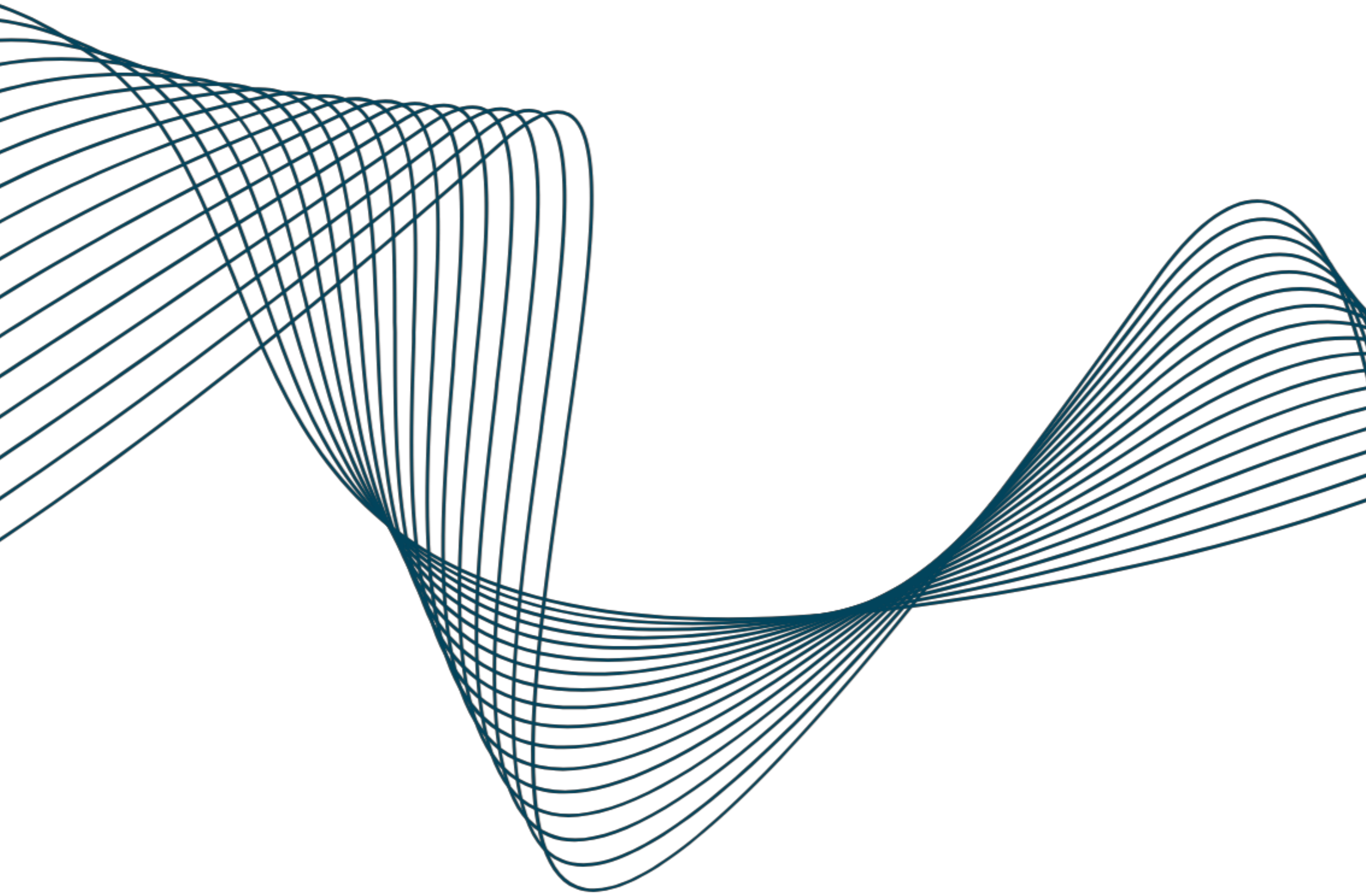
DANE

Project

Forecasting on death causes

Data

Cleaning



Import desired libraries

[1]:

```
import geopy
import pandas as pd
import datetime as dt
import scipy as sp
import matplotlib.pyplot as plt
from neuralprophet import NeuralProphet, set_log_level
import seaborn as sns
import numpy as np
import matplotlib.ticker as plticker
sns.set_theme(style="ticks")
plt.rcParams["font.family"] = "Montserrat";
import missingno as msno
import plotly.express as px
```

Import DANE data

Data was obtained from: DANE Estadisticas Vitales

[2]:

```
data_2008 = pd.read_csv('C:/data/Defun_2008.csv', sep=";", dtype=object,
             encoding = "mbcs")

data_2009 = pd.read_csv('C:/data/Defun_2009.csv', sep=";", dtype=object,
             encoding = "mbcs")

data_2010 = pd.read_csv('C:/data/Defun_2010.csv', sep=";", dtype=object,
             encoding = "mbcs")

data_2011 = pd.read_csv('C:/data/Defun_2011.csv', sep=";", dtype=object,
             encoding = "mbcs")

data_2012 = pd.read_csv('C:/data/Defun_2012.csv', sep=";", dtype=object,
             encoding = "mbcs")

data_2013 = pd.read_csv('C:/data/Defun_2013.csv', sep=";", dtype=object,
             encoding = "mbcs")

data_2014 = pd.read_csv('C:/data/Defun_2014.csv', sep=";", dtype=object,
             encoding = "mbcs")

data_2015 = pd.read_csv('C:/data/Defun_2015.csv', sep=";", dtype=object,
             encoding = "mbcs")

data_2016 = pd.read_csv('C:/data/Defun_2016.csv', sep=";", dtype=object,
             encoding = "mbcs")

data_2017 = pd.read_csv('C:/data/Defun_2017.csv', sep=";", dtype=object,
             encoding = "mbcs")

data_2018 = pd.read_csv('C:/data/Defun_2018.csv', sep=";", dtype=object,
             encoding = "mbcs")

data_2019 = pd.read_csv('C:/data/Defun_2019.csv', sep=",", dtype=object,
             encoding = "mbcs")
```

```
data_2020 = pd.read_csv('C:/data/Defun_2020.csv', sep=",", dtype=object,
              encoding = "mbcs")
```

Variable descriptions

- **COD_DPTO**: DANE codes of the political-administrative division of Colombia (departments).
- **COD_MUNIC**: DANE codes of the political-administrative division of Colombia (municipalities).
- **SIT_DEFUN**: Place where death took place.
  - 1: Hospital/clinic
  - 2: Health center/health center
  - 3: Home/domicile
  - 4: Place of work
  - 5: Public road
  - 6: Other
  - 9: No information
- **ANO**: Year of death.
- **MES**: Month of death.
- **SEXO**: Sex of deceased.
  - 1: Male
  - 2: Female
  - 3: Undertermined
- **EST_CIVIL**: Civil status.
  - 1: Unmarried and had been living with partner for two or more years
  - 2: Not married and had been living with a partner for less than two years less than two years living with partner
  - 3: Separated, divorced, divorced
  - 4: Widowed
  - 5: Was single
  - 6: Married
  - 9: No information
- **GRU_ED1**: Age of deceased.
  - 00: Less than one hour
  - 01: Less thann a day
  - 02: Between 1 to 6 days
  - 03: Between 7 to 27 days
  - 04: Between 28 to 29 days
  - 05: Between 1 to 5 months
  - 06: Between 6 to 11 months
  - 07: One year
  - 08: Between 2 to 4 years
  - 09: Between 5 to 9 years
  - 10: Between 10 to 14 years
  - 11: Between 15 to 19 years
  - 12: Between 20 to 24 years
  - 13: Between 25 to 29 years
  - 14: Between 30 to 34 years
  - 15: Between 35 to 39 years
  - 16: Between 40 to 44 years
  - 17: Between 45 to 49 years

- 18: Between 50 to 54 years
- 19: Between 55 to 59 years
- 20: Between 60 to 64 years
- 21: Between 65 to 69 years
- 22: Between 70 to 74 years
- 23: Between 75 to 79 years
- 24: Between 80 to 84 years
- 25: Between 85 to 89 years
- 26: Between 90 to 94 years
- 27: Between 95 to 99 years
- 28: More than 100 years
- 29: No information

