

Madrid_Pain_Graphs

June 12, 2020

```
[1]: ! cd ../data/; FILELIST=" 200509 200508 200507 200506 200505 200504 200503_
↪200502 200501 200430 200429 200428 200427 200426 200425 200424 200423 200422_
↪200510 200511 200512 200513 200514 200515 200516 200517 200518 200519 200520_
↪200521 200522 200523 200524 200525 200526 200527 200528 200529 200530 200609_
↪200608 200607 200606 200605 200604 200603 200602 200601 200610 200611 200612_
↪200613 200614 200615 200616 200617 200618 200619 200620 200621 200622 200623_
↪200624 200625 200626 200627 200628 200629 200630 " ; for fecha in `echo_
↪$FILELIST` ; do FILE=${fecha}_cam_covid19.pdf ; [ ! -f ../data/${FILE} ]_
↪&& echo $FILE:~~~~ && wget https://www.comunidad.madrid/sites/default/_
↪files/doc/sanidad/${FILE} 1>/dev/null 2>/dev/null && ls -altr $FILE ; done

! pwd
```

```
200612_cam_covid19.pdf:~~~~
200613_cam_covid19.pdf:~~~~
200614_cam_covid19.pdf:~~~~
200615_cam_covid19.pdf:~~~~
200616_cam_covid19.pdf:~~~~
200617_cam_covid19.pdf:~~~~
200618_cam_covid19.pdf:~~~~
200619_cam_covid19.pdf:~~~~
200620_cam_covid19.pdf:~~~~
200621_cam_covid19.pdf:~~~~
200622_cam_covid19.pdf:~~~~
200623_cam_covid19.pdf:~~~~
200624_cam_covid19.pdf:~~~~
200625_cam_covid19.pdf:~~~~
200626_cam_covid19.pdf:~~~~
200627_cam_covid19.pdf:~~~~
200628_cam_covid19.pdf:~~~~
200629_cam_covid19.pdf:~~~~
200630_cam_covid19.pdf:~~~~
/root/scripts/COVID-19/jupyter
```

```
[2]: from tabula import read_pdf
from IPython.display import display, HTML
import os
import pandas as pd
```

```

import glob
import re
from tqdm.notebook import tqdm
import warnings
warnings.filterwarnings('ignore')

os.environ["JAVA_HOME"] = "/usr/lib/jvm/java-1.8.0-openjdk-1.8.0.141-1.b16.
↳el7_3.x86_64/jre"

# Auxiliary functions
from datetime import datetime, date, time, timedelta

""" Rellenar dias vacios con interpolacion """
def interpolate_dataframe(df,freq):
    if freq == 'H':
        rng = pd.date_range(df.index.min(), df.index.max() + pd.Timedelta(23,
↳'H'), freq='H')
    elif freq == 'D' :
        rng = pd.date_range(
            datetime.strptime(str(df.index.min())[:10]+' 00:00:00',
↳"%Y-%m-%d %H:%M:%S") ,
            datetime.strptime(str(df.index.max())[:10]+' 00:00:00',
↳"%Y-%m-%d %H:%M:%S"),
            freq='D')
        df.index = pd.to_datetime(df.index)
        df2 = df.reindex(rng)
        df = df2
        for column in df.columns :
            s = pd.Series(df[column])
            s.interpolate(method="quadratic", inplace =True)
            df[column] = pd.DataFrame([s]).T
        return df

def fet_daily_date_new_format(fecha):
    df_pdf = read_pdf('../data/'+fecha+'_cam_covid19.pdf',area=(000, 600, 400,
↳800) , pages='1')
    df = df_pdf[0]
    df = df['Unnamed: 0'].astype(str).str.replace(r".", ' ').replace("(", ' ')
    df = df.T
    df.columns = df.iloc[0]
    df = df.iloc[1:]

    df = pd.DataFrame(data=df)
    df

```

```

dict = {}
dict['HOSPITALES'] = df[df['Unnamed: 0'].str.contains('Hospitales')].
→iloc[0]['Unnamed: 0'].split(' ')[0]
dict['DOMICILIOS'] = df[df['Unnamed: 0'].str.contains('Domicilios')].
→iloc[0]['Unnamed: 0'].split(' ')[0]
dict['CENTROS SOCIO SANITARIOS'] = df[df['Unnamed: 0'].str.
→contains('Centros')].iloc[0]['Unnamed: 0'].split(' ')[0]
dict['OTROS LUGARES'] = df[df['Unnamed: 0'].str.contains('otros')].
→iloc[0]['Unnamed: 0'].split(' ')[0]

cadena_a_parsear = df[df['Unnamed: 0'].str.contains('otal')].
→iloc[0]['Unnamed: 0']

dict['FALLECIDOS TOTALES'] = re.search(r'(\d+)', cadena_a_parsear)[0]

df = pd.DataFrame.from_dict(dict, orient='index').T
df['Fecha'] = pd.to_datetime(fecha, format='%y%m%d')
df.set_index('Fecha', inplace=True, drop=True)
return df

def get_daily_data(fecha):
    if fecha > '200512' :
        return fet_daily_date_new_format(fecha)

    col2str = {'dtype': str}
    kwargs = {'output_format': 'dataframe',
              'pandas_options': col2str,
              'stream': True}
    df_pdf = read_pdf('../data/'+fecha+'_cam_covid19.
→pdf', pages='1', multiple_tables = True, **kwargs)

    df = df_pdf[0]

    df = df[df['Unnamed: 0'].notna()]
    df = df[(df['Unnamed: 0']=='HOSPITALES') | (df['Unnamed: 0'] ==
→'DOMICILIOS') | (df['Unnamed: 0'] == 'CENTROS SOCIO SANITARIOS') |
→(df['Unnamed: 0'] == 'OTROS LUGARES') | (df['Unnamed: 0'] == 'FALLECIDOS
→TOTALES')]
    df = df[['Unnamed: 0', 'Unnamed: 2']]
    df['Unnamed: 2'] = df['Unnamed: 2'].astype(str).str.replace(r".", '')
    df = df.T
    df.columns = df.iloc[0]
    df = df.iloc[1:]

```

```

df['Fecha'] = pd.to_datetime(fecha, format='%y%m%d')
df = df.rename_axis(None)

df.set_index('Fecha', inplace=True, drop=True)
df.index
df.dropna()
#df = df.T
return df

def get_all_data( ):
    #BLACKLIST = ["200429", "200422"]
    #BLACKLIST = ["200514",]
    BLACKLIST = []
    df = pd.DataFrame()
    list_df = []

    pdf_list= sorted(glob.glob('../data/*_cam_covid19.pdf'),
                     key=os.path.getmtime,
                     reverse=True )

    #for pdf_file in pdf_list:

    for pdf_file in tqdm(pdf_list,
                          desc="Procesando pdfs diarios"):
        # extract fecha from username , eg : ../data/2200422_cam_covid19.pdf
        fecha = pdf_file.split('/')[2].split('_')[0]
        if fecha not in BLACKLIST:
            #print("processing", fecha)
            df = get_daily_data(fecha)
            list_df.append(df)

    df = pd.concat(list_df)
    df = df.astype(int)
    df = df.drop_duplicates()

    df = df.sort_values(by=['Fecha'], ascending=True)
    ###jaime
    #df = interpolate_dataframe(df, 'D')
    #df.index.name = 'Fecha'

    df['HOSPITALES hoy'] = df['HOSPITALES'] - df['HOSPITALES'].shift(1)
    df['CENTROS SOCIOSANITARIOS hoy'] = df['CENTROS SOCIOSANITARIOS'] -
    ↪df['CENTROS SOCIOSANITARIOS'].shift(1)
    df['FALLECIDOS TOTALES hoy'] = df['FALLECIDOS TOTALES'] - df['FALLECIDOS_
    ↪TOTALES'].shift(1)

    df = df.sort_values(by=['Fecha'], ascending=False)

```

```

    return df

total = get_all_data()

```

HBox(children=(FloatProgress(value=0.0, description='Procesando pdfs diarios', max=51.0, style=

```

Got stderr: Jun 12, 2020 9:39:50 AM
org.apache.pdfbox.pdmodel.font.PDCIDFontType2 <init>
INFO: OpenType Layout tables used in font CIDFont+F1 are not implemented in
PDFBox and will be ignored
Jun 12, 2020 9:39:50 AM org.apache.pdfbox.pdmodel.font.PDCIDFontType2 <init>
INFO: OpenType Layout tables used in font CIDFont+F2 are not implemented in
PDFBox and will be ignored
Jun 12, 2020 9:39:50 AM org.apache.pdfbox.pdmodel.font.PDCIDFontType2 <init>
INFO: OpenType Layout tables used in font CIDFont+F3 are not implemented in
PDFBox and will be ignored
Jun 12, 2020 9:39:50 AM org.apache.pdfbox.pdmodel.font.PDCIDFontType2 <init>
INFO: OpenType Layout tables used in font CIDFont+F1 are not implemented in
PDFBox and will be ignored
Jun 12, 2020 9:39:50 AM org.apache.pdfbox.pdmodel.font.PDCIDFontType2 <init>
INFO: OpenType Layout tables used in font CIDFont+F2 are not implemented in
PDFBox and will be ignored
Jun 12, 2020 9:39:50 AM org.apache.pdfbox.pdmodel.font.PDCIDFontType2 <init>
INFO: OpenType Layout tables used in font CIDFont+F3 are not implemented in
PDFBox and will be ignored
Jun 12, 2020 9:39:51 AM org.apache.pdfbox.pdmodel.font.PDCIDFontType2 <init>
INFO: OpenType Layout tables used in font CIDFont+F1 are not implemented in
PDFBox and will be ignored
Jun 12, 2020 9:39:51 AM org.apache.pdfbox.pdmodel.font.PDCIDFontType2 <init>
INFO: OpenType Layout tables used in font CIDFont+F2 are not implemented in
PDFBox and will be ignored
Jun 12, 2020 9:39:51 AM org.apache.pdfbox.pdmodel.font.PDCIDFontType2 <init>
INFO: OpenType Layout tables used in font CIDFont+F3 are not implemented in
PDFBox and will be ignored

```

```

[3]: total
      VENTANA_MEDIA_MOVIL=7
df = interpolate_dataframe(total,'D')
df.index.name = 'Fecha'
df = df.sort_values(by=['Fecha'], ascending=True)
df['HOSPITALES hoy'] = df['HOSPITALES'] - df['HOSPITALES'].shift(1)
df['CENTROS SOCIO SANITARIOS hoy'] = df['CENTROS SOCIO SANITARIOS'] - df['CENTROS_
↳SOCIO SANITARIOS'].shift(1)

```

```

df['FALLECIDOS TOTALES hoy'] = df['FALLECIDOS TOTALES'] - df['FALLECIDOS_
↳TOTALES'].shift(1)

df['MA CENTROS SOCIO SANITARIOS hoy'] = df['CENTROS SOCIO SANITARIOS hoy'].
↳rolling(window=VENTANA_MEDIA_MOVIL).mean()
df['MA HOSPITALES hoy'] = df['HOSPITALES hoy'].
↳rolling(window=VENTANA_MEDIA_MOVIL).mean()
df['MA FALLECIDOS TOTALES hoy'] = df['FALLECIDOS TOTALES hoy'].
↳rolling(window=VENTANA_MEDIA_MOVIL).mean()

df = df.sort_index(ascending=False)
df_master = df.copy()

```

```

[4]: from matplotlib import colors

def background_gradient(s, m, M, cmap='PuBu', low=0, high=0):
    rng = M - m
    norm = colors.Normalize(m - (rng * low),
                             M + (rng * high))
    normed = norm(s.values)
    c = [colors.rgb2hex(x) for x in plt.cm.get_cmap(cmap)(normed)]
    return ['background-color: %s' % color for color in c]

df.style.format ({ c : "{:20,.0f}" for c in df.columns }).
↳background_gradient(cmap='Wistia', subset= df.columns[-3:] )

```

```

[4]: <pandas.io.formats.style.Styler at 0x7f0178849588>

```

```

[5]: HTML("<h2>Gráfico muertes diarias en Madrid, según Comunidad de Madrid </h2>")

```

```

[5]: <IPython.core.display.HTML object>

```

```

[6]: import pandas as pd
import io
import matplotlib.dates as mdates
from matplotlib import pyplot as plt

df = df_master
chart_df=df[df.columns[-3:]]
chart_df.plot(legend=True,figsize=(13.5,9), marker='o')

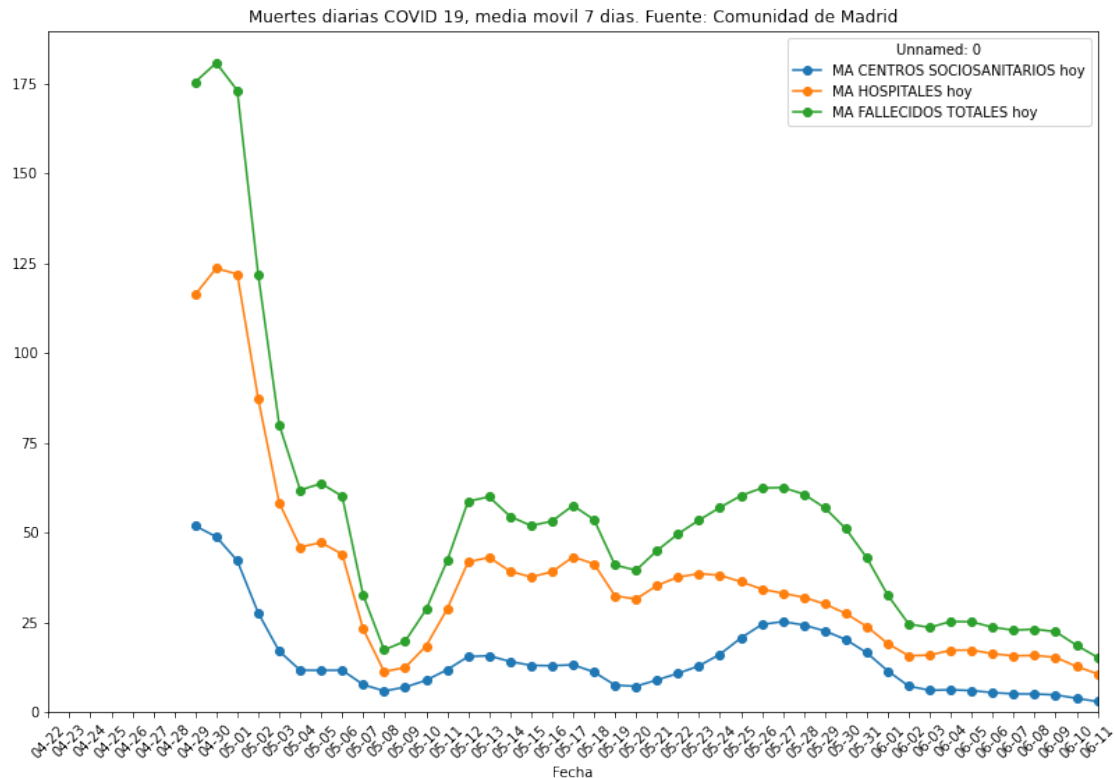
plt.gca().xaxis.set_major_formatter(mdates.DateFormatter('%m-%d'))
plt.gca().xaxis.set_major_locator(mdates.DayLocator(interval=1))
plt.xticks(rotation=45)

ax = plt.gca()

```

```
ax.set_title("Muertes diarias COVID 19, media movil_
↳"+str(VENTANA_MEDIA_MOVIL)+" dias. Fuente: Comunidad de Madrid")
ax.set_ylim(ymin=0)

plt.show()
```



```
[7]: from IPython.display import display, HTML
HTML("<h2>Comparamos los datos de hoy, de hace una semana y de un mes </h2>")
```

```
[7]: <IPython.core.display.HTML object>
```

```
[8]: df = df_master
pd.concat([df.head(1).tail(1) , df.head(7).tail(1) , df.head(30).tail(1)]).
↳astype(int)[['MA HOSPITALES hoy','MA CENTROS SOCIO SANITARIOS hoy','MA_
↳FALLECIDOS TOTALES hoy']].style.format( { c : "{:20,.0f}" for c in df.
↳columns }).background_gradient(cmap='Wistia', subset= df.columns[-3: ] )
```

```
[8]: <pandas.io.formats.style.Styler at 0x7f013a24a0f0>
```

```
[9]: from IPython.display import display, HTML
HTML("<h2>Muertes medias diarias, últimos 7 días, con datos</h2>")
```

[9]: <IPython.core.display.HTML object>

```
[10]: from datetime import date

df = df_master
inicio_crisis = df.head(7).index[6]
df=df.head(7)
dia_mas_reciente = df.index[0]
dias_transcurridos_inicio_crisis = dia_mas_reciente - inicio_crisis
df = pd.DataFrame((df.head(1).max(axis=0) - df.tail(1).max(axis=0) ) / 
↳dias_transcurridos_inicio_crisis.days ).
↳T[['HOSPITALES', 'DOMICILIOS', 'CENTROS SOCIO SANITARIOS', 'OTROS',
↳LUGARES', 'FALLECIDOS TOTALES']]
df.style.format ( { c : "{:20,.0f}" for c in df.columns } ).
↳background_gradient(cmap='Wistia' )
```

[10]: <pandas.io.formats.style.Styler at 0x7f01381f3240>

```
[11]: HTML("<h2>Muertes medias diarias desde que la comunidad de Madrid publica
↳datos</h2>")
```

[11]: <IPython.core.display.HTML object>

```
[12]: # Calculamos los incrementos medios, desde que tenemos fechas
df = df_master
df = pd.DataFrame((df.head(1).max(axis=0) - df.tail(1).max(axis=0) ) / df.
↳shape[0] ).T[['HOSPITALES', 'DOMICILIOS', 'CENTROS SOCIO SANITARIOS', 'OTROS',
↳LUGARES', 'FALLECIDOS TOTALES']]
df.style.format ( { c : "{:20,.0f}" for c in df.columns } ).
↳background_gradient(cmap='Wistia' )
```

[12]: <pandas.io.formats.style.Styler at 0x7f0138229e48>

[]: