



RAPPORT DE PROJET

THEME : Mise en place d'un système Big Data pour une étude épidémiologique

Membres du groupe :

- Guillon Xavier
- KENFACK Ivana
- LONTSIE Dilane
- MADJOU Flora

Table des matières

RAPPORT DE PROJET	1
CONTEXT	3
1. ETUDE DES BESOINS	3
1.1. Le choix de l'épidémie.....	3
1.2. Objectifs du projet.....	3
1.3. Les sources	4
2 . MISE EN PLACE DE L'ARCHITECTURE DE STOCKAGE	5
Installation de l'environnement de travail.....	5
3. TRAITEMENT DES DONNÉES	7
3.1. Traitement dans SPARK	7
3.2. Traitement dans Rstudio	8
3. VISUALISATION DES RÉSULTATS	13
SOURCES	14

CONTEXT

Dans le cadre de notre cours de Big Data, il nous a été demandé de mener un projet d'analyse avec une approche Big Data. De ce fait, nous avons choisi de nous porter vers une étude épidémiologique, qui sera le cœur de notre projet que nous explicitons au fil des pages suivantes. Le travail effectué sera structuré en 5 grandes étapes : l'étude du besoin, la mise en place de l'environnement de stockage et de traitement, l'ingestion/extraction des données, le traitement de données et la visualisation des données.

1. ETUDE DES BESOINS

1.1. Le choix de l'épidémie

Le thème du projet étant assigné, il fallait trouver une problématique en lien avec l'évolution de l'épidémie, son impact selon les critères que nous avons dû déterminer. La première décision qu'il nous a fallu prendre a été le choix de l'épidémie à étudier et nous avons décidé de nous intéresser à l'évolution du SIDA dans le monde.

1.2. Objectifs du projet

La tâche la plus importante lors de cette première étape a été de décider quels allaient être les objectifs de notre projet. Après de longues réflexions nous avons décidé de choisir comme objectif :

- Identifier les populations les plus touchées, et les raisons pour lesquelles elles le sont. Il sera question pour nous d'effectuer une analyse descriptive sur l'évolution du VIH.
- Réaliser une analyse prédictive des populations qui seront les plus touchées par le VIH.
- Évaluer l'efficacité des interventions de prévention et de traitement: Les données sur le SIDA nous aideront à aider à évaluer l'efficacité des interventions de prévention, telles que les programmes de dépistage, l'accès aux traitements antirétroviraux et la promotion de comportements sexuels à faible risque.

1.3. Les sources

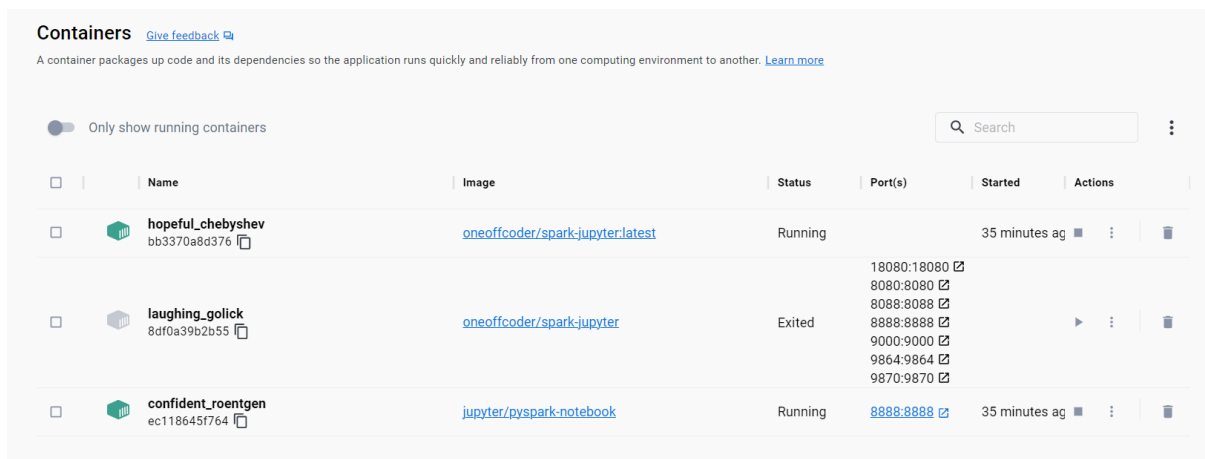
Cette étape a consisté à identifier les sources de données que nous allons utiliser pour pouvoir réaliser notre projet Big Data. Après de longues recherches, notre choix s'est porté sur le dépôt de données disponibles sur Kaggle et sur World Health Organisation (who). Les principaux avantages de ces dépôts sont qu'il est mis à jour régulièrement et qu'il est très complet (regroupe toutes les informations dont nous avons besoin pour le projet).

