Analysis of French data

This program imports the files generated by the parser (divided by month to put less load on the memory) and analyses them. It is **not language agnostic:** correct linguistic settings must be specified in **"setting up", "NLP" and "additional rules".**

First some additional rules for NER are defined. Some are general, some are language-specific, as specified in the relevant section.

The files are opened and preprocessed, then lemma frequency and NER frequency are calculated per each month and in the whole corpus. **important:** in case of empty months (so, when analysing less than one year of data) **remember to exclude them from the mean,** otherwise the mean will be distorted by the empty months.

All the dataframes are exported as CSV files for further analisys or for data visualization.

Setting up

Remember to check the folder paths.

In [1]:

```
%%capture
from tqdm.notebook import tqdm as tqdm #for progress bars
tqdm().pandas()
```

In [2]:

```
from pathlib import Path
import os
import pandas as pd
import spacy
from collections import Counter
from datetime import datetime

# Measure execution time
start_time = datetime.now()

# folder paths (1. containing a substet homogeneous by language and divided by date; 2.
for exports)
folder = Path("C://Users/copam/Desktop/jupyter test/exports_parser/FR")
export = Path("C://Users/copam/Desktop/jupyter test/exports_NLP/FR")

# month files (if need be, add other months here and in the list below).
january = open(os.path.join(folder, "1.txt"),encoding="utf8").read()
february = open(os.path.join(folder, "2.txt"),encoding="utf8").read()
march = open(os.path.join(folder, "3.txt"),encoding="utf8").read()
april = open(os.path.join(folder, "4.txt"),encoding="utf8").read()
may = open(os.path.join(folder, "5.txt"),encoding="utf8").read()
june = open(os.path.join(folder, "6.txt"),encoding="utf8").read()
july = open(os.path.join(folder, "7.txt"),encoding="utf8").read()
august = open(os.path.join(folder, "8.txt"),encoding="utf8").read()
september = open(os.path.join(folder, "9.txt"),encoding="utf8").read()
october = open(os.path.join(folder, "10.txt"),encoding="utf8").read()
november = open(os.path.join(folder, "11.txt"),encoding="utf8").read()
december = open(os.path.join(folder, "12.txt"),encoding="utf8").read()

months = [january,february,march, april, may, june, july, august, september, october, n
ovember, december]
```

NLP

Remember to check the language and the max_lenght. References on models here: https://spacy.io/models (https://spacy.io/models)

In [3]:

```
nlp = spacy.load('fr_core_news_md')
spacy_stopwords = spacy.lang.fr.stop_words.STOP_WORDS
nlp.max_length = 100000000
```

Additional rules for COVID19 NER

Remember to adapt for the specific language (below the comment). References here: https://spacy.io/usage/rule-based-matching#models-rules (https://spacy.io/usage/rule-based-matching#models-rules)

In [4]:

```
from spacy.pipeline import EntityRuler
ruler = EntityRuler(nlp)
ruler.overwrite_ents = True
patterns = [{"label": "COVID19", "pattern": "Covid"},
            {"label": "COVID19", "pattern": "covid"},
            {"label": "COVID19", "pattern": "Covid19"},
            {"label": "COVID19", "pattern": "covid19"},
            {"label": "COVID19", "pattern": "Covid 19"},
            {"label": "COVID19", "pattern": "Covid-19"},
            {"label": "COVID19", "pattern": "covid-19"},
            {"label": "COVID19", "pattern": "covid 19"},
            {"label": "COVID19", "pattern": "Corvid"},
            {"label": "COVID19", "pattern": "corvid"},
            {"label": "COVID19", "pattern": "Corvid19"},
            {"label": "COVID19", "pattern": "corvid19"},
            {"label": "COVID19", "pattern": "Corvid 19"},
            {"label": "COVID19", "pattern": "corvid 19"},
            {"label": "COVID19", "pattern": "Coronavirus"},
            {"label": "COVID19", "pattern": "coronavirus"},
            {"label": "COVID19", "pattern": "Corona virus"},
            {"label": "COVID19", "pattern": "Corona Virus"},
            {"label": "COVID19", "pattern": "corona virus"},
            {"label": "COVID19", "pattern": "COVID"},
            {"label": "COVID19", "pattern": "COVID19"},
            {"label": "COVID19", "pattern": "COVID 19"},
            {"label": "COVID19", "pattern": "2019-nCoV"},
            {"label": "COVID19", "pattern": "ncov"},
            {"label": "COVID19", "pattern": "nCoV"},
            {"label": "COVID19", "pattern": "sars"},
            {"label": "COVID19", "pattern": "SARS"},
            {"label": "COVID19", "pattern": "SARS-CoV2"},
## language-specific rules
## consider adding rules for scarce resources allocation, anxiety, ...
            {"label": "COVID19r", "pattern": "virus de Wuhan"},
            {"label": "COVID19r", "pattern": "Virus de Wuhan"},
            {"label": "COVID19r", "pattern": "virus chinois"},
            {"label": "COVID19r", "pattern": "Virus chinois"},
            {"label": "COVID19r", "pattern": "virus de la Chine"},
            {"label": "COVID19r", "pattern": "Virus de la Chine"}
           ]
ruler.add_patterns(patterns)
nlp.add_pipe(ruler)
```

In [5]:

```
file_doc = {}
for x in tqdm(months):
    file_doc[x] = nlp(x)
```

Preprocessing

In [6]:

```
# Definition of the preprocessing functions
def is_token_allowed(token):
    if (not token or not token.string.strip() or token.is_stop or token.is_punct or tok
en in spacy_stopwords):
        return False
    return True

def preprocess_token(token):
    if is_token_allowed:
        return token.lemma_.strip().lower()
```

In [7]:

```
# Actual preprocessing
complete_filtered_tokens = {}
for x in tqdm(months):
    complete_filtered_tokens[x] = [preprocess_token(token) for token in file_doc[x] if
is_token_allowed(token)]
```

Lemma frequency

calculates and exports lemma frequency, in general and per month.

In [8]:

```
lemmas_freq = {}
for x in tqdm(months):
    lemmas_freq[x] = Counter(complete_filtered_tokens[x]).most_common()
```

In [9]:

```
## january
lemmas_freq_january = lemmas_freq[january]
df_lemmas_freq_january = pd.DataFrame(lemmas_freq_january, columns={'Lemma':[1],'Count'
:[2]})
df_lemmas_freq_january.index += 1
df_lemmas_freq_january.to_csv(os.path.join(export, "lemmas\lemmas-frequency-1.csv"))

## february
lemmas_freq_february = lemmas_freq[february]
df_lemmas_freq_february = pd.DataFrame(lemmas_freq_february, columns={'Lemma':[1],'Coun
t':[2]})
df_lemmas_freq_february.index += 1
df_lemmas_freq_february.to_csv(os.path.join(export, "lemmas\lemmas-frequency-2.csv"))

## march
lemmas_freq_march = lemmas_freq[march]
df_lemmas_freq_march = pd.DataFrame(lemmas_freq_march, columns={'Lemma':[1],'Count':[2
]})
df_lemmas_freq_march.index += 1
df_lemmas_freq_march.to_csv(os.path.join(export, "lemmas\lemmas-frequency-3.csv"))

## april
lemmas_freq_april = lemmas_freq[april]
df_lemmas_freq_april = pd.DataFrame(lemmas_freq_april, columns={'Lemma':[1],'Count':[2
]})
df_lemmas_freq_april.index += 1
df_lemmas_freq_april.to_csv(os.path.join(export, "lemmas\lemmas-frequency-4.csv"))

## may
lemmas_freq_may = lemmas_freq[may]
df_lemmas_freq_may = pd.DataFrame(lemmas_freq_may, columns={'Lemma':[1],'Count':[2]})
df_lemmas_freq_may.index += 1
df_lemmas_freq_may.to_csv(os.path.join(export, "lemmas\lemmas-frequency-5.csv"))

## june
lemmas_freq_june = lemmas_freq[june]
df_lemmas_freq_june = pd.DataFrame(lemmas_freq_june, columns={'Lemma':[1],'Count':[2]})
df_lemmas_freq_june.index += 1
df_lemmas_freq_june.to_csv(os.path.join(export, "lemmas\lemmas-frequency-6.csv"))

## july
lemmas_freq_july = lemmas_freq[july]
df_lemmas_freq_july = pd.DataFrame(lemmas_freq_july, columns={'Lemma':[1],'Count':[2]})
df_lemmas_freq_july.index += 1
df_lemmas_freq_july.to_csv(os.path.join(export, "lemmas\lemmas-frequency-7.csv"))

## august
lemmas_freq_august = lemmas_freq[august]
df_lemmas_freq_august = pd.DataFrame(lemmas_freq_august, columns={'Lemma':[1],'Count':[
2]})
df_lemmas_freq_august.index += 1
df_lemmas_freq_august.to_csv(os.path.join(export, "lemmas\lemmas-frequency-8.csv"))

## september
lemmas_freq_september = lemmas_freq[september]
df_lemmas_freq_september = pd.DataFrame(lemmas_freq_september, columns={'Lemma':[1],'Co
unt':[2]})
df_lemmas_freq_september.index += 1
df_lemmas_freq_september.to_csv(os.path.join(export, "lemmas\lemmas-frequency-9.csv"))
```

```
## october
lemmas_freq_october = lemmas_freq[october]
df_lemmas_freq_october = pd.DataFrame(lemmas_freq_october, columns={'Lemma':[1],'Count'
:[2]})
df_lemmas_freq_october.index += 1
df_lemmas_freq_october.to_csv(os.path.join(export, "lemmas\lemmas-frequency-10.csv"))

## november
lemmas_freq_november = lemmas_freq[november]
df_lemmas_freq_november = pd.DataFrame(lemmas_freq_november, columns={'Lemma':[1],'Coun
t':[2]})
df_lemmas_freq_november.index += 1
df_lemmas_freq_november.to_csv(os.path.join(export, "lemmas\lemmas-frequency-11.csv"))

## december
lemmas_freq_december = lemmas_freq[december]
df_lemmas_freq_december = pd.DataFrame(lemmas_freq_december, columns={'Lemma':[1],'Coun
t':[2]})
df_lemmas_freq_december.index += 1
df_lemmas_freq_december.to_csv(os.path.join(export, "lemmas\lemmas-frequency-12.csv"))
```

