Votre titre

Votre nom

La date du jour

Contents

1 Préface

Le but est ici de reproduire des graphes semblables à ceux du South China Morning Post (SCMP), sur la page The Coronavirus Pandemic et qui montrent pour différents pays le nombre cumulé (c'est-à-dire le nombre total de cas depuis le début de l'épidémie) de personnes atteintes de la maladie à coronavirus 2019. Les données que nous utiliserons dans un premier temps sont

compilées par le Johns Hopkins University Center for Systems Science and Engineering (JHU CSSE) et sont mises à disposition sur GitHub. C'est plus particulièrement sur les données time **seriescovid19confirmedglobal.csv** (des suites chronologiques au format csv) disponibles à l'adresse: `https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covid_19_data/csse_covid_19_time_series/time_series_covid19_confirmed_global.csv`

2 Jeu de données

2.1 Téléchargement du jeu de données

Premièrement, nous allons télécharger les données en local pour permettre à nos analyses d'être effectuées, même si l'URL venait à être obsolète. Ensuite nous vérifions que les données ont bien été téléchargés en local sous le nom: SARS-COV-GLOB.csv avec l'aide du package *pandas* dans python.

```
import pandas as pd

df = pd.read_csv('SARS-COV-GLOB.csv')
```

2.2 Visualisation des données

Nous pouvons regarder à quoi ressemble le jeu de données.

```
df['Country/Region' == 'China']

Traceback (most recent call last):
  File "/home/fabien.duez/.local/lib/python3.10/site-packages/pandas/core/indexes/base
    return self._engine.get_loc(casted_key)
  File "index.pyx", line 153, in pandas._libs.index.IndexEngine.get_loc
  File "index.pyx", line 182, in pandas._libs.index.IndexEngine.get_loc
  File "pandas/_libs/hashtable_class_helper.pxi", line 7081, in pandas._libs.hashtable
  File "pandas/_libs/hashtable_class_helper.pxi", line 7089, in pandas._libs.hashtable
KeyError: False

The above exception was the direct cause of the following exception:

Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "/home/fabien.duez/.local/lib/python3.10/site-packages/pandas/core/frame.py", li
```

```
    indexer = self.columns.get_loc(key)
  File "/home/fabien.duez/.local/lib/python3.10/site-packages/pandas/core/indexes/base
    raise KeyError(key) from err
KeyError: False
```

Nous avons dans le jeu de donnée **brute** les colonnes suivantes:

1. La province/ l'état
2. Le pays/ la région
3. La latitude
4. La longitude
5. les données temporelles avec le nombre de cas par jour

3 Transformation des données

3.1 Exclusion de certains pays

Nous allons regarder l'évolution du nombre de cas cumulé au cours du temps pour le spays suivants:la Belgique (Belgium), la Chine - toutes les provinces sauf Hong-Kong (China), Hong Kong (China, Hong-Kong), la France métropolitaine (France), l'Allemagne (Germany), l'Iran (Iran), l'Italie (Italy), le Japon (Japan), la Corée du Sud (Korea, South), la Hollande sans les colonies (Netherlands), le Portugal (Portugal), l'Espagne (Spain), le Royaume-Unis sans les colonies (United Kingdom), les États-Unis (US). De ce fait nous allons transformer le tableau de manière à seulement garder ses pays et supprimer les pays non concernés par l'analyse.

```
df2 = df.loc[df['Country/Region'].isin(['Belgium','China','France','Germany','Iran','I
```

Nous pouvons maintenant vérifier si l'exclusion de certains pays a fonctionnée.

```
df2
```

```
Province/State  Country/Region        Lat        Long  1/22/20  1/23/20  1/24/20  1/25
24                                     NaN         Belgium  50.833300    4.4699
59                                  Anhui           China  31.825700  117.2264
```

```
60                                        Beijing           China  40.182400  116.4142
61                                      Chongqing           China  30.057200  107.8740
62                                         Fujian           China  26.078900  117.9874
..                                            ...             ...        ...        .
274                                   Montserrat  United Kingdom  16.742498  -62.1873
275                             Pitcairn Islands  United Kingdom -24.376800 -128.3242
276  Saint Helena, Ascension and Tristan da Cunha  United Kingdom  -7.946700  -14.3559
277                 Turks and Caicos Islands  United Kingdom  21.694000  -71.7979
278                                       NaN  United Kingdom  55.378100   -3.4360

[75 rows x 1147 columns]
```