2. MISE EN PLACE DE L'ARCHITECTURE DE STOCKAGE

Cette étape avait pour principal but la mise en place de notre environnement de travail. La technologie que nous avons choisie pour notre projet est Hadoop qui est un Framework libre et open source qui permet le stockage et le traitement de larges volumes de données dans un environnement distribué (sur plusieurs machines distinctes).

Installation de l'environnement de travail

- Nous avons utilisé Docker pour mettre en place notre environnement de travail



The screenshot shows the Docker Desktop interface with the 'Containers' tab selected. It displays a list of containers with columns for Name, Image, Status, Port(s), Started, and Actions. There are three containers listed: 'hopeful_chebyshev' (Running), 'laughing_golick' (Exited), and 'confident_roentgen' (Running). The 'hopeful_chebyshev' container is using the 'oneoffcoder/spark-jupyter:latest' image and has ports 18080, 8080, 8088, 8888, 9000, 9864, and 9870 exposed. The 'laughing_golick' container is also using the 'oneoffcoder/spark-jupyter' image but has exited. The 'confident_roentgen' container is using the 'jupyter/pyspark-notebook' image and has port 8888 exposed.

Name	Image	Status	Port(s)	Started	Actions
hopeful_chebyshev bb3370a8d376	oneoffcoder/spark-jupyter:latest	Running	18080:18080 8080:8080 8088:8088	35 minutes ag	[Stop] [Restart] [Logs] [Delete]
laughing_golick 8df0a39b2b55	oneoffcoder/spark-jupyter	Exited	8888:8888 9000:9000 9864:9864 9870:9870		[Stop] [Restart] [Logs] [Delete]
confident_roentgen ec118645f764	jupyter/pyspark-notebook	Running	8888:8888	35 minutes ag	[Stop] [Restart] [Logs] [Delete]

- Nous avons opté pour un cluster Hadoop constitué d'un maître et de deux esclaves, pour se faire nous avons importé une image contenant les éléments dont nous aurons besoin pour nos travaux, nous avons utilisé l'image offerte oneoffcoder à travers ce lien <https://github.com/oneoffcoder/docker-containers/tree/master/spark-jupyter>
- Télécharger l'image docker à l'aide de la commande **docker pull oneoffcoder/spark-jupyter**

On remarque bien la présence de notre image parmi les différentes images déjà présentes sur docker.

```
C:\Users\ivana>docker images
REPOSITORY          TAG         IMAGE ID      CREATED        SIZE
jupyter/pyspark-notebook  latest     338aa79da631  18 hours ago  4.4GB
oneoffcoder/spark-jupyter  latest     80ad930aead9  20 months ago  9.11GB
```

Enfin on lance l'exécution des différents conteneurs grâce à la commande:

```
docker run -it -p 9870:9870 -p 8088:8088 -p 8080:8080 -p 18080:18080 -p 9000:9000 -p 8888:8888 -p 9864:9864 -v $HOME/git/docker-containers/spark-jupyter/ubunturoot/ipynb:/root/ipynb -e PYSPARK_MASTER=spark://localhost:7077 oneoffcoder/spark-jupyter
```

Ainsi notre cluster Hadoop est lancé et ainsi on a procédé au stockage des données qui seront par la suite répliquées

Hadoop

Overview

Datanodes

Datanode Volume Failures

Snapshot

Startup Progress

Utilities

Browse Directory

/user

Go!

