In [1]:

```
from bs4 import BeautifulSoup
import requests
import json

#Utilizaremos la url de búsqueda para buscar las enfermedades que realmente nos interesan
#partimos de la url original que es : https://data.europa.eu/euodp/es/data/publisher/ecdc pero añadimos
# "?q="  para poder realizar la búsqueda
url = 'https://data.europa.eu/euodp/es/data/publisher/ecdc?q='

#variables que necesitamos:
#enfermedades que queremos recuperar y estudiar
enf_evaluar = ('Zika', 'Dengue', 'Chikungunya')

#Años que vamos a evaluar
year = ('2017', '2018')

#lista donde guardaremos las urls de los datos
download_file = []

#Realizamos try/catch para detectar errores, realizamos catch de la excepción
try:

    #Buscamos todas las enfermedades a estudiar
    for i, enf in enumerate(enf_evaluar):

        #concatenamos la enfermedad a la url para ir directamente a la enfermedad buscada
        broken_html = requests.get(url+enf_evaluar[i])
        broken_html.raise_for_status()

        # Pasamos el contenido HTML de la web a un objeto BeautifulSoup()
        html = BeautifulSoup(broken_html.text, "html.parser")

        # Obtenemos todos los li donde están las entradas
        entradas = html.find_all('li', {'class': 'search-result-item'})

        #recorremos los <li> con class: search-result-item  que es donde tenemos las enfermedades y sus links
        for i, entrada in enumerate(entradas):

            #Cogemos el texto para evaluar enfermedades. Con el método "getText()" no nos devuelve el HTML
            enfermedad = entrada.find('a', {'class': 'item_link'}).find('strong').get_text()
            #print('enfermedad: ', enfermedad)

            # REcuperamos el link que tenemos en el tag <a>
            item_link = entrada.find('a', {'class': 'item_link'}).get('href')
            #print('item_link: ', item_link)

            #Para cada enfermedada que queremos
            for i, enf in enumerate(enf_evaluar):

                #Evaluamos si el texto contiene alguna de nuestras enfermedades
                if (enfermedad.count(enf) > 0):

                    #url de la enfermedad para ir a buscar las descargas: le pasamos el
```

```
link obtenido anteriormente
                    url_enf = item_link
                    req_html = requests.get(url_enf)

                    #página de la enfermedad
                    enfermedad_html = BeautifulSoup(req_html.text, "html.parser")

                    #recuperamos las entradas donde tenemos la lista de los link de des
cargas
                    entradas_enf = enfermedad_html.find('ul', {'class': 'resource-list
 unstyled'}).find_all('span')

                    #Recorremos la lista de links de descarga:
                    for i, ent_enf in enumerate(entradas_enf):

                        #Recuperamos el texto que acompaña a cada 'a' con class:name, e
s donde tenemos el texto
                        descargas_text = ent_enf.find('a', {'class': 'name'})

                   # como hemos recuperado todos los 'span', el que queremos 'a class:n
ame' está dentro de un span determinado
                   #por tanto, al recorrerlos todos no todos contendrán dicho tag, por
 tanto, al recuperar no tendremos nada
                   #por no cumplir que contengan dicho tag.
                        #Filtramos si tenemos tag 'a' con class: name
                        if (descargas_text is not None):

                            #recuperamos el texto de tag que tenemos: sí guardamos la
 'descripción'
                            descargas_text = descargas_text.get_text()

                     #Cogemos solo los años que queremos evaluar, se ha comprobado que
 los años se encuentra en la descripción
                      # y los ficheros que queremos evaluar que son los que contengan 'E
pidemiological Report for'
                            if((descargas_text.count(year[0]) > 0 or descargas_text.cou
nt(year[1]) > 0)
                                                    and descargas_text.count('Epidemiological
Report for')):

                                    #print('descargas_text :', descargas_text)

                                   # Obtenemos todos los links donde tendremos los dato
s
                                   # para ello lo que hacemos es recuperar el tag anter
ior 'span' al que estamos trabajando.
