Momo

May 25, 2021

1 Informes de mortalidad

Actualizado diariamente, este documento se visualiza mejor aquí.

Datos del Sistema de Monitorización de la Mortalidad diaria, que incluye las defunciones por todas las causas procedentes de 3.929 registros civiles informatizados, que representan el 92% de la población española.

[16]:
```
# Cargamos datos
import Loading_data
from matplotlib import pyplot as plt
import warnings
warnings.filterwarnings('ignore')
import pandas as pd
from IPython.display import display, HTML
```

[17]:
```
df = pd.read_csv('https://momo.isciii.es/public/momo/data')
df.to_csv('/tmp/momo.csv')
df.head()
```

[17]:
```
     ambito cod_ambito  cod_ine_ambito nombre_ambito cod_sexo nombre_sexo  \
0  nacional        NaN             NaN           NaN      all       todos
1  nacional        NaN             NaN           NaN      all       todos
2  nacional        NaN             NaN           NaN      all       todos
3  nacional        NaN             NaN           NaN      all       todos
4  nacional        NaN             NaN           NaN      all       todos

  cod_gedad nombre_gedad fecha_defuncion  defunciones_observadas  \
0       all        todos      2019-03-28                   1142
1       all        todos      2019-03-29                   1110
2       all        todos      2019-03-30                   1133
3       all        todos      2019-03-31                   1046
4       all        todos      2019-04-01                   1113

   defunciones_observadas_lim_inf  defunciones_observadas_lim_sup  \
0                          1142.0                          1142.0
1                          1110.0                          1110.0
2                          1133.0                          1133.0
```

```
3                          1046.0                          1046.0
4                          1113.0                          1113.0

   defunciones_esperadas  defunciones_esperadas_q01  defunciones_esperadas_q99
0                 1124.5                    1006.18                    1245.06
1                 1124.5                    1006.18                    1245.06
2                 1119.5                    1001.84                    1245.06
3                 1119.5                    1010.39                    1245.06
4                 1114.5                    1010.39                    1238.86
```

[18]:
```
import janitor
import datetime



def pipeline_basic_with_query(df,query):
    ''' Basic filtering, using janitor
        Carga de datos, enriquecimiento de fechas y filtro por query␣
 ↪configurable
    '''
    LISTA_COLUMNAS_A_BORRAR = ['Unnamed: 0',
                           'defunciones_observadas_lim_inf',
                           'defunciones_observadas_lim_sup',
                           'defunciones_esperadas',
                           'defunciones_esperadas_q01',
                           'defunciones_esperadas_q99']
    return (
        df
        # Quitar: columnas
        .remove_columns(LISTA_COLUMNAS_A_BORRAR)
        .clean_names()
        # Enriquecer: fechas con columnas de años, mes y año-mes
        .rename_column( "fecha_defuncion", "date")
        .to_datetime('date')
        .join_apply(lambda x: x['date'].strftime('%Y')    ,␣
 ↪new_column_name="date_year"       )
        .join_apply(lambda x: x['date'].strftime('%m')    ,␣
 ↪new_column_name="date_month"      )
        .join_apply(lambda x: x['date'].strftime('%m-%d') ,␣
 ↪new_column_name="date_month_day"  )
        .join_apply(lambda x: x['date'].strftime('%U')    ,␣
 ↪new_column_name="date_week"       )
        .join_apply(lambda x: x['date'].strftime('%Y-%m') ,␣
 ↪new_column_name="date_year_month" )
        .join_apply(lambda x: x['date'].strftime('%Y-%U') ,␣
 ↪new_column_name="date_year_week"  )
```

```
        # Filtrar:por query
        .filter_on( query )
        .set_index('date')
    )

def pipeline_basic(df):
    query = 'ambito       == "nacional" & nombre_gedad == "todos"  &␣
↪nombre_sexo  == "todos"  '
    return pipeline_basic_with_query(df,query)




def extraer_defunciones_anuales_por_periodo(periodo_de_tiempo,query):
    '''Extrae el cuadro de comparativa por week, or year '''

    def pipeline_agregado_anual(periodo_de_tiempo,df,year):
        ''' Saca un dataframe de los datos agrupados por año'''
        return (
            df
            .filter_on('date_year == "'+year+'"'  )
            .groupby_agg( by='date_'+periodo_de_tiempo, agg='sum',␣
↪agg_column_name="defunciones_observadas", new_column_name="agregados")
            .rename_column( "agregados", year)
            .join_apply(lambda x: x['date_'+periodo_de_tiempo]  ,␣
↪new_column_name=periodo_de_tiempo )
            .set_index('date_'+periodo_de_tiempo)
            [[periodo_de_tiempo,year]]
            .drop_duplicates()
        )

    def pipeline_comparativa_anual(periodo_de_tiempo,df_2018,df_2019,df_2020):
        ''' Mergea tres dataframes de año, por periodo de tiempo'''
        return  (
        df_2018
        .merge( df_2019, on=periodo_de_tiempo, how='right')
        .merge( df_2020, on=periodo_de_tiempo, how='left')
        .sort_naturally(periodo_de_tiempo)
        .set_index(periodo_de_tiempo)
        .join_apply(lambda x: x['2020'] - x['2019'] , new_column_name="resta␣
↪2020 y 2019"  )
        )

    # Sacamos los datos y limpiamos
    df       = pd.read_csv('')
    df_basic = pipeline_basic_with_query(df,query)
```

```
    # Sacamos los datos agrupados por años
    muertes_2018  =␣
 ↪pipeline_agregado_anual(periodo_de_tiempo,df=df_basic,year='2018')
    muertes_2019  =␣
 ↪pipeline_agregado_anual(periodo_de_tiempo,df=df_basic,year='2019')
    muertes_2020  =␣
 ↪pipeline_agregado_anual(periodo_de_tiempo,df=df_basic,year='2020')

    # Generamos un solo cuadro, con columna por año
    df_comparativa_años  =␣
 ↪pipeline_comparativa_anual(periodo_de_tiempo,muertes_2018,muertes_2019,muertes_2020)
    return df_comparativa_años

def debug_extraer_defunciones_anuales_por_periodo():
    """ Solo para depurar"""
    query = 'ambito       == "nacional" & nombre_gedad == "todos"  &␣
 ↪nombre_sexo  == "todos"  '
    df_muertes_anuales_por_semana =␣
 ↪extraer_defunciones_anuales_por_periodo("week",query)
    df_muertes_anuales_por_mes    =␣
 ↪extraer_defunciones_anuales_por_periodo("month",query)
    return  df_muertes_anuales_por_semana , df_muertes_anuales_por_mes


#df1, df2 = debug_extraer_defunciones_anuales_por_periodo()
#df1
```

1.1 Sacamos el grafico comparativo de fallecimiento, para los años 2019 y 2020, por semana

[10]:
```
from matplotlib import pyplot as plt
from IPython.display import display, HTML
import pandas as pd

import numpy as np


periodo_de_tiempo="week"
query = 'ambito       == "nacional" & nombre_gedad == "todos"  & nombre_sexo ␣
 ↪== "todos"  '

df = extraer_defunciones_anuales_por_periodo(periodo_de_tiempo,query)

fig = plt.figure(figsize=(8, 6), dpi=80)
plt.xticks(rotation=90)
```

```
for ca in ['2018','2019','2020']:
    plt.plot(df[ca])
    plt.legend(df.columns)
    plt.xlabel(periodo_de_tiempo)
    plt.ylabel("Deaths by " + periodo_de_tiempo)
    fig.suptitle('Comparativa de fallecimientos por año, según MOMO',␣
 ↪fontsize=20)
plt.show()



periodo_de_tiempo="week"
query = 'ambito       == "nacional" & nombre_gedad == "todos"  & nombre_sexo ␣
 ↪== "todos"  '
df = extraer_defunciones_anuales_por_periodo(periodo_de_tiempo,query)

df.style.format({"2020": "{:20,.0f}",
                          "2018": "{:20,.0f}",
                          "2019":  "{:20,.0f}",
                          "resta 2020 y 2019": "{:20,.0f}", }).
 ↪background_gradient(cmap='Wistia',subset=['resta 2020 y 2019'])
```

```
      ␣
↪---------------------------------------------------------------------------

       FileNotFoundError                         Traceback (most recent call␣
↪last)

       <ipython-input-10-4e22be8b3266> in <module>
         9 query = 'ambito       == "nacional" & nombre_gedad == "todos"  &␣
↪nombre_sexo  == "todos"  '
        10
   ---> 11 df = extraer_defunciones_anuales_por_periodo(periodo_de_tiempo,query)
        12
        13 fig = plt.figure(figsize=(8, 6), dpi=80)


       <ipython-input-8-76b41831e557> in␣
↪extraer_defunciones_anuales_por_periodo(periodo_de_tiempo, query)
        68
        69     # Sacamos los datos y limpiamos
   ---> 70     df       = pd.read_csv('')
        71     df_basic = pipeline_basic_with_query(df,query)
        72
```

```
      ~/anaconda2/envs/jupyter/lib/python3.6/site-packages/pandas/io/parsers.
↪py in parser_f(filepath_or_buffer, sep, delimiter, header, names, index_col,␣
↪usecols, squeeze, prefix, mangle_dupe_cols, dtype, engine, converters,␣
↪true_values, false_values, skipinitialspace, skiprows, skipfooter, nrows,␣
↪na_values, keep_default_na, na_filter, verbose, skip_blank_lines, parse_dates,␣
↪infer_datetime_format, keep_date_col, date_parser, dayfirst, cache_dates,␣
↪iterator, chunksize, compression, thousands, decimal, lineterminator,␣
↪quotechar, quoting, doublequote, escapechar, comment, encoding, dialect,␣
↪error_bad_lines, warn_bad_lines, delim_whitespace, low_memory, memory_map,␣
↪float_precision)
      683         )
      684
  --> 685         return _read(filepath_or_buffer, kwds)
      686
      687     parser_f.__name__ = name


      ~/anaconda2/envs/jupyter/lib/python3.6/site-packages/pandas/io/parsers.