Trends of the lemmas per month

"general" takes the data from the whole corpus. "mean" is the mean of the months.

Important: in case of empty months (so, when analysing less than one year of data) **remember to exclude them from the mean!**

In [10]:

```
# List of all lemma dataframes
df_lemmas_freq_all = [df_lemmas_freq_january,
                      df_lemmas_freq_february,
                      df_lemmas_freq_march,
                      df_lemmas_freq_april,
                      df_lemmas_freq_may,
                      df_lemmas_freq_june,
                      df_lemmas_freq_july,
                      df_lemmas_freq_august,
                      df_lemmas_freq_september,
                      df_lemmas_freq_october,
                      df_lemmas_freq_november,
                      df_lemmas_freq_december]

# Loop for index and series
L = []
for x in df_lemmas_freq_all:
    x = x.set_index('Lemma')
    L.append(pd.Series(x.values.tolist(), index=x.index))

# All together
df_lemmas_freq_all = pd.concat(L, axis=1, keys=('1','2','3','4','5','6','7','8','9','1
0','11','12'))
df_lemmas_freq_all = df_lemmas_freq_all.fillna('0')
for month in df_lemmas_freq_all:
    df_lemmas_freq_all[month] = df_lemmas_freq_all[month].str[0]

df_lemmas_freq_all = df_lemmas_freq_all.astype('int')

# Calculate the total
lemmasums = df_lemmas_freq_all.iloc[:, [0,1,2,3,4,5,6,7,8,9,10,11]].sum(axis=1)
df_lemmas_freq_all = pd.concat([df_lemmas_freq_all, lemmasums], axis = 1)
df_lemmas_freq_all = df_lemmas_freq_all.rename(columns={0: "total"})

# Calculate the mean of the months
lemmameans = df_lemmas_freq_all.iloc[:, [0,1,2,3,4,5,6]].mean(axis=1) ## In case of emp
ty months, exclude them from the mean here! Numbers are indices, where 0 is january and
11 is december
df_lemmas_freq_all = pd.concat([df_lemmas_freq_all, lemmameans], axis = 1)
df_lemmas_freq_all = df_lemmas_freq_all.rename(columns={0: "mean"})
df_lemmas_freq_all["mean"] = (df_lemmas_freq_all["mean"].astype('float')).round(2)

# Reorder and reindex
total_col = df_lemmas_freq_all.pop("total")
df_lemmas_freq_all.insert(0, "total", total_col)
df_lemmas_freq_all.reset_index(level=0, inplace=True)
df_lemmas_freq_all = df_lemmas_freq_all.sort_values(by=['total'], ascending=False)
df_lemmas_freq_all.index = pd.RangeIndex(len(df_lemmas_freq_all.index))
df_lemmas_freq_all.index += 1
df_lemmas_freq_all["lemma"] = df_lemmas_freq_all["index"]
df_lemmas_freq_all = df_lemmas_freq_all[['lemma','total','1','2','3','4','5','6','7',
'8','9','10','11','12','mean']]


# Export and display
df_lemmas_freq_all.to_csv(os.path.join(export, "lemmas\lemmas-frequency-timeseries.csv"
))
display(df_lemmas_freq_all.head(20))
```

```
<ipython-input-10-0372892eda72>:19: DeprecationWarning: The default dtype
for empty Series will be 'object' instead of 'float64' in a future versio
n. Specify a dtype explicitly to silence this warning.
  L.append(pd.Series(x.values.tolist(), index=x.index))
```

	lemma	total	1	2	3	4	5	6	7	8	9	10	11	12	mean
1	coronavirus	1849	55	252	378	456	344	194	170	0	0	0	0	0	264.14
2	covid-19	1583	0	37	190	430	409	300	217	0	0	0	0	0	226.14
3	suisse	1503	37	96	249	424	381	158	158	0	0	0	0	0	214.71
4	pays	1442	21	182	176	360	320	231	152	0	0	0	0	0	206.00
5	cas	1338	30	197	222	271	195	250	173	0	0	0	0	0	191.14
6	crise	1127	12	24	143	364	310	185	89	0	0	0	0	0	161.00
7	pandémie	1102	0	35	114	335	286	195	137	0	0	0	0	0	157.43
8	personne	1087	13	108	180	337	187	145	117	0	0	0	0	0	155.29
9	santé	1083	19	104	211	243	252	155	99	0	0	0	0	0	154.71
10	virus	1004	28	128	175	208	195	140	130	0	0	0	0	0	143.43
11	épidémie	986	36	217	235	189	155	79	75	0	0	0	0	0	140.86
12	y	962	17	59	125	285	203	154	119	0	0	0	0	0	137.43
13	oms	934	36	141	145	220	151	114	127	0	0	0	0	0	133.43
14	temps	916	8	33	158	273	226	135	83	0	0	0	0	0	130.86
15	faire	883	8	41	160	226	195	147	106	0	0	0	0	0	126.14
16	monde	857	11	95	116	204	196	146	89	0	0	0	0	0	122.43
17	mesure	847	7	70	155	244	202	107	62	0	0	0	0	0	121.00
18	il	845	15	84	116	196	195	133	106	0	0	0	0	0	120.71
19	devoir	788	12	72	116	213	164	140	71	0	0	0	0	0	112.57
20	jour	765	11	65	135	199	152	137	66	0	0	0	0	0	109.29

NER

Calculates and exports named entity frequency, in general and per month. **Remember to check the export name.** References on NER tags here: https://spacy.io/api/annotation#named-entities (https://spacy.io/api/annotation#named-entities)

In [11]:

```
entity_list = {}
for x in tqdm(months):
    entity_list[x] = []
    for ent in file_doc[x].ents:
        entity_list[x].append((ent.text, ent.label_))

entity_counts = {}
for x in tqdm(months):
    entity_counts[x] = Counter(entity_list[x]).most_common()
    if not len(entity_counts[x]) == 0:
        enticat, count = zip(*entity_counts[x])
        entity, category = zip(*enticat)
        entity_counts[x] = tuple(zip(entity, category,count))

## january
entity_counts_january = entity_counts[january]
df_entity_counts_january = pd.DataFrame(entity_counts_january, columns={'Entity':[1],'C
ategory':[2],'Count':[3]})
df_entity_counts_january.index += 1
df_entity_counts_january.to_csv(os.path.join(export, "entities\entities-frequency-1.cs
v"))

## february
entity_counts_february = entity_counts[february]
df_entity_counts_february = pd.DataFrame(entity_counts_february, columns={'Entity':[1],
'Category':[2],'Count':[3]})
df_entity_counts_february.index += 1
df_entity_counts_february.to_csv(os.path.join(export, "entities\entities-frequency-2.cs
v"))

## march
entity_counts_march = entity_counts[march]
df_entity_counts_march = pd.DataFrame(entity_counts_march, columns={'Entity':[1],'Categ
ory':[2],'Count':[3]})
df_entity_counts_march.index += 1
df_entity_counts_march.to_csv(os.path.join(export, "entities\entities-frequency-3.csv"
))

## april
entity_counts_april = entity_counts[april]
df_entity_counts_april = pd.DataFrame(entity_counts_april, columns={'Entity':[1],'Categ
ory':[2],'Count':[3]})
df_entity_counts_april.index += 1
df_entity_counts_april.to_csv(os.path.join(export, "entities\entities-frequency-4.csv"
))

## may
entity_counts_may = entity_counts[may]
df_entity_counts_may = pd.DataFrame(entity_counts_may, columns={'Entity':[1],'Category'
:[2],'Count':[3]})
df_entity_counts_may.index += 1
df_entity_counts_may.to_csv(os.path.join(export, "entities\entities-frequency-5.csv"))

## june
entity_counts_june = entity_counts[june]
df_entity_counts_june = pd.DataFrame(entity_counts_june, columns={'Entity':[1],'Categor
y':[2],'Count':[3]})
df_entity_counts_june.index += 1
df_entity_counts_june.to_csv(os.path.join(export, "entities\entities-frequency-6.csv"))
```

```
## july
entity_counts_july = entity_counts[july]
df_entity_counts_july = pd.DataFrame(entity_counts_july, columns={'Entity':[1],'Categor
y':[2],'Count':[3]})
df_entity_counts_july.index += 1
df_entity_counts_july.to_csv(os.path.join(export, "entities\entities-frequency-7.csv"))

## august
entity_counts_august = entity_counts[august]
df_entity_counts_august = pd.DataFrame(entity_counts_august, columns={'Entity':[1],'Cat
egory':[2],'Count':[3]})
df_entity_counts_august.index += 1
df_entity_counts_august.to_csv(os.path.join(export, "entities\entities-frequency-8.csv"
))

## september
entity_counts_september = entity_counts[september]
df_entity_counts_september = pd.DataFrame(entity_counts_september, columns={'Entity':[1
],'Category':[2],'Count':[3]})
df_entity_counts_september.index += 1
df_entity_counts_september.to_csv(os.path.join(export, "entities\entities-frequency-9.c
sv"))

## october
entity_counts_october = entity_counts[october]
df_entity_counts_october = pd.DataFrame(entity_counts_october, columns={'Entity':[1],'C
ategory':[2],'Count':[3]})
df_entity_counts_october.index += 1
df_entity_counts_october.to_csv(os.path.join(export, "entities\entities-frequency-10.cs
v"))

## november
entity_counts_november = entity_counts[november]
df_entity_counts_november = pd.DataFrame(entity_counts_november, columns={'Entity':[1],
'Category':[2],'Count':[3]})
df_entity_counts_november.index += 1
df_entity_counts_november.to_csv(os.path.join(export, "entities\entities-frequency-11.c
sv"))

## december
entity_counts_december = entity_counts[december]
df_entity_counts_december = pd.DataFrame(entity_counts_december, columns={'Entity':[1],
'Category':[2],'Count':[3]})
df_entity_counts_december.index += 1
df_entity_counts_december.to_csv(os.path.join(export, "entities\entities-frequency-12.c
sv"))
```