                                   # (en la web primero tenemos el link y después la de
scripción de ese link, al evaluar la
                                   #descripción primero ya nos hemos pasado el link y p
or tanto lo recuperamos con previous)
                                    span_link = ent_enf.find_previous_sibling('span')

                                    #nos quedamos solo con la url
                                    descargas_link = span_link.find('a').get('href')
                                    #print('descarga link: ', descargas_link)

                                    if(descargas_link.count('pdf') > 0):
                                        # Guardamos la url de referencia pdf
                                        download_file.append(descargas_link)

                                    else:
```

```
                                        #Leemos el link que nos lleva a donde tenemos e
l link de download
                                        descarga = requests.get(descargas_link)
                                          # Pasamos el contenido HTML de la web a un ob
jeto BeautifulSoup()
                                        descarga_html = BeautifulSoup(descarga.text, "h
tml.parser")



                                        # Guardamos la url de referencia fichero
                                        download_file.append(descarga_html.find('a', {
'data-placement': "bottom"}).get('href'))

    print('\n download_file: ', download_file)


#else:
#    print("error request: ", broken_html.status_code)
except requests.exceptions.HTTPError as errh:
    print ("Http Error:",errh)
except requests.exceptions.ConnectionError as errc:
    print ("Error Connecting:",errc)
except requests.exceptions.Timeout as errt:
    print ("Timeout Error:",errt)
except requests.exceptions.RequestException as err:
    print ("OOps: Something Else",err)
```

```
 download_file:  ['https://www.ecdc.europa.eu/sites/default/files/document
s/AER_for_2017-Zika-virus-disease.pdf', 'https://www.ecdc.europa.eu/sites/
default/files/documents/Zika-annual-epidemiological-report-2018.pdf', 'htt
p://ecdc.europa.eu/sites/portal/files/documents/dengue-annual-epidemiologi
cal-report-2017.pdf', 'https://www.ecdc.europa.eu/sites/default/files/docu
ments/dengue-annual-epidemiological-report-2018.pdf', 'http://ecdc.europa.
eu/sites/portal/files/documents/chikungunya-virus-disease-annual-epidemiol
ogical-report-2017.pdf', 'https://www.ecdc.europa.eu/sites/default/files/d
ocuments/chikungunya-annual-epidemiological-report-2018.pdf']
```

In [2]:

```
##############################################################################################
############
#
#           Ahora definimos una serie de funciones que vamos a utilizar
#
##############################################################################################
############

# Función para Transformar las columnas
def column_transform(table):
    import pandas as pd
    import numpy as np

    df = table

    #Renombramos las columnas
    df.rename(columns=lambda x: x.replace('Unnamed: ', 'Class'), inplace=True)

    #columnas con nulos
    null_columns=df.columns[df.isnull().any()]
    sum_nan = df[null_columns].isnull().sum()       #contamos nulos de cada columna
    nrow = df.shape[0]  #numero de filas

    for i, j in enumerate(sum_nan):
        if (j == nrow):
            #eliminamos la columna con nulos en la segunda fila
            col = null_columns[i]     # el nombre de  la columna que queremos eliminar
            del df[col]

    #miramos si tenemos el primer país de las tablas
    if (df.iloc[0,0] != 'Austria'):

            df.iloc[0,:] = df.iloc[0,:].fillna('Rate')
            #Ponemos la primera fila como encabezado del dataframe (la primera fila no
 son los datos)
            df.columns = df.loc[0]

            #Reorganizamos y eliminamos la primera fila que ahora la tenemos duplicada
            df = df.reindex(df.index.drop(0))


    return df


# Función para renombrar las columnas iguales con el año que le toca
def column_rename(table, year):

    #guardamos la tabla
    df = table

    #df = column_transform(df)

    col_find = None
    #Comprobamos si existe 'Confirmed' como columna. Evaluamos la columna transformada
 a string
    for cols in df.columns.values:
         if (str(cols).count('Confirmed') > 0): col_find = cols

    Switch = False
```

```
    # Miramos si la búsqueda ha dado resultado
    if (not pd.isnull(col_find)):
        fin = df.columns.get_loc(col_find) - 1     # si exsite le damos el valor que oc
upa esta columna
    else:
        fin = len(df.columns) - 1    # en caso contrario nos quedamos con el total de c
olumnas - 1 (ya que python empieza en 0)
        Switch = True

    #Cambiamos los nombres de las columnas repetidas para poder eliminarlas
    #variables necesarias
    a = year
    year_array=[]
    # El máximo rango para los años es 5 porque siempre tendremos 4 años atrás respecto
al año que tratamos
    if fin > 5:
        max = 5
    else:
        max = fin

    for i in range(max):
        a = year - ((max - 1) - i)       #el más pequeño primero
        year_array.append(str(a))   # lo convertimos en string para concatenar

    # inicializamos variables
    j = 0
    cols = []
    #A partir de la primera columna hasta la columna 'Confirmed' (o última columna) ren
ombramos las columnas
    #ya que tienen el mismo nombre y las distinguiremos por años
    #  Vamos hacia atràs
    for i,column in enumerate(df.columns):
        if (i != 0 and i <= fin):
            #si la columna no tiene valor nulo la renombramos añadiendo el año
            cols.append(f'{column}_{year_array[j]}')

            #numeramos siempre que switch esté activado
            if (Switch):
                j = j + 1
            else:
                if (i%2 == 0):  j = j + 1
            continue
        cols.append(column)

    df.columns = cols


    return df

#Función para eliminar filas (filas primeras que no contienen paises)
def row_delete(table):

    df = table

    #reseteamos índices
    df = df.reset_index(drop=True)

    #borramos filas que no tengan el primer país
    for i in range(4):
        if (df.iloc[0,0] != 'Austria'):
                #eliminamos  fila
```

```
                df = df.drop([i])

    #reseteamos índices
    df = df.reset_index(drop=True)

    return df

#Función para dividir columnas que se hayan recuperados dobles
def column_split(table, year):
    import pandas as pd
    import numpy as np

    #inicializamos variables
    df = table
    df2 = []

    #numero de columnas que tenemos
    length_col = len(df.columns)

    #Empezamos en el cero y queremos la columna anterior a la última
    col = length_col - 2

    #Si la columna anterior a la última se puede dividir en 2, es que tenemos 2 columna
s dentro
    if len(df.iloc[0, col].split(' ')) > 1:
        #inicializamos variables
        c1 = []
        c2 = []
        # Los nulos los ponemos como espacios (evitamos errores, pero los dejamos local
izables)
        df = df.fillna(' ')

        #Recorremos todas las filas de la columna doble y generamos dos columnas indepe
ndientes
        for i in range(df.shape[0]):
            c1.append(str(df.iloc[i, col].split(' ')[0]))
            c2.append(str(df.iloc[i, col].split(' ')[1]))

        #Creamos un nuevo dataframe para añadir las nuevas columnas:
        #primero nos quedamos con las columnas anteriores a la duplicada
        df2 = df.iloc[:,0:col]

        #añadimos c1 como columna siguiente (tendrá el mismo nombre columna doble)
        df2[df.columns[col]] = c1

        #Añadimos c2 como columna siguiente (en este caso damos otro nombre, el nombres
es sacado de los pdf's)
        df2[f'ASR_{year}'] = c2

        #Añadimos la última columna del dataframe original
        df2[df.columns[(length_col - 1)]] = df.iloc[:,-1]

        #devolvemos el dataframe resultante
        df = df2

    return df
```

In [5]:

```
#Hemos de instalar tabula-py si no lo tenemos instalado
#!pip3 install tabula-py

#Nos importamos la librería necesaria
import tabula
import pandas as pd
import numpy as np


#!java -version ---> Nos hemos de asegurar que tenemos  Java instalado y en variables d
e entorno

#En la página dos tenemos la tabla.
#Lattice = True obliga a extraer los PDF mediante la extracción en modo reticular.
#Reconoce cada una de las celdas en función de líneas de control o bordes de cada celd
a.
#url consultada: "https://aegis4048.github.io/parse-pdf-files-while-retaining-structure
-with-tabula-py"

for i in range(len(download_file)):
    pdf_file_path = download_file[i]
    print('url pdf:', pdf_file_path)
    df_result = []
    table = []

    #Para Zika 2018 tenemos que seguir otro formato de extracción
    if (pdf_file_path.count('Zika') > 0 and pdf_file_path.count('2018') > 0):
        table = tabula.read_pdf(pdf_file_path, lattice=True,pages = "2,3", guess = Fals
e)
    else:
        table = tabula.read_pdf(pdf_file_path, pages = "2,3", multiple_tables=True)

    if (pdf_file_path.count('2017') > 0):
        year = 2017
    else:
        if(pdf_file_path.count('2018') > 0):
            year = 2018

    #La tabla es una lista de 2 elementos. En uno de los dos elementos tenemos la tabla
    #print(table)
    if (type(table) is list):
        for j in range(len(table)):
            if (isinstance(table[j], pd.DataFrame) and len(table[j].columns) > 3):
                df_result=table[j]
                df_result=column_transform(df_result)

                df_result=column_rename(df_result,year)

                df_result=row_delete(df_result)

                df_result=column_split(df_result,year)

    #renombramos primera columna
    df_result = df_result.rename(columns={df_result.columns[0]: 'Country'})

    filename = pdf_file_path.split('/')[7].split('.')[0] + '.csv'
    df_result.to_csv(filename,  sep='\t', encoding='utf-8')
    print('Guardado csv: ', filename)
```

```
print('\n\n Ya tienes todas tus tablas\n\n --\t DISFURTA  ---')
```

```
url pdf: https://www.ecdc.europa.eu/sites/default/files/documents/AER_for_
2017-Zika-virus-disease.pdf
Guardado csv:  AER_for_2017-Zika-virus-disease.csv
url pdf: https://www.ecdc.europa.eu/sites/default/files/documents/Zika-ann
ual-epidemiological-report-2018.pdf
Guardado csv:  Zika-annual-epidemiological-report-2018.csv
url pdf: http://ecdc.europa.eu/sites/portal/files/documents/dengue-annual-
epidemiological-report-2017.pdf
Guardado csv:  dengue-annual-epidemiological-report-2017.csv
url pdf: https://www.ecdc.europa.eu/sites/default/files/documents/dengue-a
nnual-epidemiological-report-2018.pdf
Guardado csv:  dengue-annual-epidemiological-report-2018.csv
url pdf: http://ecdc.europa.eu/sites/portal/files/documents/chikungunya-vi
rus-disease-annual-epidemiological-report-2017.pdf

Got stderr: abr 04, 2020 12:35:06 PM org.apache.pdfbox.pdmodel.font.PDCIDF
ontType2 <init>
INFORMACIÓN: OpenType Layout tables used in font ABCDEE+Tahoma,Bold are no
t implemented in PDFBox and will be ignored
abr 04, 2020 12:35:07 PM org.apache.pdfbox.pdmodel.font.PDCIDFontType2 <in
it>
INFORMACIÓN: OpenType Layout tables used in font ABCDEE+Tahoma,Bold are no
t implemented in PDFBox and will be ignored
abr 04, 2020 12:35:07 PM org.apache.pdfbox.pdmodel.font.PDCIDFontType2 <in
it>
INFORMACIÓN: OpenType Layout tables used in font ABCDEE+Tahoma,Bold are no
t implemented in PDFBox and will be ignored
abr 04, 2020 12:35:07 PM org.apache.pdfbox.pdmodel.font.PDCIDFontType2 <in
it>
INFORMACIÓN: OpenType Layout tables used in font ABCDEE+Tahoma are not imp
lemented in PDFBox and will be ignored
abr 04, 2020 12:35:07 PM org.apache.pdfbox.pdmodel.font.PDCIDFontType2 <in
it>
INFORMACIÓN: OpenType Layout tables used in font ABCDEE+Tahoma are not imp
lemented in PDFBox and will be ignored
abr 04, 2020 12:35:07 PM org.apache.pdfbox.pdmodel.font.PDCIDFontType2 <in
it>
INFORMACIÓN: OpenType Layout tables used in font ABCDEE+Tahoma,Bold are no
t implemented in PDFBox and will be ignored
abr 04, 2020 12:35:07 PM org.apache.pdfbox.pdmodel.font.PDCIDFontType2 <in
it>
INFORMACIÓN: OpenType Layout tables used in font ABCDEE+Tahoma are not imp
lemented in PDFBox and will be ignored
abr 04, 2020 12:35:07 PM org.apache.pdfbox.pdmodel.font.PDCIDFontType2 <in
it>
INFORMACIÓN: OpenType Layout tables used in font ABCDEE+Tahoma,Bold are no
t implemented in PDFBox and will be ignored


Guardado csv:  chikungunya-virus-disease-annual-epidemiological-report-201
7.csv
url pdf: https://www.ecdc.europa.eu/sites/default/files/documents/chikungu
nya-annual-epidemiological-report-2018.pdf
Guardado csv:  chikungunya-annual-epidemiological-report-2018.csv


 Ya tienes todas tus tablas

 --      DISFURTA  ---
```