Show

25

entries

Search:

<input type="checkbox"/>	Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name	
<input type="checkbox"/>	-rw-r--r--	dr.who	supergroup	243.15 KB	Apr 12 22:40	1	128 MB	data1.csv	
<input type="checkbox"/>	-rw-r--r--	dr.who	supergroup	243.15 KB	Apr 12 22:41	1	128 MB	data2.csv	
<input type="checkbox"/>	-rw-r--r--	dr.who	supergroup	158.51 KB	Mar 31 10:35	1	128 MB	new-cases-of-hiv-infection.csv	
<input type="checkbox"/>	-rw-r--r--	dr.who	supergroup	42.44 KB	Mar 31 10:49	1	128 MB	tb-patients-living-with-hiv-receiving-art.csv	
<input type="checkbox"/>	-rw-r--r--	dr.who	supergroup	51.49 KB	Mar 31 10:51	1	128 MB	tb-patients-tested-positive-for-hiv.csv	
<input type="checkbox"/>	-rw-r--r--	dr.who	supergroup	100.05 KB	Mar 31 10:48	1	128 MB	tb-related-deaths-hiv.csv	

Showing 1 to 6 of 6 entries

Previous

1

Next

Hadoop, 2019.

3. TRAITEMENT DES DONNÉES

3.1. Traitement dans SPARK

Le traitement des données avec Py Spark est un processus qui permet d'analyser et de manipuler de grandes quantités de données en utilisant la puissance de calcul distribuée. C'est ainsi que nous avons opté pour le nettoyage et le traitement de données afin de sortir des analyses pertinentes.

Tout d'abord, il était important pour nous de charger les données dans une session Spark en utilisant des fonctions de lecture de fichiers telles que `read()`. Ensuite, nous avons procédé au nettoyage des données en utilisant des transformations Py Spark pour supprimer les données manquantes, filtrer les valeurs aberrantes et formater les données en vue d'une analyse ultérieure.

```
[ ]: 1 from pyspark.sql import SparkSession
      2 import numpy as np
      3 import pandas as pd
      4 import matplotlib.pyplot as plt
      5 import seaborn as sns
      6
      7
      8 spark = SparkSession.builder \
      9     .appName("my_app_name") \
     10     .master("local[*]") \
     11     .getOrCreate()
```

```
[ ]: 1
Charger les données
```

```
[ ]: 1 df = spark.read.load('hdfs://localhost:8020/user/new-cases-of-hiv-infection.csv',
      format='com.databricks.spark.csv',header='true',sep=',',inferSchema='true')
```

```
[3]: 1 df.show(50)
```

Une fois les données nettoyées, il est temps d'effectuer des opérations de transformation pour identifier les populations les plus touchées par le VIH. Par exemple des opérations de jointure et de regroupement pour combiner les données de différents ensembles de données.

Afghanistan	AFG	1990	88	3	95.512
Afghanistan	AFG	1990	88	3	105.271
Afghanistan	AFG	1990	88	3	110.409

only showing top 20 rows

summary	Entity	Code	Year	Incidence - VIH/SIDA - Sexe	TB patients living with HIV receiving ART	Estimated TB-related deaths
count	942300	942300	942300	942300	942300	942300
mean	null	null	2004.5	15986.759524567547	1576.5124482648837	3249.6289371413277
stddev	null	null	8.65544604112354	54861.73896506539	8163.078707494035	11545.196622258329
min	Afghanistan	AFG	1990	0	0	0.0
max	Zimbabwe	ZWE	2019	668406	141755	101819.0

```
[ ]: 1 from pyspark.sql.functions import to_date
2 # Afficher Le schéma du dataframe
3 DF.printSchema()
4
5 # Convertir la colonne "year" en type "date"
6 DF = DF.withColumn("Year", to_date(DF["Year"].cast("string"), "yyyy"))
7
8 # Afficher Le schéma du dataframe mis à jour
9 DF.printSchema()
10
```

Calcul du pourcentage de valeurs manquantes

```
[ ]: 1 import matplotlib.pyplot as plt
2
3 # Calculer le pourcentage de valeurs manquantes pour chaque colonne
4 missing_percentages = DF.select([(sum(col(c).isNull().cast("int")) / df.count() * 100).alias(c) for c in df.columns])
5
6 # Transformer le résultat en Pandas DataFrame pour la visualisation
7 missing_percentages_pd = missing_percentages.toPandas().transpose()
8 missing_percentages_pd.columns = ['% de valeurs manquantes']
9
10 # Afficher le diagramme à barres
11 missing_percentages_pd.plot(kind='bar', figsize=(12,6))
12 plt.title('Pourcentage de valeurs manquantes par colonne')
13 plt.xlabel('Colonnes')
14 plt.ylabel('% de valeurs manquantes')
15 plt.show()
```

3.2. Traitement dans Rstudio

3.2.1. Mise en place du conteneur de rstudio

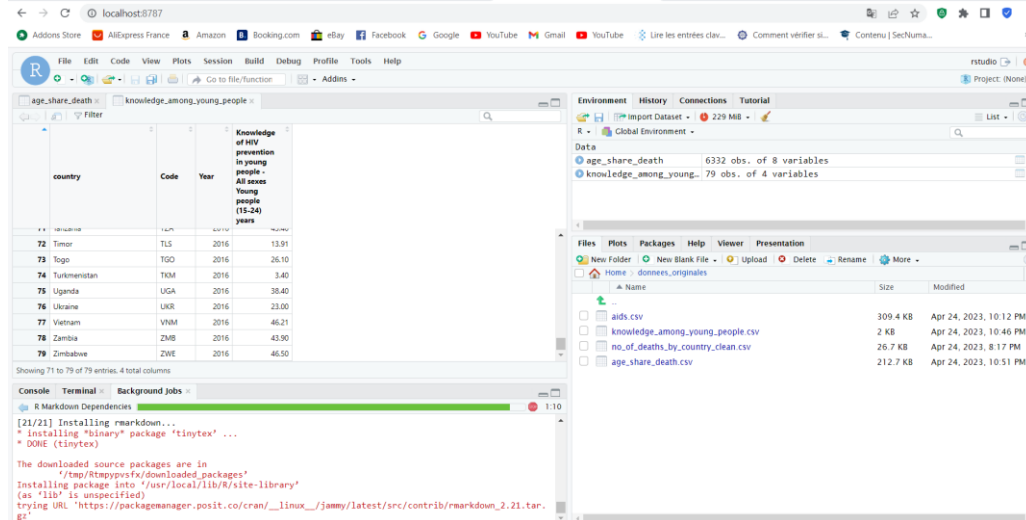
Dans cette partie du projet, nous allons accéder aux services de rstudio grâce à un mappage de ports. Nous allons mapper un port dans docker avec un port de la machine locale, c'est-à-dire que nous allons relier un port du conteneur vers un port de l'ordinateur, afin d'exécuter un conteneur docker à partir de l'image 'rocker/rstudio'. Le port 8787 du conteneur est mappé avec le port 8787 d'un ordinateur en local. La commande **docker container ls** permet de voir sur quels ports de la machine sont exposé un conteneur.

```
PS C:\I2_Flora\SEMESTRE_2\Big_Data\Projet_Big_data\dossier_image_R_projet\mon_image_run> docker run -p 8787:8787 -e PASS
WORD=monmotdepasse --name mon_image_rstudio rocker/rstudio
[s6-init] making user provided files available at /var/run/s6/etc...exited 0.
[s6-init] ensuring user provided files have correct perms...exited 0.
[fix-attrs.d] applying ownership & permissions fixes...
[fix-attrs.d] done.
[cont-init.d] executing container initialization scripts...
[cont-init.d] 01_set_env: executing...
skipping /var/run/s6/container_environment/HOME
skipping /var/run/s6/container_environment/PASSWORD
skipping /var/run/s6/container_environment/RSTUDIO_VERSION
[cont-init.d] 01_set_env: exited 0.
[cont-init.d] 02_userconf: executing...
[cont-init.d] 02_userconf: exited 0.
[cont-init.d] done.
[services.d] starting services
[services.d] done.
```


3.2.2. Connection à Rstudio

Nom d'utilisateur: **rstudio**; Mot de passe= **monmotdepasse**

Obtenir l'environnement de travail utilisé à l'intérieur du container docker :



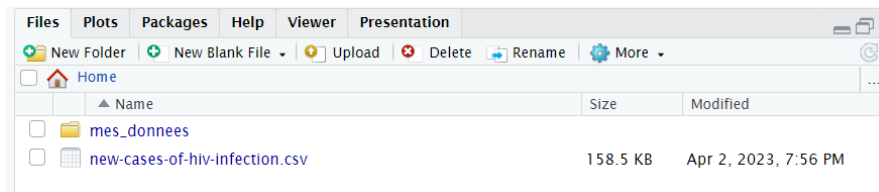
3.2.3. Obtenir les données à partir de l'ordinateur vers docker

Nous allons créer un dossier nommé « mes_données » et charger les data sets à l'intérieur :

- Ouvrir l'invite de commande
- Exécuter la commande suivante : `docker cp chemin_vers _le_fichier_source nom_du_container:/home/rstudio`. Dans le cadre de nos analyses : **`docker cp C:\I2_Flora\SEMESTRE_2\Big_Data\Projet_Big_data\data_set\originaux\new-cases-of-hiv-infection.csv mon_image_rstudio:/home/rstudio`**

```
Windows PowerShell
specified.
PS C:\I2_Flora\SEMESTRE_2\Big_Data\Projet_Big_data\dossier_image_R_projet> docker cp C:\I2_Flora\SEMESTRE_2\Big_Data\Projet_Big_data\data_set\originaux\new-cases-of-hiv-infection.csv mon_image_rstudio:/home/donnees_originales
CreateFile C:\I2_Flora\SEMESTRE_2\Big_Data\Projet_Big_data\data_set\originaux\new-cases-of-hiv-infection.csv: The system cannot find the file specified.
PS C:\I2_Flora\SEMESTRE_2\Big_Data\Projet_Big_data\dossier_image_R_projet> docker cp C:\I2_Flora\SEMESTRE_2\Big_Data\Projet_Big_data\data_set\originaux\new-cases-of-hiv-infection.csv mon_image_rstudio:/home/rstudio
CreateFile C:\I2_Flora\SEMESTRE_2\Big_Data\Projet_Big_data\data_set\originaux\new-cases-of-hiv-infection.csv: The system cannot find the file specified.
PS C:\I2_Flora\SEMESTRE_2\Big_Data\Projet_Big_data\dossier_image_R_projet> docker cp C:\I2_Flora\SEMESTRE_2\Big_Data\Projet_Big_data\data_set\originaux\mes_dataset_a_moi\new-cases-of-hiv-infection.csv mon_image_rstudio:/home/rstudio/donnees_originales
PS C:\I2_Flora\SEMESTRE_2\Big_Data\Projet_Big_data\dossier_image_R_projet> docker cp C:\I2_Flora\SEMESTRE_2\Big_Data\Projet_Big_data\data_set\originaux\mes_dataset_a_moi\new-cases-of-hiv-infection.csv mon_image_rstudio:/home/rstudio/donnees_originales
CreateFile C:\I2_Flora\SEMESTRE_2\Big_Data\Projet_Big_data\data_set\originaux\mes_dataset_a_moi\new-cases-of-hiv-infection.csv: The system cannot find the file specified.
PS C:\I2_Flora\SEMESTRE_2\Big_Data\Projet_Big_data\dossier_image_R_projet> docker cp C:\I2_Flora\SEMESTRE_2\Big_Data\Projet_Big_data\data_set\originaux\mes_dataset_a_moi\new-cases-of-hiv-infection.csv mon_image_rstudio:/home/rstudio/donnees_originales
PS C:\I2_Flora\SEMESTRE_2\Big_Data\Projet_Big_data\dossier_image_R_projet>
```

Résultat :



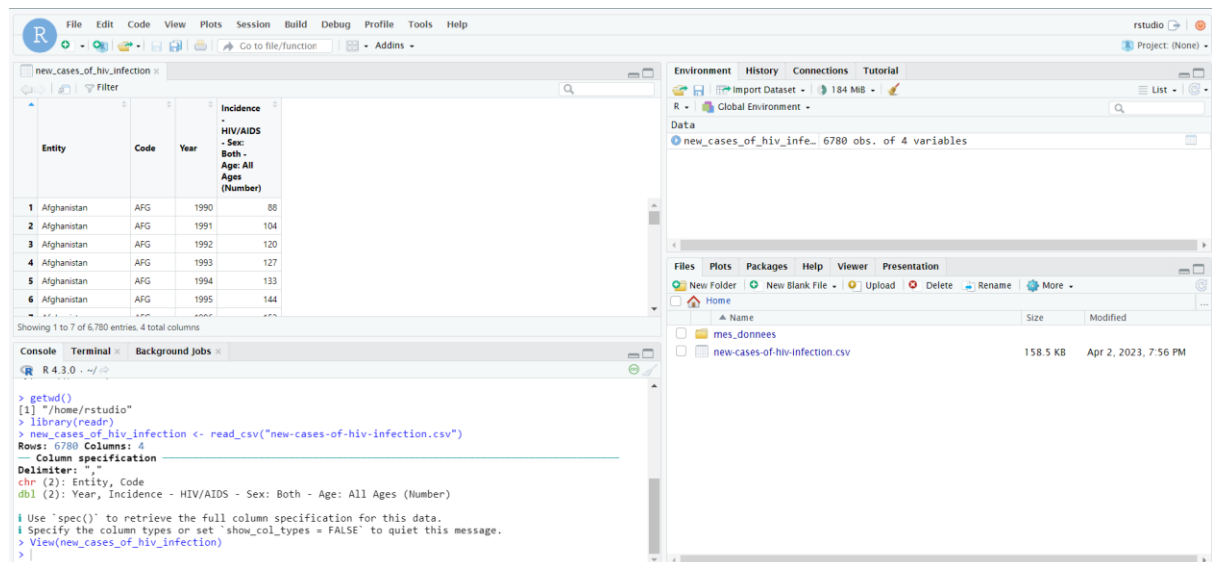
Nous allons répéter cette commande pour tous nos data set.

3.2.4. Construction du data lake

Pour construire le data Lake dans rstudio, nous avons créé deux répertoires dans l'environnement de travail de rstudio partitionné de la manière suivante :

- Un répertoire donnees originales pour tous les différents dataset qui seront analysé
- Un répertoire donnees_analyses qui va stocker les résultats issus des analyses

Il est nécessaire d'importer nos fichiers dans l'environnement R avec la fonction `read_csv` qui permet de lire un fichier CSV et de stocker les données qu'il contient dans un objet.



3.2.5. Charger les résultats des analyses dans l'ordinateur

Utiliser la commande suivante : `Docker cp source destination` ; en remplaçant « source » et « destinations » par les chemins d'accès du fichiers sources vers sa destination dans le pc

Exemple : `docker cp mon_image_rstudio:/home/rstudio/ new_cases_of_hiv_infection C:\I2_Flora\SEMESTRE_2\Big_Data\Projet_Big_data\data_set\donnees_analyses`

3.2.6. Nettoyage des données avec rstudio

- Examiner le type de colonnes

```
# Nettoyage des donnees "Knowledge-among-young-people"

```{r}
Changer le fichier CSV dans RStudio
data <- read.csv("../donnees originales/knowledge_among_young_people.csv")

Afficher la structure des données
str(data)

Afficher le contenu de chaque colonne
View(data)

```
```

- Vérifier s'il y a des valeurs manquantes

```
```{r}
Vérifier s'il y a des valeurs manquantes
anyNA(data)
```
```

```
[1] FALSE
```

- Supprimer les valeurs manquantes ou colonnes vides
- Identifier les colonnes redondantes ou inutiles et les supprimer

```
# on peut maintenant verifier s'il y'a des colonnes redondantes
```{r}
library(caret)
Séparation des données en numériques et non numériques
num_data <- data[, sapply(data, is.numeric)]
non_num_data <- data[, sapply(data, negate(is.numeric))]

Vérification des colonnes redondantes pour les valeurs numériques
find_redundant_cols <- function(data) {
 cor_matrix <- cor(data, use = "complete.obs")
 redundant_cols <- findCorrelation(cor_matrix, cutoff = 0.9, verbose = FALSE)
 return(redundant_cols)
}

Chunk 11
```

```

####{r}
Afficher les noms de colonnes avant la suppression
print(names(num_data))

#definition de la fonction
find_redundant_cols <- function(num_data) {
 cor_mat <- cor(num_data, use = "complete.obs")
 diag(cor_mat) <- 0
 redundant_cols <- apply(cor_mat, 2, function(x) any(abs(x) > 0.8))
 return(redundant_cols)
}

Supprimer les colonnes numeriques redondantes
redundant_cols <- find_redundant_cols(num_data)
data <- data[, !redundant_cols]

Afficher les noms de colonnes après la suppression
print(names(num_data))
####

```

Chunk 11

Terminal Background Jobs

4.3.0

- Renommer des colonnes
- Renvoyer les données nettoyées vers le pc

```

PS C:\I2_Flora\SEMESTRE_2\Big_Data\Projet_Big_data\dossier_image_R_projet> docker cp mon_image_rstudio:/home/rstudio/donnees_analyses/knowledge_among_young_people_nettoyé.csv C:\I2_Flora\SEMESTRE_2\Big_Data\Projet_Big_data\data_set\donnees_analyses\csv_analyses
PS C:\I2_Flora\SEMESTRE_2\Big_Data\Projet_Big_data\dossier_image_R_projet>

```

### 3.2.7. Analyses avec rstudio

Nous allons effectuer une régression linéaire des data set et répondre aux questions suivantes : quelles populations seront les plus touchées par le sida dans le monde? Qu'est-ce que les pays de ces populations ont en commun?

Tout d'abord, nous avons les données qui vont de 1990 à 2022. Nous souhaitons faire une analyse de donnée entre 2018 et 2022. Nous utilisons la fonction « subset() » de R pour sélectionner les données entre 2017 et 2019.

Nous allons séparer ces données en données d'entraînement et de test car il ne s'agit pas de mémoriser les caractéristiques de l'ensemble des données mais d'apprendre des généralisations qui peuvent être appliqués à d'autres modèles

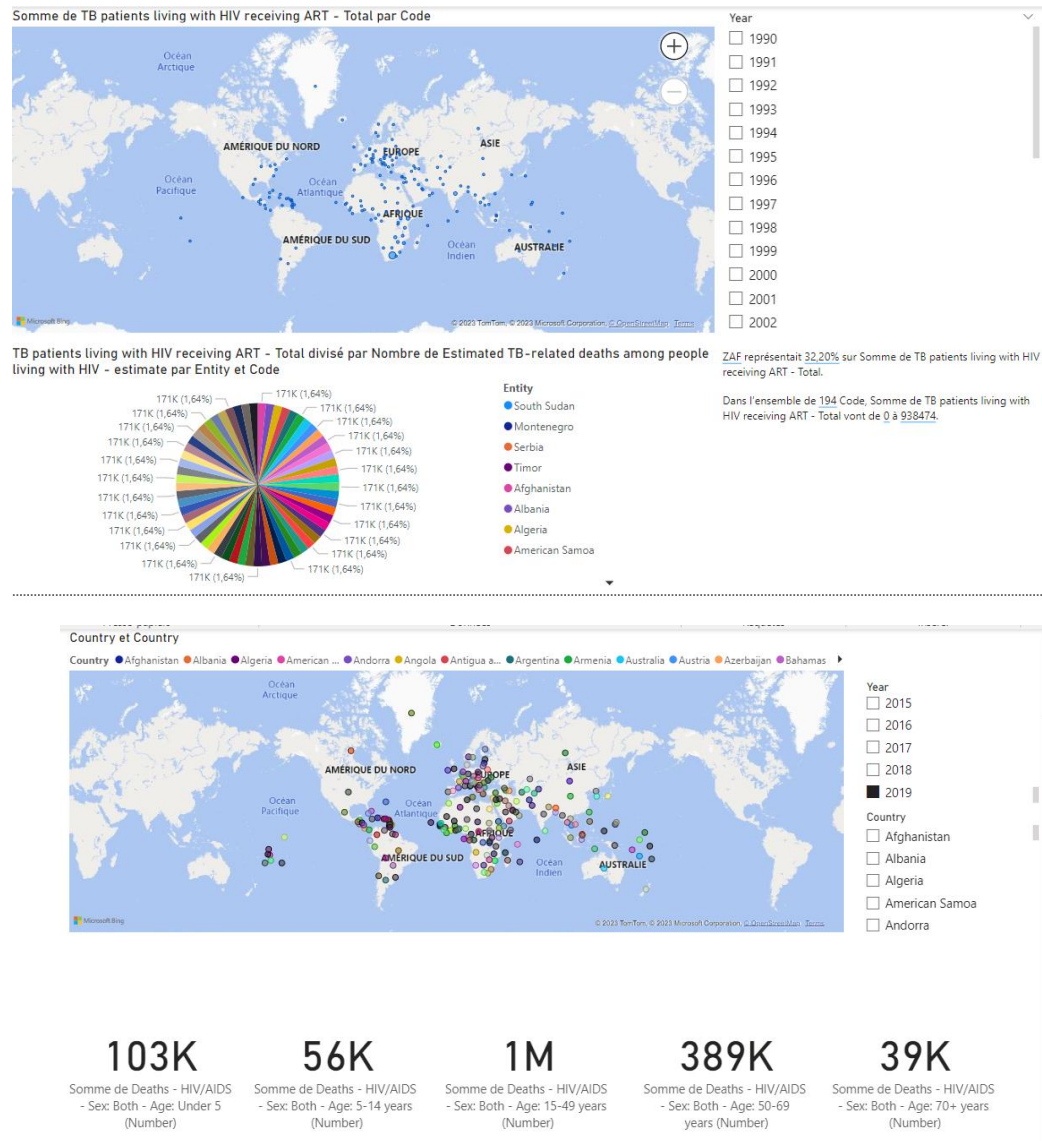
### 3.2.8. Création des visuels avec power bi

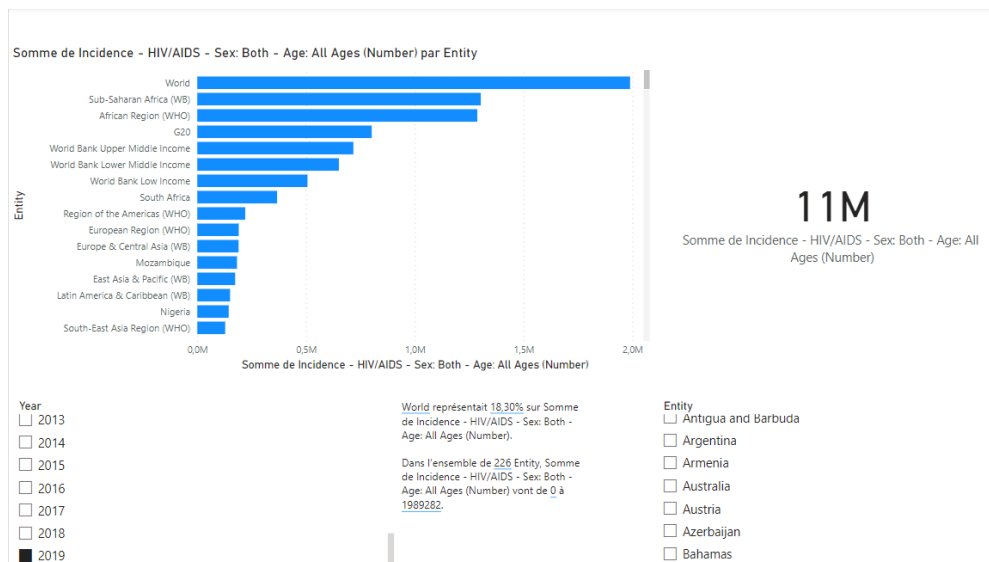
Avant de visualiser les données, il est important de les nettoyer. Nous allons donc utiliser rstudio pour préparer nos données avant de créer les visuels.

- Ouvrir rstudio et importer les différents fichiers csv et analyser leur contenu (les colonnes, vérifier les données manquantes et les supprimer, renommer les colonnes)
- Faire une copie des dataset afin de garder les fichiers originaux
- Sauvegarder dans un fichier csv
- Ouvrir power bi et importer le fichiers csv
- Créer les différents visuels

## 4. VISUALISATION DES RÉSULTATS

La visualisation des données est un outil précieux pour l'interprétation des données. Afin de rendre notre analyse plus accessible et compréhensible, nous avons décidé d'élaborer des Reporting par l'outil Power BI afin de mieux représenter les résultats d'analyse.





## SOURCES

- <https://www.kaggle.com/datasets/imdevskp/hiv-aids-dataset/discussion>
- <https://www.kaggle.com/datasets/suvojithaldar/aids-dataset>
- <https://www.kaggle.com/datasets/programmerdai/hiv-aids>
- <https://www.kaggle.com/datasets/imdevskp/hiv-aids-dataset?resource=download>
- [https://www.kaggle.com/datasets/neharaute/hivaids?select=antiviral\\_therapy\\_among\\_people.csv](https://www.kaggle.com/datasets/neharaute/hivaids?select=antiviral_therapy_among_people.csv)
- <https://www.who.int/data/collections>
- [https://www.who.int/data/gho/data/indicators/indicator-details/GHO/new-hiv-infections-\(per-1000-uninfected-population\)](https://www.who.int/data/gho/data/indicators/indicator-details/GHO/new-hiv-infections-(per-1000-uninfected-population))
- [https://www.who.int/data/gho/data/indicators/indicator-details/GHO/estimated-antiretroviral-therapy-coverage-among-people-living-with-hiv-\(-\)](https://www.who.int/data/gho/data/indicators/indicator-details/GHO/estimated-antiretroviral-therapy-coverage-among-people-living-with-hiv-(-))
- <https://www.who.int/data/gho/data/indicators/indicator-details/GHO/reported-number-of-people-receiving-antiretroviral-therapy>