- **NIVEL_EDU**: Last level of studies of the deceased.
  - 1: Preschool
  - 2: Elementary school
  - 3: Secondary school
  - 4: Academic or classical middle school
  - 5: Technical high school
  - 6: Normalista
  - 7: Technical professional
  - 8: Technological
  - 9: Bachelor's degree
  - 10: Specialization
  - 11: Master's degree
  - 12: PhD
  - 13: None
  - 99: No information
- **OCUPACION**: What was the last usual occupation of the the deceased.
- **IDPERTET**: According to the culture, people or physical physical features, the deceased was or was recognized as:
  - 1: Indigenous
  - 2: Rom (Gypsy)
  - 3: Raizal of the San Andres and Providencia Archipelago
  - 4: Palenquero of San Basilio
  - 5: Black, Afro-Colombian or Afro-descendant
  - 6: None of the above
  - 9: No information
- **SEG_SOCIAL**: Social security regime of the deceased.
  - 1: Contributory
  - 2: Subsidized
  - 3: Exception
  - 4: Special
  - 5: Uninsured
  - 9: No information
- **IDADMISALU**: Health Care Administration Entity to which the deceased belonged.
  - 1: Health Promoting Entity
  - 2: Health Promoting Entity - Subsidized
  - 3: Adapted health entity
  - 4: Special health entity
  - 5: Excepted health entity
  - 9: No information
- **ASIS_MED**: Received medical assistance medical assistance during the process that led to his death?
  - 1: Yes

- 2: No
- 3: Ignored
- 4: No information

Death Causes Descriptions

From 2008 to 2018

- **C_DIR1**: Direct cause.
- **C_ANT1**: Medical history 1.
- **C_ANT2**: Medical history 2.
- **C_ANT3**: Medical history 3.
- **C_PAT1**: Other important diseases 1.
- **C_PAT2**: Other important diseases 2.
- **CBAS1**: Code of the basic cause of death
- **CAU_HOMOL**: Basic cause grouped in Lista 105 - Colombia

From 2019 to 2020

- **CAUSA_MULT**: refers to the multiple cause codes ICD-10-WHO codes after processing by MultiMUSE and uni causal selection engine IRIS, for: the **antecedent causes** separated by the character "/" and the **pathological cause** separated by the character "*" This new field replaces the previous one-to-one code detail for direct, antecedent and pathological causes, used in bases from year 2018 backwards.
- **CBAS1**: Code of the basic cause of death
- **CAU_HOMOL**: Basic cause grouped in Lista 105 - Colombia

Keeping useful columns

[3]:

```
# Rename columns in 2014 dataset

data_2014.columns = data_2014.columns.str.upper()

# Keep important columns for 2008-2020 datasets

cols_to_keep = ['COD_DPTO',
                'COD_MUNIC',
                'SIT_DEFUN',
                'ANO',
                'MES',
                'SEXO',
                'EST_CIVIL',
                'GRU_ED1',
                'NIVEL_EDU',
                'OCUPACION',
                'IDPERTET',
                'SEG_SOCIAL',
                'ASIS_MED',
                'C_DIR1',
                'C_ANT1',
                'C_ANT2',
                'C_ANT3',
                'C_PAT1',
                'C_PAT2',
                'CAUSA_MULT',
                'C_BAS1',
                'CAU_HOMOL']
```

```
# Extract the data from useful columns for each year

df_2008 = data_2008[data_2008.columns[data_2008.columns.isin(cols_to_keep)]]
df_2009 = data_2009[data_2009.columns[data_2009.columns.isin(cols_to_keep)]]
df_2010 = data_2010[data_2010.columns[data_2010.columns.isin(cols_to_keep)]]
df_2011 = data_2011[data_2011.columns[data_2011.columns.isin(cols_to_keep)]]
df_2012 = data_2012[data_2012.columns[data_2012.columns.isin(cols_to_keep)]]
df_2013 = data_2013[data_2013.columns[data_2013.columns.isin(cols_to_keep)]]
df_2014 = data_2014[data_2014.columns[data_2014.columns.isin(cols_to_keep)]]
df_2015 = data_2015[data_2015.columns[data_2015.columns.isin(cols_to_keep)]]
df_2016 = data_2016[data_2016.columns[data_2016.columns.isin(cols_to_keep)]]
df_2017 = data_2017[data_2017.columns[data_2017.columns.isin(cols_to_keep)]]
df_2018 = data_2018[data_2018.columns[data_2018.columns.isin(cols_to_keep)]]
df_2019 = data_2019[data_2019.columns[data_2019.columns.isin(cols_to_keep)]]
df_2020 = data_2020[data_2020.columns[data_2020.columns.isin(cols_to_keep)]]
```

