),
  class = 'cell-border stripe hover',
  rownames = FALSE,
  caption = htmltools::tags$caption(
    style = 'caption-side: bottom; text-align: center;',
    "Tableau interactif des données COVID-19 pour les pays de l'Union Européenne"
  )
)
```

Sauvegarde des données pour le dashboard

Sauvegarde des fichiers d'intéret pour la posterieur création du dashboard:

```
#Pour la création d'un dashboard je sauvegarde en locale les df necessaires:
save(
  latest_by_country,
  latest_by_country_with_confirmed,
  deaths_over_time,
  stacked_bar_interactive,
  interactive_table,
  file = "dashboard_data.RData"
)
```

4. Création du dashboard (Nom du script = app.R)

Ce code implémente un dashboard interactif en utilisant Shiny. Il permet d'explorer les données COVID-19 des pays de l'Union Européenne à travers quatre onglets : une carte interactive, une courbe temporelle, un graphique à barres empilées, et un tableau interactif. Les données sont chargées depuis le fichier préalablement sauvegardé dashboard_data.RData."

```
#charger les donnes pour le dashboard
load("dashboard_data.RData")

library(shiny)
library(shinydashboard)
library(DT)
library(plotly)
```

```r
library(leaflet)

# UI du dashboard
ui <- dashboardPage(
  dashboardHeader(title = "Dashboard COVID-19 à l'Union Européenne"),
  dashboardSidebar(
    sidebarMenu(
      menuItem("Carte interactive", tabName = "map", icon = icon("globe")),
      menuItem("Évolution temporelle", tabName = "line", icon = icon("chart-line")),
      menuItem("Graphique par pays", tabName = "bar", icon = icon("bar-chart")),
      menuItem("Tableau interactif", tabName = "table", icon = icon("table"))
    )
  ),
  dashboardBody(
    tabItems(
      #onglet pour la carte:
      tabItem(tabName = "map",
              h2("Carte des morts par COVID-19"),
              plotlyOutput("map", height = 600)
      ),

      #onglet pour la courbe épidémique
      tabItem(tabName = "line",
              h2("Évolution des morts totales"),
              plotlyOutput("line_plot", height = 600)
      ),

      #onglet pour le graphique des barres empilées
      tabItem(tabName = "bar",
              h2("Cas confirmés et morts par pays"),
              plotlyOutput("bar_plot", height = 600)
      ),

      #onglet pour la table intéractive
      tabItem(tabName = "table",
              h2("Tableau interactif des données COVID-19"),
              DTOutput("table", height = 600)
      )
    )
  )
)

# Server dashboard
server <- function(input, output) {


  output$map <- renderPlotly({
    choropleth_map
  })


  output$line_plot <- renderPlotly({
    courbe_epidemique
```

```
  })


  output$bar_plot <- renderPlotly({
    graph_barres_empilées
  })


  output$table <- renderDT({
    datatable(
      interactive_table,
      options = list(
        pageLength = 10,
        autoWidth = TRUE,
        dom = 'Bfrtip',
        buttons = c('copy', 'csv', 'excel', 'pdf', 'print')
      ),
      class = 'cell-border stripe hover',
      rownames = FALSE
    )
  })
}

# execution du dashboard:
shinyApp(ui, server)
```

5. Conclusion

Ce projet a permis de réaliser une analyse approfondie des données COVID-19 pour les pays de l'Union Européenne, en combinant plusieurs étapes de traitement et de visualisation des données. Voici un résumé des principaux aspects abordés :

1. Bibliothèques utilisées Pour mener à bien ce projet, nous avons utilisé diverses bibliothèques R, chacune jouant un rôle spécifique :

-dplyr : pour le filtrage, la manipulation et l'agrégation des données. -readr : pour lire efficacement les fichiers CSV. -plotly : pour créer des graphiques interactifs, tels que des cartes et des courbes. -DT : pour générer des tableaux interactifs, permettant une exploration flexible des données. -shiny et shinydashboard : pour concevoir un tableau de bord dynamique et interactif. -leaflet : pour des visualisations géographiques, bien que finalement remplacée par Plotly pour ce projet.

2. Problèmes rencontrés et solutions

Lors de l'exploration initiale des données, nous avons rencontré plusieurs défis :

-Colonnes en double et types incohérents : Certaines colonnes (Last Update et Last_Update) étaient redondantes et avaient des formats différents. Nous avons uniformisé ces colonnes en les convertissant au format POSIXct.

-Valeurs manquantes et incohérences : Des valeurs NA et des doublons dans les données par pays et par date ont nécessité un nettoyage approfondi.

-Pays avec subdivisions : Les données étaient ventilées par provinces pour certains pays, ce qui a conduit à une duplication des valeurs. Pour résoudre ce problème, nous avons agrégé les données au niveau national en utilisant les dernières mises à jour disponibles pour chaque pays.

-Périodes manquantes : Pour la courbe temporelle, il a fallu sélectionner des dates spécifiques (février et octobre de chaque année) pour obtenir une représentation cohérente de l'évolution.

3. Visualisations et résultats

Plusieurs visualisations interactives ont été créées pour mieux comprendre l'impact du COVID-19 dans les pays de l'Union Européenne :

-Carte des décès totaux : La carte a révélé des disparités importantes entre les pays en termes de décès, avec des chiffres particulièrement élevés dans des pays comme la France, l'Allemagne et l'Italie.

-Courbe épidémique : Cette visualisation a montré une augmentation significative des décès au cours des premières vagues (2020) et des vagues ultérieures, avant une stabilisation vers 2023.

-Graphique à barres empilées : La comparaison des cas confirmés et des décès a mis en évidence les écarts significatifs entre ces deux métriques, montrant que certains pays (comme l'Allemagne et la France) ont enregistré un nombre élevé de cas avec des taux de mortalité relativement faibles.

-Tableau interactif : Ce tableau a fourni une vue synthétique des données, y compris les coordonnées moyennes (latitude et longitude) pour chaque pays, permettant une exploration plus approfondie des métriques clés.

4. Contribution du tableau de bord

Le tableau de bord interactif conçu avec Shiny et Plotly a permis de centraliser toutes les visualisations et analyses dans une interface simple et intuitive. Ce tableau de bord offre aux utilisateurs une exploration dynamique des données, combinant des cartes, des courbes, des graphiques et des tableaux.

En conclusion, ce projet démontre l'importance d'un nettoyage rigoureux des données pour garantir des analyses fiables. Les bibliothèques R utilisées ont permis de traiter un volume de données conséquent (plus de 3 millions de lignes) et de produire des visualisations pertinentes. Les résultats obtenus mettent en lumière l'impact varié du COVID-19 sur les pays de l'Union Européenne et offrent une base solide pour de futures analyses comparatives ou prédictives.

ANNNEXE:

Note sur l'interactivité

Pour respecter les exigences de présentation en format PDF, les graphiques interactifs et le tableau ont été convertis en images statiques. Cela garantit une compatibilité totale avec le format demandé, mais cela limite l'interactivité initialement prévue.

Cependant, une version HTML du projet, incluant toutes les fonctionnalités interactives, est disponible sur demande. Cette version permet d'explorer dynamiquement les données grâce aux graphiques interactifs et au tableau, offrant ainsi une expérience utilisateur plus immersive.

LES GRAPHIQUES:

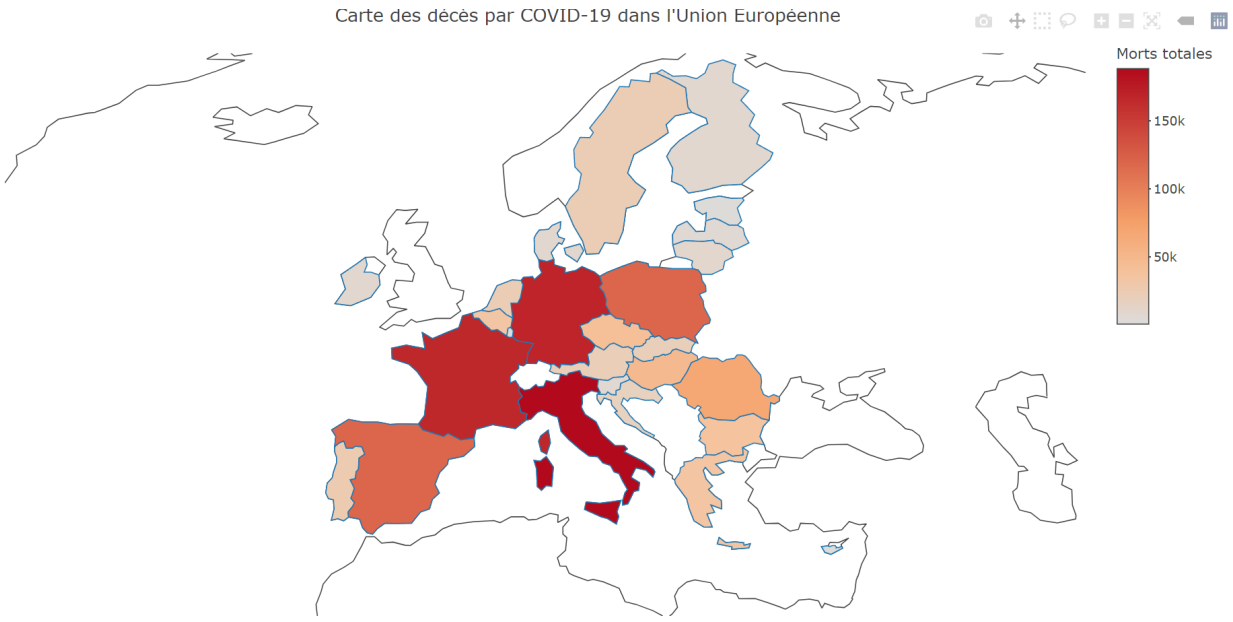

Figure 1: Carte des décès à l'UE

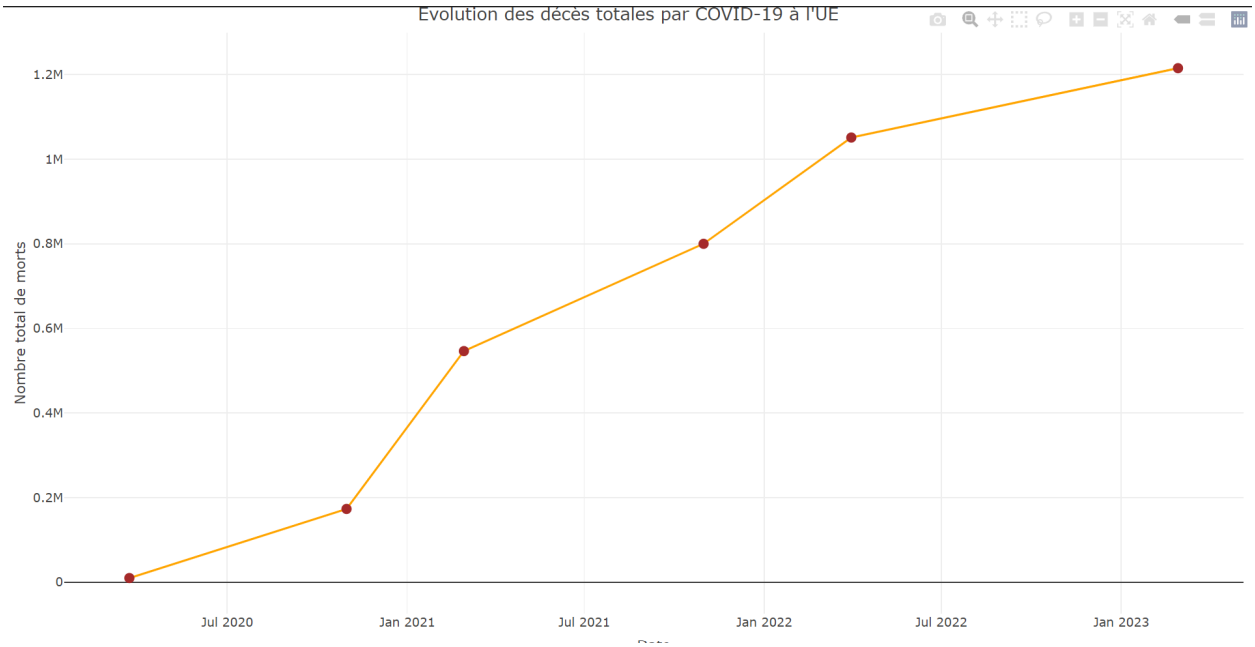

Figure 2: Courbe épidémique

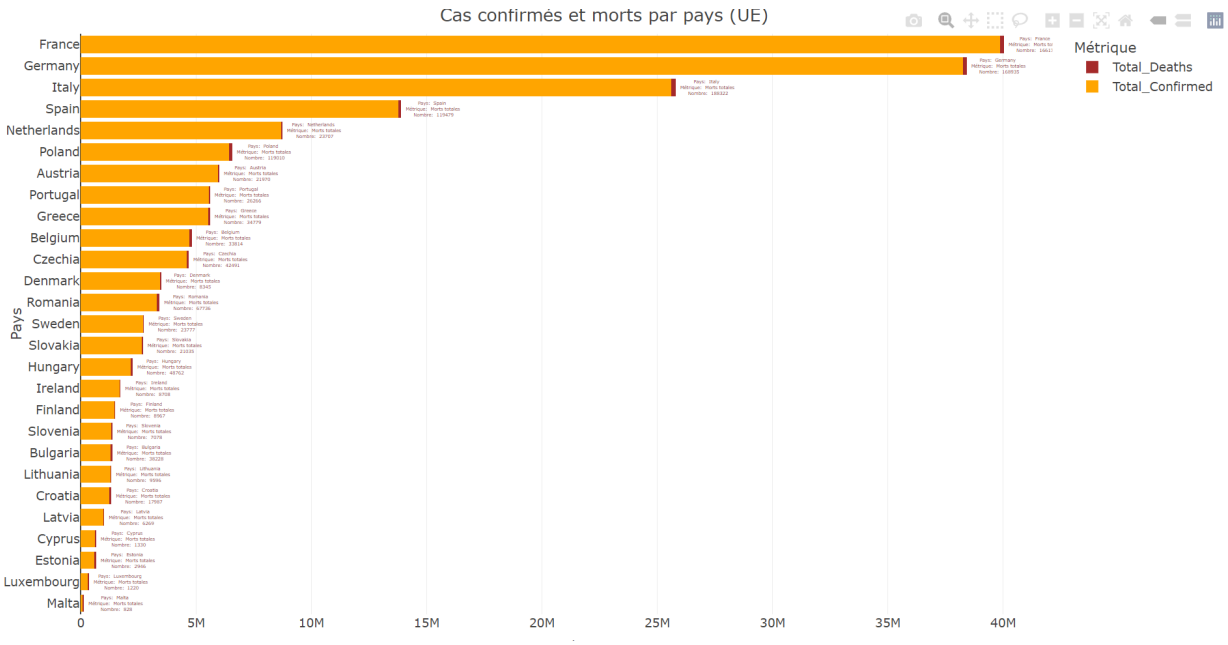

Figure 3: Graphique à barres empilées

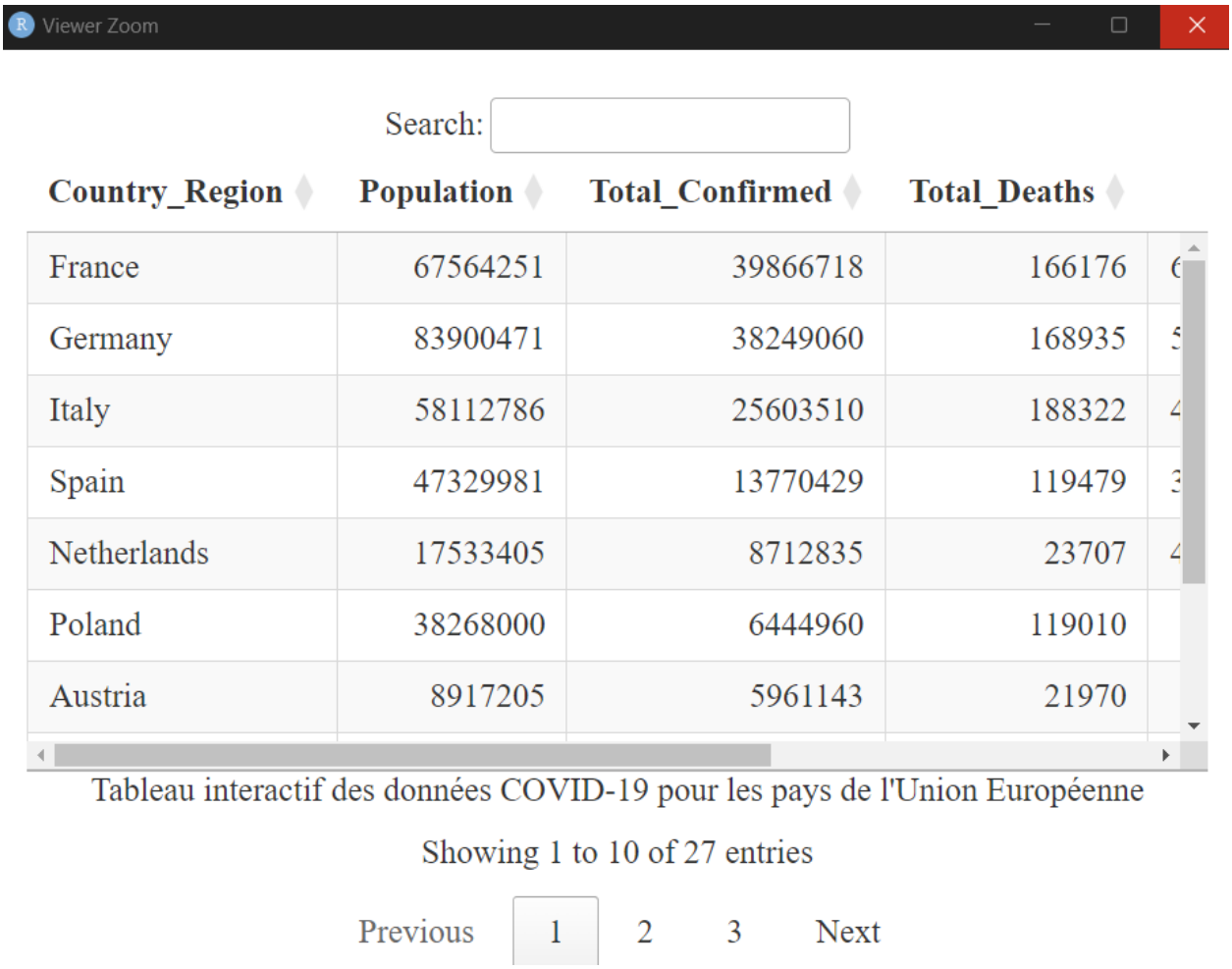

Figure 4: Tableau des données COVID-19