↪py in _read(filepath_or_buffer, kwds)
      455
      456     # Create the parser.
  --> 457     parser = TextFileReader(fp_or_buf, **kwds)
      458
      459     if chunksize or iterator:


      ~/anaconda2/envs/jupyter/lib/python3.6/site-packages/pandas/io/parsers.
↪py in __init__(self, f, engine, **kwds)
      893             self.options["has_index_names"] = kwds["has_index_names"]
      894
  --> 895         self._make_engine(self.engine)
      896
      897     def close(self):


      ~/anaconda2/envs/jupyter/lib/python3.6/site-packages/pandas/io/parsers.
↪py in _make_engine(self, engine)
     1133     def _make_engine(self, engine="c"):
     1134         if engine == "c":
  -> 1135             self._engine = CParserWrapper(self.f, **self.options)
     1136         else:
     1137             if engine == "python":


      ~/anaconda2/envs/jupyter/lib/python3.6/site-packages/pandas/io/parsers.
↪py in __init__(self, src, **kwds)
     1915         kwds["usecols"] = self.usecols
```

```
   1916
-> 1917         self._reader = parsers.TextReader(src, **kwds)
   1918         self.unnamed_cols = self._reader.unnamed_cols
   1919

pandas/_libs/parsers.pyx in pandas._libs.parsers.TextReader.__cinit__()

pandas/_libs/parsers.pyx in pandas._libs.parsers.TextReader.
↪_setup_parser_source()

FileNotFoundError: [Errno 2] File b'' does not exist: b''
```

[5]:
```
def get_current_year_comparison(query):
    """Saca muertos del año en curso en el ambito como argumento"""
    df          = pd.read_csv('/tmp/momo.csv')
    df = pipeline_basic_with_query(df,query)

    semana_actual = df.tail(1).date_week.values[0]
    year_actual = df.tail(1).date_year.values[0]
    date_month_day_actual= df.tail(1).date_month_day.values[0]
    year_last = str(int(year_actual)-1)

    death_this_year_today = df.query( f"date_year == '{year_actual}' ").
 ↪defunciones_observadas.sum()
    deaht_last_year_today = df.query( f"date_year == '{year_last}' and␣
 ↪date_month_day <= '{date_month_day_actual}' ").defunciones_observadas.sum()
    deaths_this_year_excess = death_this_year_today - deaht_last_year_today
    return deaths_this_year_excess

query = f""" ambito == "nacional" & nombre_gedad == "todos" & nombre_sexo  ==␣
 ↪"todos"  """
deaths_this_year_excess = get_current_year_comparison(query)

display(HTML(f"<h4 id='excedente'>Excdente de muertes de este año, respecto al␣
 ↪año anterior:</h4><h2>{deaths_this_year_excess:,.0f} </h2>"))
```

```
<IPython.core.display.HTML object>
```

[6]:
```
query = f""" nombre_ambito == "Madrid, Comunidad de" & nombre_gedad == "todos"␣
 ↪& nombre_sexo  == "todos"  """
deaths_this_year_excess = get_current_year_comparison(query)
```

```
display(HTML(f"<h4 id='excedentemadrid'>Excedente de muertes de este año en␣
 ↪Madrid, respecto al año anterior:</h4><h2>{deaths_this_year_excess:,.0f} </
 ↪h2>"))
```

```
<IPython.core.display.HTML object>
```