Get Datetime format for contained data

[4]:

```
import warnings
warnings.filterwarnings("ignore")


def dates_df(df):
    """
    dates_df adds the strings inside ANO and MES columns
    and it turns them to datetime
    """
    df["Date"] = df["ANO"] + "-" + df["MES"]
    df["Date"] = pd.to_datetime(df["Date"])
    df = df.drop(columns=["ANO", "MES"], inplace=True)

df_list = [df_2008, df_2009, df_2010, df_2011, df_2012, df_2013, df_2014,
           df_2015, df_2016, df_2017, df_2018, df_2019, df_2020]

for k in df_list:
    dates_df(k)
```

[5]:

```
warnings.filterwarnings("default")
```

[6]:

```
df_2019.head(3)
```

[6]:

	COD_DPTO	COD_MUNIC	SIT_DEFUN	SEXO
0	66	682	5	2
1	17	001	1	1

	EST_CIVIL	GRU_ED1	NIVEL_EDU
0	5	15	03
1	6	16	02

	OCUPACION	IDPERTET	SEG_SOCIAL	ASIS_MED
0	NaN	6	2	2
1	Agrcultor	6	2	1

	CAUSA_MULT	C_BAS1	CAU_HOMOL	Date
0	S269/S269/T141 X99	X994	101	2019-11-01
1	C712	C712	031	2019-12-01

Add latitude/longitude for each DPTO/MUNIC

The table **DANE - RELACIÓN DE MUNICIPIOS Y DEPARTAMENTOS ACTUALIZADOS** was retrieved from DANE DIVIPola and data was scraped using tabula:

[7]:

```
import tabula

df_dane_codes = tabula.read_pdf("C:/data/Tabla-Códigos-Dane.pdf", pages="all")
df_dane_codes[0]

...

df_dane_codes[23]
```

[8]:

```
# Prepare data and DANE codes

depto_munic = pd.read_csv('C:/data/depto_munic.csv', sep=";", dtype=object,
               encoding = "mbcs")

depto_munic = depto_munic.drop(columns=("Unnamed: 4"))
```

[8]:

COD_DPTO	DPTO	COD_MUNIC	MUNIC	full_location
5	ANTIOQUIA	1	Medellin	medellin, antioquia, colombia
5	ANTIOQUIA	2	Abejorral	abejorral, antioquia, colombia

Function to assign latitude/longitude for each location

The code used in this function comes from StackOverflow

[9]:

```
    from geopy.geocoders import Nominatim
    from geopy.exc import GeocoderTimedOut
    # You define col corresponding to adress, it can be one
    col_addr = ["full_location"]
    geocode = geopy.geocoders.Nominatim(user_agent="lukws").geocode

    def geopoints(row):
        search=""
        for x in col_addr:
            search = search + str(row[x]) +'  '

        if search is not None:
            print(row.name+1,end="\r")
            try:
                search_location = geocode(search, timeout=5)
                return search_location.latitude,search_location.longitude
            except (AttributeError, GeocoderTimedOut):
                print("Got an error on index : ",row.name)
                return 0,0
    print("Number adress to locate /",len(depto_munic),":")
    depto_munic['latitude'],depto_munic['longitude'] =
                                zip(*depto_munic.apply(geopoints, axis=1))
```

[9]:

```
Number adress to located / 1120 :
Got an error on index :  42
Got an error on index :  76
Got an error on index :  237
Got an error on index :  307
Got an error on index :  449
Got an error on index :  532
Got an error on index :  536
Got an error on index :  560
Got an error on index :  580
Got an error on index :  676
Got an error on index :  728
Got an error on index :  749
Got an error on index :  774
Got an error on index :  883
Got an error on index :  955
1120
```

Check latitude and longitude errors and save as .csv

The missing values where manually imputed.

[10]:

```
depto_munic.head(2)
```

[10]:

	COD_DPTO	DPTO	COD_MUNIC	MUNIC	full_location	latitude	longitude
0	5	ANTIOQUIA	1	Medellin	medellin, antioquia, colombia	6.244338	-75.573553
1	5	ANTIOQUIA	2	Abejorral	abejorral, antioquia, colombia	5.804950	-75.429842

[11]:

```
depto_munic = depto_munic.drop(columns=["full_location"])
```

[11]:

	COD_DPTO	DPTO	COD_MUNIC	MUNIC	latitude	longitude
0	5	ANTIOQUIA	1	Medellin	6.244338	-75.573553
1	5	ANTIOQUIA	2	Abejorral	5.804950	-75.429842

[12]:

```
# Export latitude and longitue as a file
depto_munic.to_csv(r'C:/data/complete_locations.csv',index = False, header=True)
```

Searching for each Latitude, Longitude

[13]:

```
from geopy.geocoders import Nominatim
geolocator = Nominatim(user_agent="your_name")
munic = "Abejorral"
dpto = "Antioquia"
country ="Colombia"
```

```
loc = geolocator.geocode(munic+","+dpto+","+ country)
print("latitude is :" ,loc.latitude,"\nlongtitude is:" ,loc.longitude)
```

[13]:

```
latitude is : 5.8049504
longtitude is: -75.42984181835139
```

Change format of COD_DPTO and COD_MUNIC for 2019-2020 datasets

[14]:

```
import warnings
warnings.filterwarnings("ignore")

df_2019['COD_DPTO'] = df_2019['COD_DPTO'].str.lstrip('0')
df_2019['COD_MUNIC'] = df_2019['COD_MUNIC'].str.lstrip('0')
df_2020['COD_DPTO'] = df_2020['COD_DPTO'].str.lstrip('0')
df_2020['COD_MUNIC'] = df_2020['COD_MUNIC'].str.lstrip('0')
```

[15]:

```
warnings.filterwarnings("default")
df_2019['COD_DPTO'].value_counts(),
df_2020['COD_MUNIC'].value_counts()
```

[15]:

```
1      191218
276      2611
754      2589
520      2574
834      2387
        …
446         2
883         1
884         1
888         1
362         1
Name: COD_MUNIC, Length: 580, dtype: int64
```

Format location data to add it to original dataframes

[16]:

```
df_locations = pd.read_csv('C:/data/complete_locations.csv', sep=";",
                dtype=object, encoding = "utf-8")

df_locations = df_locations.drop(columns=["Unnamed: 6"])
```

[17]:

```
def full_location_code(df):
    """
    Get the full code for "municipio" and "deparatamento"
    i. e.: Antioquia(5), Medellin(1)  = 51
    """
    df["LOC_CODE"] = df["COD_DPTO"] + df["COD_MUNIC"]
```

[18]:

```
import warnings
warnings.filterwarnings("ignore")

"""
df_list = [df_2008, df_2009, df_2010, df_2011, df_2012, df_2013, df_2014,
           df_2015, df_2016, df_2017, df_2018, df_2019, df_2020]
"""
```

```

for k in df_list:
    full_location_code(k)
```

[19]:

```
warnings.filterwarnings("default")
```

[20]:

```
df_2019.head(2)
```

[20]:

	COD_DPTO	COD_MUNIC	SIT_DEFUN	SEXO	EST_CIVIL	GRU_ED1	NIVEL_EDU	OCUPACION	IDPERTET	SEG_SOCIAL	ASIS_MED	CAUSA_MULT	C_BAS1	CAU_HOMOL	Date	LOC_CODE
0	66	682	5	2	5	15	03	NaN	6	2	2	S269/S269/T141 X99	X994	101	2019-11-01	66682
1	17	1	1	1	6	16	02	Agricultor	6	2	1	C712	C712	031	2019-12-01	171

Create a dict based on LOC_CODE and latitude/longitude

[21]:

```
full_location_code(df_locations)

df_locations["lat_long"] = df_locations["latitude"] + "," +
                            df_locations["longitude"]

df_locations.head(3)
```

[21]:

	COD_DPTO	DPTO	COD_MUNIC	MUNIC	latitude	longitude	LOC_CODE	lat_long
0	5	ANTIOQUIA	1	MEDELLIN	6.2443382	-75.573553	51	6.242443382,-75.57573553
1	5	ANTIOQUIA	2	ABEJORRAL	5.8049504	-75.429841	52	5.8049504,-75.429841

[22]:

```
def add_location(df):

    """

    Add latitude and longitude as new
    column based on LOC_CODE in each row
    mapping previously created dictionary

    """
    lat_long_dict = dict(zip(df_locations.LOC_CODE, df_locations.lat_long))
    df['LOCATION'] = df['LOC_CODE'].map(lat_long_dict)
    df[['LAT', 'LONG']] = df['LOCATION'].str.split(',', expand=True)
    df = df.drop(columns=["LOCATION", "COD_DPTO", "COD_MUNIC"], inplace=True)
```

[23]:

```
import warnings
warnings.filterwarnings("ignore")


"""
df_list = [df_2008, df_2009, df_2010, df_2011, df_2012, df_2013, df_2014,
           df_2015, df_2016, df_2017, df_2018, df_2019, df_2020]
"""


for k in df_list:
    full_location_code(k)
```

[24]:

```
for k in df_list:
    add_location(k)
```

[25]:

```
warnings.filterwarnings("default")
```

[26]:

```
df_2019.head(2)
```

[26]:

	SIT_DEFUN	SEXO	EST_CIVIL	GRU_ED1	NIVEL_EDU
0	5	2	5	15	03
1	1	1	6	16	02

	OCUPACION	IDPERTET	SEG_SOCIAL	ASIS_MED
0	NaN	6	2	2
1	Agricultor	6	2	1

	CAUSA_MULT	C_BAS1	CAU_HOMOL
0	S269/S269/T141 X99	X994	101
1	C712	C712	031

	Date	LOC_CODE	LAT	LONG
0	2019-11-01	66682	4.8650127	-75.6212436
1	2019-12-01	171	5.0668907	-75.5066661

Use the same format for Death Causes (2008-2018 and 2019-2020)

[27]:

```
df_2019.columns
```

[27]:

```
Index(['SIT_DEFUN', 'SEXO', 'EST_CIVIL', 'GRU_ED1', 'NIVEL_EDU', 'OCUPACION',
       'IDPERTET', 'SEG_SOCIAL', 'ASIS_MED', 'CAUSA_MULT', 'C_BAS1',
       'CAU_HOMOL', 'Date', 'LOC_CODE', 'LAT', 'LONG'],
      dtype='object')
```

[28]:

```
import warnings
warnings.filterwarnings("ignore")
```

[29]:

```
df_2019[["C_ANT", "C_PAT"]] = df_2019["CAUSA_MULT"].str.split("\*", expand=True)
df_2019["C_ANT"] = df_2019["C_ANT"].str.replace(' ', '/', regex=False)
df_2019["C_PAT"] = df_2019["C_PAT"].str.replace(' ', '/', regex=False)
df_2019["C_PAT1"] = df_2019["C_PAT"].str.split('/', expand = True)[0]
```

```
df_2019["C_PAT2"] = df_2019["C_PAT"].str.split('/', expand = True)[1]
df_2019["C_DIR1"] = df_2019["C_ANT"].str.split('/', expand = True)[0]
df_2019["C_ANT1"] = df_2019["C_ANT"].str.split('/', expand = True)[1]
df_2019["C_ANT2"] = df_2019["C_ANT"].str.split('/', expand = True)[2]
df_2019["C_ANT3"] = df_2019["C_ANT"].str.split('/', expand = True)[3]

df_2019 = df_2019[['SIT_DEFUN', 'SEXO', 'EST_CIVIL',
                   'GRU_ED1', 'NIVEL_EDU', 'OCUPACION', 'IDPERTET', 'SEG_SOCIAL',
                   'ASIS_MED', 'C_DIR1', 'C_ANT1', 'C_ANT2', 'C_ANT3',
                   'C_PAT1', 'C_PAT2', 'C_BAS1', 'CAU_HOMOL', 'Date',
                   'LOC_CODE', 'LAT', 'LONG']]

df_2020[["C_ANT", "C_PAT"]] = df_2020["CAUSA_MULT"].str.split("\*", expand=True)
df_2020["C_ANT"] = df_2020["C_ANT"].str.replace(' ', '/', regex=False)
df_2020["C_PAT"] = df_2020["C_PAT"].str.replace(' ', '/', regex=False)
df_2020["C_PAT1"] = df_2020["C_PAT"].str.split('/', expand = True)[0]
df_2020["C_PAT2"] = df_2020["C_PAT"].str.split('/', expand = True)[1]
df_2020["C_DIR1"] = df_2020["C_ANT"].str.split('/', expand = True)[0]
df_2020["C_ANT1"] = df_2020["C_ANT"].str.split('/', expand = True)[1]
df_2020["C_ANT2"] = df_2020["C_ANT"].str.split('/', expand = True)[2]
df_2020["C_ANT3"] = df_2020["C_ANT"].str.split('/', expand = True)[3]

df_2020 = df_2020[['SIT_DEFUN', 'SEXO', 'EST_CIVIL',
                   'GRU_ED1', 'NIVEL_EDU', 'OCUPACION', 'IDPERTET', 'SEG_SOCIAL',
                    'ASIS_MED', 'C_DIR1', 'C_ANT1', 'C_ANT2', 'C_ANT3',
                   'C_PAT1', 'C_PAT2', 'C_BAS1', 'CAU_HOMOL', 'Date',
                   'LOC_CODE', 'LAT', 'LONG']]
```

[30]:

```
warnings.filterwarnings("default")
```

Append all dataframes in a final one

[31]:

```
appended_data = pd.concat(df_list)
# write DataFrame to a .csv file
appended_data.to_csv(r'C:/data/complete.csv', index = False, header=True)
```

Check the new dataset

[32]:

```
all_data = pd.read_csv('C:/data/complete.csv', sep=",",
                       dtype=object, encoding = "utf-8")

# Check the dataset

all_data.info()
```

[32]:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2855415 entries, 0 to 2855414
Data columns (total 21 columns):
 #   Column      Dtype
---  ------      -----
```

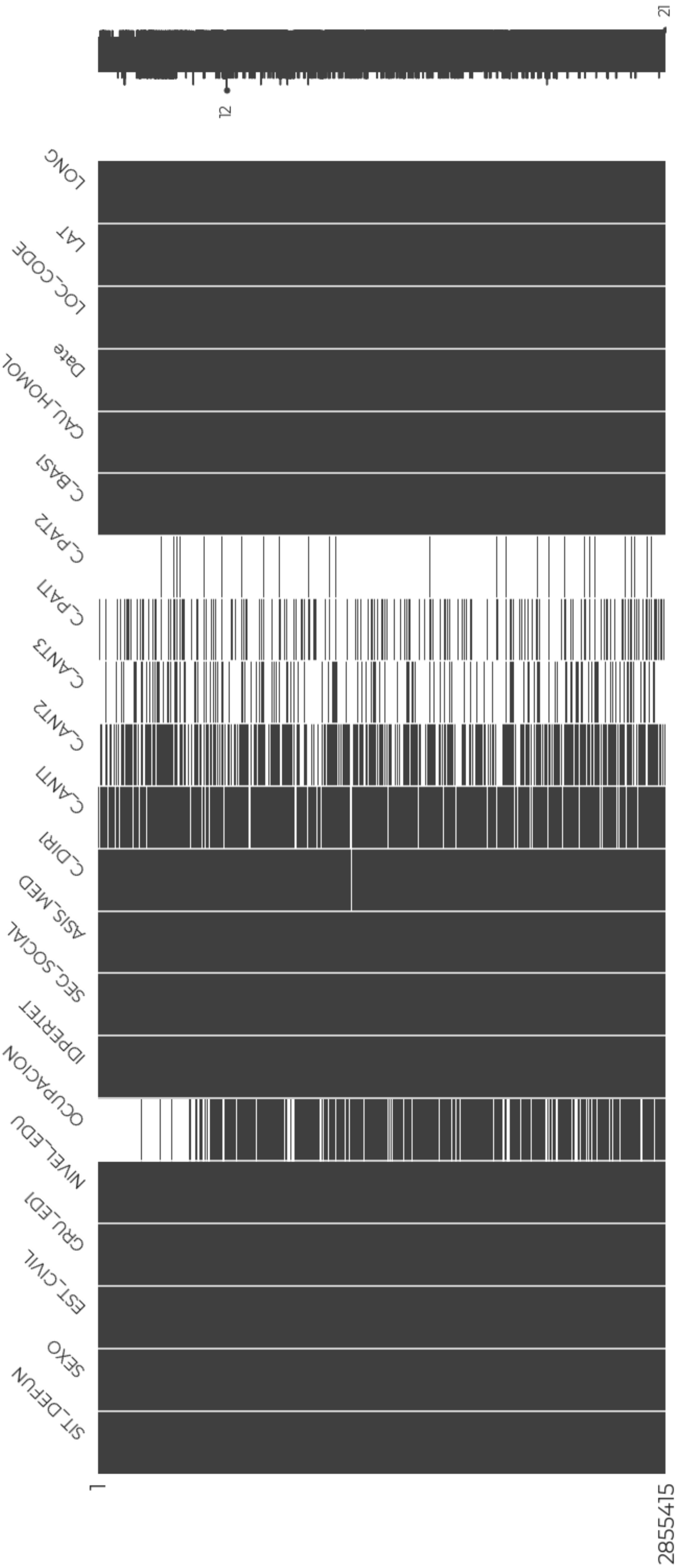
```
 0   SIT_DEFUN   object
 1   SEXO        object
 2   EST_CIVIL   object
 3   GRU_ED1     object
 4   NIVEL_EDU   object
 5   OCUPACION   object
 6   IDPERTET    object
 7   SEG_SOCIAL  object
 8   ASIS_MED    object
 9   C_DIR1      object
 10  C_ANT1      object
 11  C_ANT2      object
 12  C_ANT3      object
 13  C_PAT1      object
 14  C_PAT2      object
 15  C_BAS1      object
 16  CAU_HOMOL   object
 17  Date        object
 18  LOC_CODE    object
 19  LAT         object
 20  LONG        object
dtypes: object(21)
memory usage: 457.5+ MB
```

[33]:
```
all_data.isna().sum()
```

[33]:
```
SIT_DEFUN           0
SEXO                0
EST_CIVIL           1
GRU_ED1             0
NIVEL_EDU           6
OCUPACION      785896 #Corresponds to 27.52% of the whole dataset
IDPERTET            6
SEG_SOCIAL          0
ASIS_MED            0
C_DIR1           6509 #Corresponds to 0.22% of the whole dataset
C_ANT1         228612
C_ANT2         963325
C_ANT3        2101419
C_PAT1        1937792
C_PAT2        2648078
C_BAS1              0
CAU_HOMOL           0
Date                0
LOC_CODE            0
LAT               311 #Corresponds to 0.011% of the whole dataset
LONG              311 #Corresponds to 0.011% of the whole dataset
dtype: int64
```

[34]:
```
msno.matrix(all_data);
```

[34]:


SIT_DEFUN
SEXO
EST_CIVIL
GRU_ED1
NIVEL_EDU
OCUPACION
IDPERTET
SEG_SOCIAL
ASIS_MED
C_DIR1
C_ANT1
C_ANT2
C_ANT3
C_PAT1
C_PAT2
C_BAS1
CAU_HOMOL
Date
LOC_CODE
LAT
LONG
1
2855415
12
21