Cette ligne de commande nous montre simplement le début et la fin du tableau mais ne nous renseigne pas sur les pays présent. Nous pouvons regarder plus précisément dnas la colonne Country/Region si les pays demandés sont bien dans le tableau

```
df2['Country/Region'].unique()
```

```
array(['Belgium', 'China', 'France', 'Germany', 'Iran', 'Italy', 'Japan',
       'Korea, South', 'Netherlands', 'Portugal', 'Spain', 'US',
       'United Kingdom'], dtype=object)
```

Nous observons bien les pays demandés. La première transformation du jeu de données est réussie.

3.2 Exclusion des territoires hors métropole pour la France, le Royaume-Uni et la Hollande

Nous devons maintenant exclure les territoires d'outre-mer et les résidus coloniaux de la France, du Royaume-Uni et de la Hollande.

3.2.1 France

Premièrement, nous allons regarder quelles sont les territoires hors métropole présent dans le jeu de données

```
df2FR = df2.loc[df2['Country/Region'] == 'France']
print(df2FR)
```

```
Province/State Country/Region        Lat        Long  1/22/20  1/23/20  1/24/20  1/25/
120              French Guiana         France   3.933900  -53.125800        0        0
121           French Polynesia         France -17.679700  149.406800        0        0
122                 Guadeloupe         France  16.265000  -61.551000        0        0
123                 Martinique         France  14.641500  -61.024200        0        0
124                    Mayotte         France -12.827500   45.166244        0        0
125              New Caledonia         France -20.904305  165.618042        0        0
126                    Reunion         France -21.115100   55.536400        0        0
127            Saint Barthelemy         France  17.900000  -62.833300        0        0
128  Saint Pierre and Miquelon          France  46.885200  -56.315900        0        0
129                   St Martin         France  18.070800  -63.050100        0        0
130          Wallis and Futuna         France -14.293800 -178.116500        0        0
131                        NaN         France  46.227600    2.213700        0        0

[12 rows x 1147 columns]
```

Nous pouvons voir que la france métropolitaine occupe la dernière ligne de ce tableau (**Province/State = NaN**). Voici la liste des territoires à enlever: French Guiana, French Polynesia, Guadeloupe, Martinique, Mayotte, New Caledonia, Reunion, Saint Barthelemy, Saint Pierre and Miquelon, St Martin, Wallis and Futuna.

```
df3 = df2.loc[~df2['Province/State'].isin(['French Guiana','French Polynesia','Guadelo
print(df3.loc[df3['Country/Region'] == 'France'])
```

```
Province/State Country/Region      Lat    Long  1/22/20  1/23/20  1/24/20  1/25/20  1/
131            NaN         France  46.2276  2.2137        0        0        2        3

[1 rows x 1147 columns]
```

Il ne reste bien que la France métropolitaine! Nous pouvons passer au Royaume-Uni.

3.2.2 Royaume-Uni

Nous allons reprendre la même méthode que pour la France.

```
df2RU = df2.loc[df2['Country/Region'] == 'United Kingdom']
print(df2RU)
```

```
Province/State  Country/Region        Lat        Long  1/22/20  1/23/20  1/24/20  1/25
264                                   Anguilla  United Kingdom  18.220600  -63.0686
265                                    Bermuda  United Kingdom  32.307800  -64.7505
266                      British Virgin Islands  United Kingdom  18.420700  -64.6400
267                              Cayman Islands  United Kingdom  19.313300  -81.2546
268                            Channel Islands  United Kingdom  49.372300   -2.3644
269                  Falkland Islands (Malvinas)  United Kingdom -51.796300  -59.5236
270                                  Gibraltar  United Kingdom  36.140800   -5.3536
271                                   Guernsey  United Kingdom  49.448196   -2.5894
272                                 Isle of Man  United Kingdom  54.236100   -4.5481
273                                      Jersey  United Kingdom  49.213800   -2.1358
274                                   Montserrat  United Kingdom  16.742498  -62.1873
275                               Pitcairn Islands  United Kingdom -24.376800 -128.3242
276  Saint Helena, Ascension and Tristan da Cunha  United Kingdom  -7.946700  -14.3559
277                       Turks and Caicos Islands  United Kingdom  21.694000  -71.7979
278                                           NaN  United Kingdom  55.378100   -3.4360

[15 rows x 1147 columns]
```

```
df4 = df3.loc[~df3['Province/State'].isin(['Anguilla','Bermuda','British Virgin Islands
print(df4.loc[df4['Country/Region'] == 'United Kingdom'])
```

```
Province/State  Country/Region      Lat   Long  1/22/20  1/23/20  1/24/20  1/25/20  1/
278            NaN  United Kingdom  55.3781 -3.436        0        0        0        0

[1 rows x 1147 columns]
```

Enfin nous allons nous occuper de la Hollande.

3.2.3 Hollande

```
df2HO = df2.loc[df2['Country/Region'] == 'Netherlands']
print(df2HO)
```

```
Province/State Country/Region      Lat     Long  1/22/20  1/23/20  1/24/20  1/25/20  1
196                             Aruba    Netherlands  12.5211 -69.9683        0
197  Bonaire, Sint Eustatius and Saba    Netherlands  12.1784 -68.2385        0
198                           Curacao    Netherlands  12.1696 -68.9900        0
199                       Sint Maarten    Netherlands  18.0425 -63.0548        0
200                                NaN    Netherlands  52.1326   5.2913        0

[5 rows x 1147 columns]
```

```
df5 = df4.loc[~df4['Province/State'].isin(['Aruba','Bonaire, Sint Eustatius and Saba',
print(df5.loc[df5['Country/Region'] == 'Netherlands'])
```

```
Province/State Country/Region      Lat    Long  1/22/20  1/23/20  1/24/20  1/25/20  1/
200            NaN    Netherlands  52.1326  5.2913        0        0        0        0

[1 rows x 1147 columns]
```

Nous pouvons maintenant vérifier si le jeu de données ne contient que les métropoles.

```
print(df5.loc[(df5['Country/Region'] == 'France') | (df5['Country/Region'] == 'United
```

```
Province/State  Country/Region      Lat    Long  1/22/20  1/23/20  1/24/20  1/25/20  1
131            NaN          France  46.2276  2.2137        0        0        2        3
200            NaN     Netherlands  52.1326  5.2913        0        0        0        0
278            NaN  United Kingdom  55.3781 -3.4360        0        0        0        0

[3 rows x 1147 columns]
```

Nous voyons que seul les métropoles sont présentes dans le jeu de données!

3.3 Modification des données de la région de Hong-Kong

Nous allons maintenant remplacer la valeur Chine pour Hong-Kong en 'Hong-Kong' de manière à différencier la chine et ses provinces et Hong-Kong.

```
df5.loc[df5['Province/State'] == 'Hong Kong']
```

```
Province/State Country/Region   Lat   Long  1/22/20  1/23/20  1/24/20  1/25/20  1/26/20
71      Hong Kong          China  22.3  114.2        0        2        2        5

[1 rows x 1147 columns]
```

Nous voyons que la ligne de Hong-Kong correspond à la ligne 71 et que pour l'instant, la valeur de la colonne **Country/Region** correspond à la Chine. Nous allons pouvoir modifier cette ligne!

```
df5.loc[[71], ['Country/Region']] = "Hong Kong"
```

Nous pouvons vérifier si la ligne contenant Hong- Kong a changé de valeur pour la colonne Country/Region

```
df5.loc[df5['Country/Region'] == 'Hong Kong']
```

```
Province/State Country/Region   Lat   Long  1/22/20  1/23/20  1/24/20  1/25/20  1/26/2
71      Hong Kong      Hong Kong  22.3  114.2        0        2        2        5
```

```
[1 rows x 1147 columns]
```

Parfait! La valeur a bien été remplacée. nous pouvons dorénavant nous occuper d'enlever les colonnes qui ne sont pas nécessaires.

3.4 Retrait des colonnes inutiles pour l'analyse

Dans notre tableau certaines colonnes comme les province, la latitude et la longitude ne seront pas utile lors de l'analyse nous pouvons donc les supprimer du jeu de données.

```
df6 = df5.drop(['Province/State','Lat','Long'], axis=1)
```

Nous pouvons maintenant vérifier si les colonnes que nous voulions supprimer sont absentes du tableau.

```
df6[:5]
```

```
Country/Region  1/22/20  1/23/20  1/24/20  1/25/20  1/26/20  1/27/20  1/28/20  1/29/20
24        Belgium        0        0        0        0        0        0        0
59          China        1        9       15       39       60       70      106
60          China       14       22       36       41       68       80       91
61          China        6        9       27       57       75      110      132
62          China        1        5       10       18       35       59       80
```

```
[5 rows x 1144 columns]
```

Il ne reste que les colonnes **Country/Region** et les **colonnes temporelles**.

3.5 Modifications du tableau

Le tableau que nous avons pour l'instant ne nous permet pas de produire des courbes du nombre de cas cumulés par pays. De ce fait nous devons transformer le tableau pour donner :

Pays	Date	Nombre-de-cas
Belgique	01/01/2020	0
Belgique	02/01/2020	4
Belgique	03/01/2020	56
.	.	.
.	.	.
.	.	.

```
df7 = pd.melt(df6, id_vars='Country/Region',var_name='Date', value_name='Nombre_de_cas
print(df7)

Country/Region     Date  Nombre_de_cas
0              Belgium  1/22/20              0
1                China  1/22/20              1
2                China  1/22/20             14
3                China  1/22/20              6
4                China  1/22/20              1
...       ...             ...
52573     Netherlands   3/9/23        8599981
52574        Portugal   3/9/23        5570473
52575           Spain   3/9/23       13770429
52576              US   3/9/23      103802702
52577  United Kingdom   3/9/23       24425309

[52578 rows x 3 columns]
```

Nous devons maintenant transformer la date donnée dès le départ en date compréhensible par python pour permettre de trier en fonction en fonction du pays et en fonction de la date nos nombre de cas.

```
df7['Date'] = pd.to_datetime(df7['Date'], format='%m/%d/%y')

print(df7)

Country/Region       Date  Nombre_de_cas
0              Belgium 2020-01-22              0
1                China 2020-01-22              1
2                China 2020-01-22             14
3                China 2020-01-22              6
4                China 2020-01-22              1
...         ...             ...
```

```
52573     Netherlands 2023-03-09        8599981
52574        Portugal 2023-03-09        5570473
52575           Spain 2023-03-09       13770429
52576              US 2023-03-09      103802702
52577  United Kingdom 2023-03-09       24425309

[52578 rows x 3 columns]
```

Les dates sont mises à jour et sont comprises par python nous pouvons maintenant trier le tableau.

```
df8 = df7.sort_values(by=['Country/Region','Date'])
print(df8)
```

```
Country/Region       Date  Nombre_de_cas
0              Belgium 2020-01-22              0
46             Belgium 2020-01-23              0
92             Belgium 2020-01-24              0
138            Belgium 2020-01-25              0
184            Belgium 2020-01-26              0
...        ...            ...
52393  United Kingdom 2023-03-05       24396530
52439  United Kingdom 2023-03-06       24396530
52485  United Kingdom 2023-03-07       24396530
52531  United Kingdom 2023-03-08       24396530
52577  United Kingdom 2023-03-09       24425309

[52578 rows x 3 columns]
```

Le tableau est trié!

3.6 Données manquantes ?

Nous allons maintenant vérifier de potentielles données manquantes dans notre tableau.

```
NA = df8.isna().sum()
print(NA)
```

```
Country/Region    0
Date              0
Nombre_de_cas     0
dtype: int64
```

Aucune donnée semble manquante nous allons pouvoir passer à la création des graphiques. Mais avant nous allons séparer les différents pays en différents tableaux. Premièrement nous allons regarder quels sont les différents pays. Cela a déjà été fait plus haut mais nous allons le refaire ci-dessous:

3.7 Subdivision du jeu de données

```
df8['Country/Region'].unique()
```

```
array(['Belgium', 'China', 'France', 'Germany', 'Hong Kong', 'Iran',
       'Italy', 'Japan', 'Korea, South', 'Netherlands', 'Portugal',
       'Spain', 'US', 'United Kingdom'], dtype=object)
```

Nous allons créer un tableaux pour chaque pays ci-dessus:

```
dfBel = df8[df8["Country/Region"] == 'Belgium']
print(dfBel['Country/Region'].unique())
dfChina = df8[df8["Country/Region"] == 'China']
print(dfChina['Country/Region'].unique())
dfFR = df8[df8["Country/Region"] == 'France']
print(dfFR['Country/Region'].unique())
dfGER = df8[df8["Country/Region"] == 'Germany']
print(dfGER['Country/Region'].unique())
dfHK = df8[df8["Country/Region"] == 'Hong Kong']
print(dfHK['Country/Region'].unique())
dfIran = df8[df8["Country/Region"] == 'Iran']
print(dfIran['Country/Region'].unique())
dfIT = df8[df8["Country/Region"] == 'Italy']
print(dfIT['Country/Region'].unique())
dfJP = df8[df8["Country/Region"] == 'Japan']
print(dfJP['Country/Region'].unique())
dfSK = df8[df8["Country/Region"] == 'Korea, South']
print(dfSK['Country/Region'].unique())
dfNTH = df8[df8["Country/Region"] == 'Netherlands']
print(dfNTH['Country/Region'].unique())
dfPOG = df8[df8["Country/Region"] == 'Portugal']
print(dfPOG['Country/Region'].unique())
dfSP = df8[df8["Country/Region"] == 'Spain']
print(dfSP['Country/Region'].unique())
dfUS = df8[df8["Country/Region"] == 'US']
print(dfUS['Country/Region'].unique())
```

```
dfUK = df8[df8["Country/Region"] == 'United Kingdom']
print(dfUK['Country/Region'].unique())
```

```
['Belgium']
['China']
['France']
['Germany']
['Hong Kong']
['Iran']
['Italy']
['Japan']
['Korea, South']
['Netherlands']
['Portugal']
['Spain']
['US']
['United Kingdom']
```

Nous avons correctement subdivisé nos tableaux! Cependant les différentes provinces de la Chine n'ont toujours pas été regroupé ce qui donne plusieurs nombre de cas par jour.

```
dfChina
```

```
Country/Region       Date  Nombre_de_cas
1               China 2020-01-22              1
2               China 2020-01-22             14
3               China 2020-01-22              6
4               China 2020-01-22              1
5               China 2020-01-22              0
...         ...             ...
52562           China 2023-03-09           1647
52563           China 2023-03-09        1521816
52564           China 2023-03-09           3089
52565           China 2023-03-09           9743
52566           China 2023-03-09          11848

[37719 rows x 3 columns]
```

Nous allons donc aggréger le tableau pour n'avoir qu'une valeur par jour!

```
dfC = dfChina.groupby(['Country/Region','Date'],as_index= False).sum().reset_index()
print(dfC)
```

```
index Country/Region       Date  Nombre_de_cas
0         0          China 2020-01-22            548
1         1          China 2020-01-23            641
2         2          China 2020-01-24            918
3         3          China 2020-01-25           1401
4         4          China 2020-01-26           2067
...             ...         ...             ...
1138   1138          China 2023-03-05        2027418
1139   1139          China 2023-03-06        2027418
1140   1140          China 2023-03-07        2027418
1141   1141          China 2023-03-08        2027418
1142   1142          China 2023-03-09        2027418

[1143 rows x 4 columns]
```

Nous allons enfin pouvoir représenter graphiquement le jeu de données. Prenons d'abord un exemple avec le Japon!

4 Représentation graphique

```
import matplotlib.pyplot as plt
import matplotlib.axes as axs


plt.plot('Date', 'Nombre_de_cas', data=dfJP)
plt.xlabel('Date')
plt.ylabel('Nombre de cas')
plt.suptitle('Nombre de cas au Japon')

plt.savefig('JPL.png')
```

La représentation graphique est sauvegardée dans le dossier avec le document computationnel! Nous allons maintenant essayer avec une échelle logarithmique.

```
plt.plot('Date', 'Nombre_de_cas', data=dfJP)
plt.semilogy()
plt.xlabel('Date')
plt.ylabel('Nombre de cas')
plt.suptitle('Nombre de cas au Japon')

plt.savefig('JPLOG.png')
```

Nous allons pouvoir passer au plot de chaque pays! TOut d'abord avec une échelle linéaire:

```
plt.plot('Date', 'Nombre_de_cas', data=dfBel, label='Belgium')
plt.plot('Date', 'Nombre_de_cas', data=dfC, label='China')
plt.plot('Date', 'Nombre_de_cas', data=dfFR, label='France')
plt.plot('Date', 'Nombre_de_cas', data=dfGER, label='Germany')
plt.plot('Date', 'Nombre_de_cas', data=dfHK, label='Hong Kong')
plt.plot('Date', 'Nombre_de_cas', data=dfIran, label='Iran')
plt.plot('Date', 'Nombre_de_cas', data=dfIT, label='Italy')
plt.plot('Date', 'Nombre_de_cas', data=dfJP, label='Japan')
plt.plot('Date', 'Nombre_de_cas', data=dfSK, label='South Korea')
plt.plot('Date', 'Nombre_de_cas', data=dfNTH, label='Netherlands')
plt.plot('Date', 'Nombre_de_cas', data=dfPOG, label='Portugal')
plt.plot('Date', 'Nombre_de_cas', data=dfSP, label='Spain')
plt.plot('Date', 'Nombre_de_cas', data=dfUS, label='US')
plt.plot('Date', 'Nombre_de_cas', data=dfUK, label='United Kingdom')

plt.xlabel('Date')
plt.ylabel('Nombre de cas')
plt.legend()

plt.savefig('DATAL.png')
```

et efin avec une échelle logarithmique.

```
plt.plot('Date', 'Nombre_de_cas', data=dfBel, label='Belgium')
plt.plot('Date', 'Nombre_de_cas', data=dfC, label='China')
plt.plot('Date', 'Nombre_de_cas', data=dfFR, label='France')
plt.plot('Date', 'Nombre_de_cas', data=dfGER, label='Germany')
plt.plot('Date', 'Nombre_de_cas', data=dfHK, label='Hong Kong')
plt.plot('Date', 'Nombre_de_cas', data=dfIran, label='Iran')
```

```
plt.plot('Date', 'Nombre_de_cas', data=dfIT, label='Italy')
plt.plot('Date', 'Nombre_de_cas', data=dfJP, label='Japan')
plt.plot('Date', 'Nombre_de_cas', data=dfSK, label='South Korea')
plt.plot('Date', 'Nombre_de_cas', data=dfNTH, label='Netherlands')
plt.plot('Date', 'Nombre_de_cas', data=dfPOG, label='Portugal')
plt.plot('Date', 'Nombre_de_cas', data=dfSP, label='Spain')
plt.plot('Date', 'Nombre_de_cas', data=dfUS, label='US')
plt.plot('Date', 'Nombre_de_cas', data=dfUK, label='United Kingdom')

plt.semilogy()
plt.xlabel('Date')
plt.ylabel('Nombre de cas')
plt.legend()

plt.savefig('DATALOG.png')
```

Tous les graphiques doivent être dans votre dossier où se situe le document computationnel.