[7]:
```
# Sacamos las muertes en madrid de hombres y de mujeres

import numpy as np
import seaborn as sns


def pipeline_comparativa_semestral_diaria(df):
    return  (
        df
        .filter_on(" defunciones_observadas > 0")
        .
 ↪remove_columns(['nombre_gedad','ambito','cod_ambito','cod_ine_ambito','nombre_ambito','cod_
        .rename_column( "nombre_sexo"    , "sexo")
        .rename_column( "date_year_month",  "mes")
        )

# Sacamos los datos de 2019
df          = pd.read_csv('/tmp/momo.csv')
query       = '  date_year == "2019" & nombre_ambito       == "Madrid,␣
 ↪Comunidad de" & nombre_gedad == "todos"  & nombre_sexo  != "todos" & ␣
 ↪date_month < "13" '
df_madrid_2019 =  pipeline_basic_with_query(df,query)
df_madrid_2019 =  pipeline_comparativa_semestral_diaria(df_madrid_2019)

# Sacamos los datos de 2020
df       = pd.read_csv('/tmp/momo.csv')
query = ' date_year == "2020"  &  nombre_ambito       == "Madrid, Comunidad de"␣
 ↪& nombre_gedad == "todos"  & nombre_sexo  != "todos" &  date_month < "13" '
df_madrid_2020 =  pipeline_basic_with_query(df,query)
df_madrid_2020 =  pipeline_comparativa_semestral_diaria(df_madrid_2020)




df_madrid_2019
```

[7]:

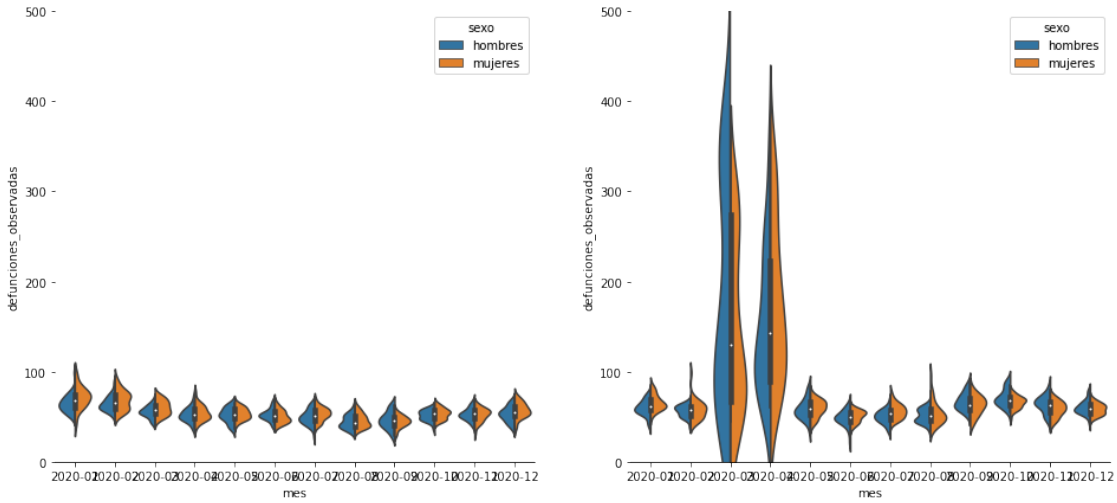
date	sexo	defunciones_observadas	date_month_day	mes
2019-01-01	hombres	60	01-01	2019-01
2019-01-02	hombres	55	01-02	2019-01

```
2019-01-03  hombres                    63          01-03  2019-01
2019-01-04  hombres                    62          01-04  2019-01
2019-01-05  hombres                    55          01-05  2019-01
…              …                    …             …       …
2019-12-27  mujeres                    61          12-27  2019-12
2019-12-28  mujeres                    58          12-28  2019-12
2019-12-29  mujeres                    57          12-29  2019-12
2019-12-30  mujeres                    54          12-30  2019-12
2019-12-31  mujeres                    68          12-31  2019-12

[730 rows x 4 columns]
```

[8]:
```
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

display(HTML("<h2>Distribucion muertes en Madrid  </h2>"))
display(HTML("<h3>Comparativa de defunciones, entre el primer semestre de 2019␣
 ↪y el del 2020</h3>"))

f, axes = plt.subplots(1 , 2 ,figsize=(16, 7), sharex=True)
sns.despine(left=True)

# Mismo limites, para poder comparar entre años
axes[0].set_ylim([0,500])
axes[1].set_ylim([0,500])


sns.violinplot(x="mes", y="defunciones_observadas", hue="sexo",
                    data=df_madrid_2019,  split=True, scale="count", ax=axes[0]␣
 ↪)


sns.violinplot(x="mes", y="defunciones_observadas", hue="sexo",
                    data=df_madrid_2020,  split=True, scale="count", ax=axes[1])
```

```
<IPython.core.display.HTML object>


<IPython.core.display.HTML object>
```