Trends of the entities per month

"general" takes the data from the whole corpus. "mean" is the mean of the months.

Important: in case of empty months (so, when analysing less than one year of data) **remember to exclude them from the mean!**

In [12]:

```
# Merging entity and category (for better indexing)
df_entity_counts_january['Entity / Category'] = df_entity_counts_january['Entity'] + '
 // ' + df_entity_counts_january['Category']
df1 = df_entity_counts_january[['Entity / Category', 'Count']]

df_entity_counts_february['Entity / Category'] = df_entity_counts_february['Entity'] +
' // ' + df_entity_counts_february['Category']
df2 = df_entity_counts_february[['Entity / Category', 'Count']]

df_entity_counts_march['Entity / Category'] = df_entity_counts_march['Entity'] + ' // '
+ df_entity_counts_march['Category']
df3 = df_entity_counts_march[['Entity / Category', 'Count']]

df_entity_counts_april['Entity / Category'] = df_entity_counts_april['Entity'] + ' // '
+ df_entity_counts_april['Category']
df4 = df_entity_counts_april[['Entity / Category', 'Count']]

df_entity_counts_may['Entity / Category'] = df_entity_counts_may['Entity'] + ' // ' + d
f_entity_counts_may['Category']
df5 = df_entity_counts_may[['Entity / Category', 'Count']]

df_entity_counts_june['Entity / Category'] = df_entity_counts_june['Entity'] + ' // ' +
df_entity_counts_june['Category']
df6 = df_entity_counts_june[['Entity / Category', 'Count']]

df_entity_counts_july['Entity / Category'] = df_entity_counts_july['Entity'] + ' // ' +
df_entity_counts_july['Category']
df7 = df_entity_counts_july[['Entity / Category', 'Count']]

df_entity_counts_august['Entity / Category'] = df_entity_counts_august['Entity'] + ' //
' + df_entity_counts_august['Category']
df8 = df_entity_counts_august[['Entity / Category', 'Count']]

df_entity_counts_september['Entity / Category'] = df_entity_counts_september['Entity']
+ ' // ' + df_entity_counts_september['Category']
df9 = df_entity_counts_september[['Entity / Category', 'Count']]

df_entity_counts_october['Entity / Category'] = df_entity_counts_october['Entity'] + '
 // ' + df_entity_counts_october['Category']
df10 = df_entity_counts_october[['Entity / Category', 'Count']]

df_entity_counts_november['Entity / Category'] = df_entity_counts_november['Entity'] +
' // ' + df_entity_counts_november['Category']
df11 = df_entity_counts_november[['Entity / Category', 'Count']]

df_entity_counts_december['Entity / Category'] = df_entity_counts_december['Entity'] +
' // ' + df_entity_counts_december['Category']
df12 = df_entity_counts_december[['Entity / Category', 'Count']]


# List of all entity dataframes
df_ent_freq_all = [df1,df2,df3,df4,df5,df6,df7,df8,df9,df10,df11,df12]

# Loop for index and series
L = []
for x in df_ent_freq_all:
    x = x.set_index('Entity / Category')
    L.append(pd.Series(x.values.tolist(), index=x.index))
```

```
# All together
df_ent_freq_all = pd.concat(L, axis=1, keys=('1','2','3','4','5','6','7','8','9','10',
'11','12'))
df_ent_freq_all = df_ent_freq_all.fillna('0')
for month in df_ent_freq_all:
    df_ent_freq_all[month] = df_ent_freq_all[month].str[0]
df_ent_freq_all = df_ent_freq_all.astype('int')

# Calculate the total
entysums = df_ent_freq_all.iloc[:, [0,1,2,3,4,5,6,7,8,9,10,11]].sum(axis=1)
df_ent_freq_all = pd.concat([df_ent_freq_all, entysums], axis = 1)
df_ent_freq_all = df_ent_freq_all.rename(columns={0: "total"})

# Calculate the mean of the months
entymeans = df_ent_freq_all.iloc[:, [0,1,2,3,4,5,6]].mean(axis=1) ## In case of empty m
onths, exclude them from the mean here! Numbers are indices, where 0 is january and 11
 is december
df_ent_freq_all = pd.concat([df_ent_freq_all, entymeans], axis = 1)
df_ent_freq_all = df_ent_freq_all.rename(columns={0: "mean"})
df_ent_freq_all["mean"] = (df_ent_freq_all["mean"].astype('float')).round(2)

# Reorder and reindex
total_col_e = df_ent_freq_all.pop("total")
df_ent_freq_all.insert(0, "total", total_col_e)
df_ent_freq_all.reset_index(level=0, inplace=True)
df_ent_freq_all = df_ent_freq_all.rename(columns={"index": "enticat"})
df_ent_freq_all = df_ent_freq_all.sort_values(by=['total'], ascending=False)
df_ent_freq_all.index = pd.RangeIndex(len(df_ent_freq_all.index))
df_ent_freq_all.index += 1
df_ent_freq_all[['entity','category']] = df_ent_freq_all.enticat.str.split(" // ",expan
d=True,)
df_ent_freq_all = df_ent_freq_all[['entity','category','total','1','2','3','4','5','6',
'7','8','9','10','11','12','mean']]

# Export and display
df_ent_freq_all.to_csv(os.path.join(export, "entities\entities-frequency-timeseries.cs
v"))
display(df_ent_freq_all.head(20))
```

```
<ipython-input-12-de8688ba0d63>:46: DeprecationWarning: The default dtype
for empty Series will be 'object' instead of 'float64' in a future versio
n. Specify a dtype explicitly to silence this warning.
  L.append(pd.Series(x.values.tolist(), index=x.index))
```

	entity	category	total	1	2	3	4	5	6	7	8	9	10	11	12	mea
1	coronavirus	COVID19	1621	51	220	326	400	304	171	149	0	0	0	0	0	231.5
2	Covid-19	MISC	1325	0	31	171	349	339	261	174	0	0	0	0	0	189.2
3	OMS	ORG	1082	44	171	165	254	173	130	145	0	0	0	0	0	154.5
4	Suisse	LOC	851	23	52	143	244	204	93	92	0	0	0	0	0	121.5
5	Chine	LOC	450	40	155	30	92	63	34	36	0	0	0	0	0	64.2
6	Etats-Unis	LOC	325	8	19	29	132	52	53	32	0	0	0	0	0	46.4
7	Genève	LOC	287	4	41	55	65	78	24	20	0	0	0	0	0	41.0
8	Conseil fédéral	ORG	238	0	2	35	70	62	51	18	0	0	0	0	0	34.0
9	Europe	LOC	210	3	28	20	75	32	39	13	0	0	0	0	0	30.0
10	Le Temps	ORG	202	3	7	40	56	42	31	23	0	0	0	0	0	28.8
11	la Chine	LOC	185	10	41	35	38	27	20	14	0	0	0	0	0	26.4
12	Italie	LOC	184	0	36	51	41	27	22	7	0	0	0	0	0	26.2
13	France	LOC	183	2	32	27	55	31	20	16	0	0	0	0	0	26.1
14	Organisation mondiale de la santé	ORG	174	4	28	31	38	28	25	20	0	0	0	0	0	24.8
15	Etats	LOC	160	0	21	32	41	29	17	20	0	0	0	0	0	22.8
16	Wuhan	LOC	151	16	54	10	31	21	8	11	0	0	0	0	0	21.5
17	SARS	MISC	144	0	0	37	51	20	13	23	0	0	0	0	0	20.5
18	CoV-2	MISC	141	0	0	35	48	22	14	22	0	0	0	0	0	20.1
19	OFSP	ORG	132	10	1	34	32	29	17	9	0	0	0	0	0	18.8
20	Allemagne	LOC	130	3	11	22	39	23	18	14	0	0	0	0	0	18.5

Locations

Remember to change the category according to the linguistic model!

In [13]:

```
df_entity_counts_location = df_ent_freq_all[df_ent_freq_all["category"] == "LOC"]
df_entity_counts_location = df_entity_counts_location.reset_index(drop=True)
df_entity_counts_location.index += 1
df_entity_counts_location.to_csv(os.path.join(export, "entities\entities-frequency-0-general-locations.csv"))
display(df_entity_counts_location.head(20))
```

	entity	category	total	1	2	3	4	5	6	7	8	9	10	11	12	mean
1	Suisse	LOC	851	23	52	143	244	204	93	92	0	0	0	0	0	121.57
2	Chine	LOC	450	40	155	30	92	63	34	36	0	0	0	0	0	64.29
3	Etats-Unis	LOC	325	8	19	29	132	52	53	32	0	0	0	0	0	46.43
4	Genève	LOC	287	4	41	55	65	78	24	20	0	0	0	0	0	41.00
5	Europe	LOC	210	3	28	20	75	32	39	13	0	0	0	0	0	30.00
6	la Chine	LOC	185	10	41	35	38	27	20	14	0	0	0	0	0	26.43
7	Italie	LOC	184	0	36	51	41	27	22	7	0	0	0	0	0	26.29
8	France	LOC	183	2	32	27	55	31	20	16	0	0	0	0	0	26.14
9	Etats	LOC	160	0	21	32	41	29	17	20	0	0	0	0	0	22.86
10	Wuhan	LOC	151	16	54	10	31	21	8	11	0	0	0	0	0	21.57
11	Allemagne	LOC	130	3	11	22	39	23	18	14	0	0	0	0	0	18.57
12	Pékin	LOC	115	6	23	6	17	31	22	10	0	0	0	0	0	16.43
13	la France	LOC	88	4	3	13	27	24	11	6	0	0	0	0	0	12.57
14	Etat	LOC	86	2	5	10	26	21	15	7	0	0	0	0	0	12.29
15	Iran	LOC	84	3	38	18	8	2	2	13	0	0	0	0	0	12.00
16	Espagne	LOC	84	1	3	10	34	20	8	8	0	0	0	0	0	12.00
17	Confédération	LOC	83	3	0	9	14	24	17	16	0	0	0	0	0	11.86
18	Berne	LOC	65	2	0	14	8	24	12	5	0	0	0	0	0	9.29
19	Afrique	LOC	65	0	5	6	18	10	18	8	0	0	0	0	0	9.29
20	Tessin	LOC	64	0	4	24	16	13	5	2	0	0	0	0	0	9.14

Persons

Remember to change the category according to the linguistic model!

In [14]:

```
df_entity_counts_person = df_ent_freq_all[df_ent_freq_all["category"] == "PER"]
df_entity_counts_person = df_entity_counts_person.reset_index(drop=True)
df_entity_counts_person.index += 1
df_entity_counts_person.to_csv(os.path.join(export, "entities\entities-frequency-0-general-persons.csv"))
display(df_entity_counts_person.head(20))
```

	entity	category	total	1	2	3	4	5	6	7	8	9	10	11	12	mean
1	Tedros Adhanom Ghebreyesus	PER	123	5	28	20	31	9	17	13	0	0	0	0	0	17.57
2	Donald Trump	PER	99	0	3	6	45	15	19	11	0	0	0	0	0	14.14
3	Alain Berset	PER	85	0	1	12	6	43	19	4	0	0	0	0	0	12.14
4	Roche	PER	61	2	1	4	25	19	5	5	0	0	0	0	0	8.71
5	Emmanuel Macron	PER	43	1	11	5	16	8	1	1	0	0	0	0	0	6.14
6	Michael Ryan	PER	37	1	7	4	3	7	9	6	0	0	0	0	0	5.29
7	Daniel Koch	PER	33	2	0	13	6	8	4	0	0	0	0	0	0	4.71
8	Xi Jinping	PER	31	5	9	0	2	14	0	1	0	0	0	0	0	4.43
9	Didier Pittet	PER	30	3	0	4	1	16	1	5	0	0	0	0	0	4.29
10	Steven Mnuchin	PER	29	0	0	0	29	0	0	0	0	0	0	0	0	4.14
11	M. Tedros	PER	24	1	4	2	6	5	3	3	0	0	0	0	0	3.43
12	Angela Merkel	PER	22	0	0	5	8	6	3	0	0	0	0	0	0	3.14
13	Karin Keller	PER	20	0	0	0	7	2	11	0	0	0	0	0	0	2.86
14	Sutter	PER	20	0	0	0	7	2	11	0	0	0	0	0	0	2.86
15	Juan Carlos	PER	18	0	0	0	18	0	0	0	0	0	0	0	0	2.57
16	Washington	PER	18	1	1	1	9	6	0	0	0	0	0	0	0	2.57
17	A. K.	PER	17	0	0	0	0	0	0	17	0	0	0	0	0	2.43
18	Jair Bolsonaro	PER	16	0	0	0	5	1	1	9	0	0	0	0	0	2.29
19	Mercredi	PER	15	0	1	2	8	4	0	0	0	0	0	0	0	2.14
20	Matthias Egger	PER	14	0	0	0	0	2	9	3	0	0	0	0	0	2.00

Organizations

Remember to change the category according to the linguistic model!

In [15]:

```
df_entity_counts_organization = df_ent_freq_all[df_ent_freq_all["category"] == "ORG"]
df_entity_counts_organization = df_entity_counts_organization.reset_index(drop=True)
df_entity_counts_organization.index += 1
df_entity_counts_organization.to_csv(os.path.join(export, "entities\entities-frequency-0-general-organizations.csv"))
display(df_entity_counts_organization.head(20))
```

	entity	category	total	1	2	3	4	5	6	7	8	9	10	11	12	mea
1	OMS	ORG	1082	44	171	165	254	173	130	145	0	0	0	0	0	154.5
2	Conseil fédéral	ORG	238	0	2	35	70	62	51	18	0	0	0	0	0	34.0
3	Le Temps	ORG	202	3	7	40	56	42	31	23	0	0	0	0	0	28.8
4	Organisation mondiale de la santé	ORG	174	4	28	31	38	28	25	20	0	0	0	0	0	24.8
5	OFSP	ORG	132	10	1	34	32	29	17	9	0	0	0	0	0	18.8
6	AFP	ORG	119	2	11	24	26	23	21	12	0	0	0	0	0	17.0
7	ONU	ORG	75	0	4	15	31	5	14	6	0	0	0	0	0	10.7
8	MSF	ORG	63	0	0	6	4	16	25	12	0	0	0	0	0	9.0
9	Office fédéral de la santé publique	ORG	59	3	0	20	20	9	5	2	0	0	0	0	0	8.4
10	UE	ORG	54	0	4	4	12	16	8	10	0	0	0	0	0	7.7
11	Union européenne	ORG	48	1	7	2	13	12	8	5	0	0	0	0	0	6.8
12	EPFL	ORG	45	0	1	2	24	8	6	4	0	0	0	0	0	6.4
13	BNS	ORG	45	5	0	14	7	2	17	0	0	0	0	0	0	6.4
14	Novartis	ORG	41	2	1	4	13	15	2	4	0	0	0	0	0	5.8
15	OIT	ORG	40	0	0	0	10	9	21	0	0	0	0	0	0	5.7
16	FMI	ORG	37	0	6	2	17	3	8	1	0	0	0	0	0	5.2
17	Google	ORG	34	1	1	6	19	6	1	0	0	0	0	0	0	4.8
18	Sanofi	ORG	33	0	0	8	8	14	1	2	0	0	0	0	0	4.7
19	l'ONU	ORG	33	1	11	5	5	5	2	4	0	0	0	0	0	4.7
20	FDA	ORG	33	0	0	2	22	7	2	0	0	0	0	0	0	4.7

COVID19

Remember to change the category according to the linguistic model!

In [16]:

```
df_entity_counts_COVID19 = df_ent_freq_all[df_ent_freq_all["category"] == "COVID19"]
df_entity_counts_COVID19 = df_entity_counts_COVID19.reset_index(drop=True)
df_entity_counts_COVID19.index += 1
df_entity_counts_COVID19.to_csv(os.path.join(export, "entities\entities-frequency-0-general-COVID19.csv"))
display(df_entity_counts_COVID19.head(20))
```

	entity	category	total	1	2	3	4	5	6	7	8	9	10	11	12	mea
1	coronavirus	COVID19	1621	51	220	326	400	304	171	149	0	0	0	0	0	231.5
2	Covid	COVID19	19	0	0	0	3	8	2	6	0	0	0	0	0	2.7
3	2019-nCoV	COVID19	18	7	8	1	1	0	1	0	0	0	0	0	0	2.5
4	Covid-19	COVID19	14	0	0	1	3	4	1	5	0	0	0	0	0	2.0
5	SARS	COVID19	8	0	0	3	1	2	1	1	0	0	0	0	0	1.1
6	Coronavirus	COVID19	6	1	0	2	1	1	1	0	0	0	0	0	0	0.8
7	covid	COVID19	4	0	0	0	1	2	1	0	0	0	0	0	0	0.5
8	covid-19	COVID19	3	0	0	0	0	0	0	3	0	0	0	0	0	0.4
9	covid19	COVID19	3	0	0	0	1	0	0	2	0	0	0	0	0	0.4
10	corona virus	COVID19	2	0	0	0	0	0	1	1	0	0	0	0	0	0.2
11	nCoV	COVID19	2	0	2	0	0	0	0	0	0	0	0	0	0	0.2
12	covid 19	COVID19	1	0	0	0	0	0	1	0	0	0	0	0	0	0.1

COVID19r

Remember to change the category according to the linguistic model!

In [17]:

```
df_entity_counts_COVID19r = df_ent_freq_all[df_ent_freq_all["category"] == "COVID19r"]
df_entity_counts_COVID19r = df_entity_counts_COVID19r.reset_index(drop=True)
df_entity_counts_COVID19r.index += 1
df_entity_counts_COVID19r.to_csv(os.path.join(export, "entities\entities-frequency-0-general-COVID19r.csv"))
display(df_entity_counts_COVID19r.head(20))
```

	entity	category	total	1	2	3	4	5	6	7	8	9	10	11	12	mean
1	virus chinois	COVID19r	1	1	0	0	0	0	0	0	0	0	0	0	0	0.14

In [18]:

```
end_time = datetime.now()
print('Data elaborated in {}'.format(end_time - start_time))
```

```
Data elaborated in 0:02:03.074263
```