[8]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f1695d27390>
```



[9]:
```
# Aux functions
def print_categorical_variables(df):
    """ Get a dict with categorical variables"""
    my_dict = {}
    cols = df.columns
    num_cols = df._get_numeric_data().columns
    # Show categorical values
    categorical = list(set(cols) - set(num_cols))
    for i in categorical :
        if 'echa' not in i.lower() : my_dict[i] = df[i].unique()
    return my_dict
df       = pd.read_csv('/tmp/momo.csv')
my_dict = print_categorical_variables(df)
my_dict
```

[9]:
```
{'cod_ambito': array([nan, 'AN', 'AR', 'AS', 'IB', 'CN', 'CB', 'CL', 'CM', 'CT',
'VC',
        'EX', 'GA', 'MD', 'MC', 'NC', 'PV', 'RI', 'CE', 'ML'], dtype=object),
 'ambito': array(['nacional', 'ccaa'], dtype=object),
 'nombre_ambito': array([nan, 'Andalucía', 'Aragón', 'Asturias, Principado de',
        'Balears, Illes', 'Canarias', 'Cantabria', 'Castilla y León',
        'Castilla - La Mancha', 'Cataluña', 'Comunitat Valenciana',
        'Extremadura', 'Galicia', 'Madrid, Comunidad de',
        'Murcia, Región de', 'Navarra, Comunidad Foral de', 'País Vasco',
        'Rioja, La', 'Ceuta', 'Melilla'], dtype=object),
 'cod_sexo': array(['all', '1', '6'], dtype=object),
 'cod_gedad': array(['all', 'menos_65', '65_74', 'mas_74'], dtype=object),
 'nombre_gedad': array(['todos', 'edad < 65', 'edad 65-74', 'edad > 75'],
dtype=object),
 'nombre_sexo': array(['todos', 'hombres', 'mujeres'], dtype=object)}
```

[35]:
```
momo2020 = pd.read_csv("/root/scripts/COVID-19/data/momo2019.csv", sep='\t',)
```

[25]:
```
df = pd.read_csv('https://momo.isciii.es/public/momo/data')
df.to_csv('/tmp/momo.csv')
df.head()
```

[25]:

	ambito	cod_ambito	cod_ine_ambito	nombre_ambito	cod_sexo	nombre_sexo	cod_gedad	nombre_gedad	fecha_defuncion	defunciones_observadas	defunciones_observadas_lim_inf	defunciones_observadas_lim_sup	defunciones_esperadas	defunciones_esperadas_q01	defunciones_esperadas_q99
0	nacional	NaN	NaN	NaN	all	todos	all	todos	2019-03-28	1142	1142.0	1142.0	1124.5	1006.18	1245.06
1	nacional	NaN	NaN	NaN	all	todos	all	todos	2019-03-29	1110	1110.0	1110.0	1124.5	1006.18	1245.06
2	nacional	NaN	NaN	NaN	all	todos	all	todos	2019-03-30	1133	1133.0	1133.0	1119.5	1001.84	1245.06
3	nacional	NaN	NaN	NaN	all	todos	all	todos	2019-03-31	1046	1046.0	1046.0	1119.5	1010.39	1245.06
4	nacional	NaN	NaN	NaN	all	todos	all	todos	2019-04-01	1113	1113.0	1113.0	1114.5	1010.39	1238.86

[27]:
```
# Sacamos los datos y limpiamos
df       = pd.read_csv('/tmp/momo.csv')
query = 'ambito       == "nacional" & nombre_gedad == "todos"  & nombre_sexo ␣
 ↪== "todos"  '

df_basic = pipeline_basic_with_query(df,query)
```

[32]:
```
df_basic.head(3)
```

[32]:

date	ambito	cod_ambito	cod_ine_ambito	nombre_ambito	cod_sexo
2019-03-28	nacional	NaN	NaN	NaN	all
2019-03-29	nacional	NaN	NaN	NaN	all

```
2019-03-30  nacional        NaN             NaN           NaN      all

           nombre_sexo cod_gedad nombre_gedad  defunciones_observadas  \
date
2019-03-28       todos       all        todos                    1142
2019-03-29       todos       all        todos                    1110
2019-03-30       todos       all        todos                    1133

           date_year date_month date_month_day date_week date_year_month  \
date
2019-03-28      2019         03          03-28        12         2019-03
2019-03-29      2019         03          03-29        12         2019-03
2019-03-30      2019         03          03-30        12         2019-03

           date_year_week
date
2019-03-28        2019-12
2019-03-29        2019-12
2019-03-30        2019-12
```

[30]:
```
df
```

[30]:
```
        Unnamed: 0    ambito cod_ambito  cod_ine_ambito nombre_ambito  \
0                0  nacional        NaN             NaN           NaN
1                1  nacional        NaN             NaN           NaN
2                2  nacional        NaN             NaN           NaN
3                3  nacional        NaN             NaN           NaN
4                4  nacional        NaN             NaN           NaN
…              …       …        …              …           …
179755      179755      ccaa         ML            19.0       Melilla
179756      179756      ccaa         ML            19.0       Melilla
179757      179757      ccaa         ML            19.0       Melilla
179758      179758      ccaa         ML            19.0       Melilla
179759      179759      ccaa         ML            19.0       Melilla

       cod_sexo nombre_sexo cod_gedad nombre_gedad fecha_defuncion  \
0           all       todos       all        todos      2019-03-28
1           all       todos       all        todos      2019-03-29
2           all       todos       all        todos      2019-03-30
3           all       todos       all        todos      2019-03-31
4           all       todos       all        todos      2019-04-01
…         …          …       …           …                …
179755        6     mujeres    mas_74    edad > 75      2021-04-10
179756        6     mujeres    mas_74    edad > 75      2021-04-11
179757        6     mujeres    mas_74    edad > 75      2021-04-12
179758        6     mujeres    mas_74    edad > 75      2021-04-13
179759        6     mujeres    mas_74    edad > 75      2021-04-14
```

```
        defunciones_observadas  defunciones_observadas_lim_inf  \
0                         1142                     1142.000000
1                         1110                     1110.000000
2                         1133                     1133.000000
3                         1046                     1046.000000
4                         1113                     1113.000000
…                          …                             …
179755                       0                        0.000000
179756                       0                        0.000000
179757                       0                        0.000000
179758                       0                        0.000000
179759                       1                        0.204872

        defunciones_observadas_lim_sup  defunciones_esperadas  \
0                         1142.000000                 1124.5
1                         1110.000000                 1124.5
2                         1133.000000                 1119.5
3                         1046.000000                 1119.5
4                         1113.000000                 1114.5
…                                …                      …
179755                       0.220211                    0.0
179756                       0.275007                    0.0
179757                       0.377218                    0.0
179758                       1.544990                    0.0
179759                       1.976183                    0.0

        defunciones_esperadas_q01  defunciones_esperadas_q99
0                         1006.18                    1245.06
1                         1006.18                    1245.06
2                         1001.84                    1245.06
3                         1010.39                    1245.06
4                         1010.39                    1238.86
…                           …                          …
179755                       0.00                       2.00
179756                       0.00                       2.00
179757                       0.00                       2.00
179758                       0.00                       2.00
179759                       0.00                       2.00

[179760 rows x 16 columns]
```

[33]:
```
periodo_de_tiempo="week"
year="2021"
def pipeline_agregado_anual(periodo_de_tiempo,df,year):
    ''' Saca un dataframe de los datos agrupados por año'''
    return (
```

```
        df
        .filter_on('date_year == "'+year+'"'  )
        .groupby_agg( by='date_'+periodo_de_tiempo, agg='sum',␣
↪agg_column_name="defunciones_observadas", new_column_name="agregados")
        .rename_column( "agregados", year)
        .join_apply(lambda x: x['date_'+periodo_de_tiempo]  ,␣
↪new_column_name=periodo_de_tiempo )
        .set_index('date_'+periodo_de_tiempo)
        [[periodo_de_tiempo,year]]
        .drop_duplicates()
    )
df_2021 = pipeline_agregado_anual(periodo_de_tiempo,df_basic,year)
df_2021
```

[33]:

date_week	week	2021
00	00	2586
01	01	9799
02	02	11231
03	03	12263
04	04	12253
05	05	11211
06	06	10070
07	07	9142
08	08	8612
09	09	8177
10	10	7961
11	11	7748
12	12	7924
13	13	7784
14	14	5626
15	15	1782

[37]:
```
momo2020['2021'] = df_2021['2021']
momo2020.join_apply(lambda x: x['2021'] - x['2020'] , new_column_name="resta␣
↪2021 y 2020"  )
```

[37]:

	week	2018	2019	2020	resta 2020 y 2019	2021	resta 2021 y 2020
0	0	NaN	6282	4849	-1433	NaN	NaN
1	1	NaN	9368	8874	-494	NaN	NaN
2	2	NaN	9960	9396	-564	NaN	NaN
3	3	NaN	9548	9395	-153	NaN	NaN
4	4	NaN	9654	9497	-157	NaN	NaN
5	5	NaN	9209	8891	-318	NaN	NaN
6	6	NaN	8888	8459	-429	NaN	NaN
7	7	NaN	8725	7930	-795	NaN	NaN
8	8	NaN	8422	7990	-432	NaN	NaN

9	9	NaN	8351	8021	-330	NaN	NaN	
10	10	NaN	7872	8760	888	NaN	NaN	
11	11	NaN	7576	11459	3883	NaN	NaN	
12	12	NaN	7756	17834	10078	NaN	NaN	
13	13	NaN	7795	19293	11498	NaN	NaN	
14	14	NaN	7583	16429	8846	NaN	NaN	
15	15	NaN	7472	12791	5319	NaN	NaN	
16	16	NaN	7286	10123	2837	NaN	NaN	
17	17	NaN	7460	8591	1131	NaN	NaN	
18	18	NaN	7342	8213	871	NaN	NaN	
19	19	NaN	7195	7114	-81	NaN	NaN	
20	20	NaN	7073	7210	137	NaN	NaN	
21	21	NaN	7255	7294	39	NaN	NaN	
22	22	NaN	7219	6851	-368	NaN	NaN	
23	23	NaN	6728	6522	-206	NaN	NaN	
24	24	NaN	6829	6601	-228	NaN	NaN	
25	25	NaN	7112	7131	19	NaN	NaN	
26	26	NaN	7557	7152	-405	NaN	NaN	
27	27	NaN	7194	7367	173	NaN	NaN	
28	28	NaN	6858	7184	326	NaN	NaN	
29	29	NaN	7000	7343	343	NaN	NaN	
30	30	NaN	6757	7892	1135	NaN	NaN	
31	31	NaN	6921	8057	1136	NaN	NaN	
32	32	NaN	6695	7853	1158	NaN	NaN	
33	33	NaN	6553	7427	874	NaN	NaN	
34	34	NaN	6606	7593	987	NaN	NaN	
35	35	NaN	6517	7287	770	NaN	NaN	
36	36	NaN	6623	7580	957	NaN	NaN	
37	37	NaN	6604	7831	1227	NaN	NaN	
38	38	NaN	6660	7633	973	NaN	NaN	
39	39	NaN	6707	7803	1096	NaN	NaN	
40	40	NaN	6941	8128	1187	NaN	NaN	
41	41	NaN	6999	8087	1088	NaN	NaN	
42	42	NaN	7092	8949	1857	NaN	NaN	
43	43	NaN	7278	9149	1871	NaN	NaN	
44	44	NaN	7311	9742	2431	NaN	NaN	
45	45	NaN	7628	9584	1956	NaN	NaN	
46	46	NaN	7887	9398	1511	NaN	NaN	
47	47	NaN	7836	9093	1257	NaN	NaN	
48	48	NaN	7603	9046	1443	NaN	NaN	
49	49	NaN	7895	9067	1172	NaN	NaN	
50	50	NaN	7895	8908	1013	NaN	NaN	
51	51	NaN	7577	8705	1128	NaN	NaN	
52	52	1111.0	3185	6159	2974	NaN	NaN	

[38]:
```
df_2021['2021']
```

[38]:
```
date_week
00     2586
01     9799
02    11231
03    12263
04    12253
05    11211
06    10070
07     9142
08     8612
09     8177
10     7961
11     7748
12     7924
13     7784
14     5626
15     1782
Name: 2021, dtype: int64
```

[39]:
```
momo2020['2021'] = df_2021['2021']
momo2020
```

[39]:

	week	2018	2019	2020	resta 2020 y 2019	2021
0	0	NaN	6282	4849	-1433	NaN
1	1	NaN	9368	8874	-494	NaN
2	2	NaN	9960	9396	-564	NaN
3	3	NaN	9548	9395	-153	NaN
4	4	NaN	9654	9497	-157	NaN
5	5	NaN	9209	8891	-318	NaN
6	6	NaN	8888	8459	-429	NaN
7	7	NaN	8725	7930	-795	NaN
8	8	NaN	8422	7990	-432	NaN
9	9	NaN	8351	8021	-330	NaN
10	10	NaN	7872	8760	888	NaN
11	11	NaN	7576	11459	3883	NaN
12	12	NaN	7756	17834	10078	NaN
13	13	NaN	7795	19293	11498	NaN
14	14	NaN	7583	16429	8846	NaN
15	15	NaN	7472	12791	5319	NaN
16	16	NaN	7286	10123	2837	NaN
17	17	NaN	7460	8591	1131	NaN
18	18	NaN	7342	8213	871	NaN
19	19	NaN	7195	7114	-81	NaN
20	20	NaN	7073	7210	137	NaN
21	21	NaN	7255	7294	39	NaN
22	22	NaN	7219	6851	-368	NaN
23	23	NaN	6728	6522	-206	NaN

```
24    24     NaN  6829   6601              -228   NaN
25    25     NaN  7112   7131                19   NaN
26    26     NaN  7557   7152              -405   NaN
27    27     NaN  7194   7367               173   NaN
28    28     NaN  6858   7184               326   NaN
29    29     NaN  7000   7343               343   NaN
30    30     NaN  6757   7892              1135   NaN
31    31     NaN  6921   8057              1136   NaN
32    32     NaN  6695   7853              1158   NaN
33    33     NaN  6553   7427               874   NaN
34    34     NaN  6606   7593               987   NaN
35    35     NaN  6517   7287               770   NaN
36    36     NaN  6623   7580               957   NaN
37    37     NaN  6604   7831              1227   NaN
38    38     NaN  6660   7633               973   NaN
39    39     NaN  6707   7803              1096   NaN
40    40     NaN  6941   8128              1187   NaN
41    41     NaN  6999   8087              1088   NaN
42    42     NaN  7092   8949              1857   NaN
43    43     NaN  7278   9149              1871   NaN
44    44     NaN  7311   9742              2431   NaN
45    45     NaN  7628   9584              1956   NaN
46    46     NaN  7887   9398              1511   NaN
47    47     NaN  7836   9093              1257   NaN
48    48     NaN  7603   9046              1443   NaN
49    49     NaN  7895   9067              1172   NaN
50    50     NaN  7895   8908              1013   NaN
51    51     NaN  7577   8705              1128   NaN
52    52  1111.0  3185   6159              2974   NaN
```

[44]:
```
momo2020.merge( df_2021, on=periodo_de_tiempo, how='right')
```

```
␣
↪---------------------------------------------------------------------------

ValueError                                Traceback (most recent call␣
↪last)

<ipython-input-44-22a875393981> in <module>
----> 1 momo2020.merge( df_2021, on=periodo_de_tiempo, how='right')


~/anaconda2/envs/jupyter/lib/python3.6/site-packages/pandas/core/frame.
↪py in merge(self, right, how, on, left_on, right_on, left_index, right_index,␣
↪sort, suffixes, copy, indicator, validate)
7347             copy=copy,
```

```
   7348             indicator=indicator,
-> 7349             validate=validate,
   7350         )
   7351
```

```
~/anaconda2/envs/jupyter/lib/python3.6/site-packages/pandas/core/reshape/merge.py in merge(left, right, how, on, left_on, right_on, left_index, right_index, sort, suffixes, copy, indicator, validate)
     79         copy=copy,
     80         indicator=indicator,
---> 81         validate=validate,
     82     )
     83     return op.get_result()
```

```
~/anaconda2/envs/jupyter/lib/python3.6/site-packages/pandas/core/reshape/merge.py in __init__(self, left, right, how, on, left_on, right_on, axis, left_index, right_index, sort, suffixes, copy, indicator, validate)
    628         # validate the merge keys dtypes. We may need to coerce
    629         # to avoid incompat dtypes
--> 630         self._maybe_coerce_merge_keys()
    631
    632         # If argument passed to validate,
```

```
~/anaconda2/envs/jupyter/lib/python3.6/site-packages/pandas/core/reshape/merge.py in _maybe_coerce_merge_keys(self)
   1136                     inferred_right in string_types and inferred_left not in string_types
   1137                 ):
-> 1138                     raise ValueError(msg)
   1139
   1140             # datetimelikes must match exactly
```

```
ValueError: You are trying to merge on int64 and object columns. If you wish to proceed you should use pd.concat
```

[ ]:

[43]:
```
df_2021.dtypes
```

[43]:
```
week    object
2021     int64
dtype: object
```

[ ]: