

# Madrid\_Pain\_Graphs

February 24, 2021

## 1 Informes de la comunidad de Madrid

Actualizado diariamente, este documento se [visualiza mejor aquí](#).

Datos de la situación de la infección por coronavirus en la Comunidad de Madrid.

Nos descargamos los datos, agrupamos, y calculamos :

- Gráfico de seguimiento.
- Muertes medias diarias, últimos 7 días.
- Muertes medias diarias desde que la comunidad de Madrid publica datos.

```
[1]: # Miramos si hay nuevos datos a descargar.

!# 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

# Miramos solo hoy y los ultimos diez dias
! cd ../data/; FILELIST=`seq -w 0 2 | while read i ; do date +%y%m%d -d "$i day_
→ago" ; done` ; for fecha in `echo $FILELIST` ; do _
→FILE=${fecha}_cam_covid19.pdf ; [ ! -f ../data/${FILE} ] && echo $FILE:::::_
→ && wget https://www.comunidad.madrid/sites/default/files/aud/sanidad/
→$FILE 1>/dev/null 2>/dev/null && ls -altr $FILE ; done

! cd ../data/; FILELIST=`seq -w 0 2 | while read i ; do date +%y%m%d -d "$i day_
→ago" ; done` ; 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
```

```

! cd ../data/; FILELIST=`seq -w 0 2 | while read i ; do date +%y%m%d -d "$i day_
→ago" ; done` ; 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 $FILE ; done
! cd ../data/; FILELIST=`seq -w 0 2 | while read i ; do date +%Y%m%d -d "$i day_
→ago" ; done` ; 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 $FILE ; done
! cd ../data/; FILELIST=`seq -w 0 2 | while read i ; do date +%y%m%d -d "$i day_
→ago" ; done` ; for fecha in `echo $FILELIST` ; do
→FILE=${fecha}_cam_covid19.pdf ; [ ! -f ../data/${FILE} ] && echo $FILE:::::
→ && wget https://www.comunidad.madrid/sites/default/files/${FILE} 1>/dev/
→null 1>/dev/null 2>/dev/null && ls -altr $FILE ; done
! cd ../data/; FILELIST=`seq -w 0 2 | while read i ; do date +%-%d.%-m.%Y -d "$i_
→day ago" ; done` ; for fecha in `echo $FILELIST` ; do FILE=${fecha}_2.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 $FILE ; done
! cd ../data/; FILELIST=`seq -w 0 2 | while read i ; do date +%-%d.%-m.%Y -d "$i_
→day ago" ; done` ; for fecha in `echo $FILELIST` ; do FILE=${fecha}.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 $FILE ; done
! cd ../data/; FILELIST=`seq -w 0 2 | while read i ; do date +%y%m%d -d "$i day_
→ago" ; done` ; 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

```

```

210130cam_covid19.pdf:::::
210129cam_covid19.pdf:::::
210128cam_covid19.pdf:::::
20210130_cam_covid19.pdf:::::
20210129_cam_covid19.pdf:::::
20210128_cam_covid19.pdf:::::
30.1.2021_2.pdf:::::
29.1.2021_2.pdf:::::
28.1.2021_2.pdf:::::
30.1.2021.pdf:::::
29.1.2021.pdf:::::
28.1.2021.pdf:::::
210130cam_covid19.pdf:::::
210129cam_covid19.pdf:::::
210128cam_covid19.pdf:::::

```

```

[ ]: 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
import os.path
#import datetime
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

df_cache = pd.read_csv("/root/kaggle/covid19-madrid/madrid_results.csv")

def query_cache(fecha):
    """ Query cache file to avoid parse pdf
    return empty dataframe is not found"""
    try :
        if '.' in fecha:
            date_regexp='%d.%m.%Y'
        else:
            date_regexp='%y%m%d'
        date_formatted = datetime.strptime(fecha,date_regexp ).
↳strptime('%Y-%m-%d')
        df = df_cache.query( 'Fecha==@date_formatted')
    except:
        print("Cache miss:" , fecha)
        return pd.DataFrame()
        #print(f"fecha {fecha},{date_formatted}")

    try:
        df['Fecha'] = pd.to_datetime(date_formatted, format='%y-%m-%d')
    except :
        df['Fecha'] = pd.to_datetime(date_formatted, format='%Y-%m-%d')
    df.set_index('Fecha', inplace=True, drop=True)
    return df

""" 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 get_daily_date_new_format(fecha,filename):

    file_path = '../data/'+fecha+'_cam_covid19.pdf'
    if not os.path.isfile(file_path):
        file_path = '../data/'+fecha+'cam_covid19.pdf'
    if not os.path.isfile(file_path):
        file_path = filename

    #print("Analizando:" + file_path)
    df_pdf = read_pdf(file_path,area=(000, 600, 400, 800) , pages='1')

    # Parche, para los saltos de linea en el pdf
    if 'Unnamed: 0' not in df_pdf[0].columns :
        #df_pdf = read_pdf(file_path,area=(000, 500, 400, 800) , pages='1')
        return pd.DataFrame()

    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:]

    #print("2 get_daily_date_new_format")

    df = pd.DataFrame(data=df)
    df

    dict = {}

```

```

try:
    df2_pdf = read_pdf(file_path, area=(300, 100, 800, 400) , pages='1')
    dict['PACIENTES UCI DIA'] = df2_pdf[0].loc[3:3].values[0][1].
    ↪replace(".", '')
    dict['PACIENTES UCI ACUMULADOS'] = df2_pdf[0].loc[6:6].values[0][1].
    ↪replace(".", '')
    except Exception as e:
        print(f"{fecha} mal parseada: {e}")

    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]
    #print("3 get_daily_date_new_format")

    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]

    #print("4 get_daily_date_new_format")
    df = pd.DataFrame.from_dict(dict, orient='index').T
    #print("4.5 get_daily_date_new_format")

    if '.' in fecha :
        try:
            df['Fecha'] = pd.to_datetime(fecha, format='%d.%m.%Y')
        except :
            df['Fecha'] = pd.to_datetime(fecha, format='%d.%m.%y')
    else:
        try:
            df['Fecha'] = pd.to_datetime(fecha, format='%y%m%d')
        except :
            df['Fecha'] = pd.to_datetime(fecha, format='%Y%m%d')

    #print("5 get_daily_date_new_format")

    df.set_index('Fecha', inplace=True, drop=True)
    #print(df)
    return df

```

```

def get_daily_data(fecha,filename):
    #print(f"""get_daily_data: {fecha}""")
    #print(f"""../data/{fecha}_cam_covid19.pdf""")

    if fecha > '200512' :
        return get_daily_date_new_format(fecha,filename)

    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 SOCIOSANITARIOS') |
→(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= (glob.glob('../data/*_covid19.pdf'),
    #           key=os.path.getmtime,

```

```

#                                     reverse=True )
pdf_list= set(glob.glob('../data/*202*pdf') + glob.glob('../data/
↪*cam_covid19.pdf'))

for pdf_file in tqdm(pdf_list,
                      desc="Procesando pdfs diarios"):
    # extract fecha from username , eg : ../data/2200422_cam_covid19.pdf

    format_point_occurences = pdf_file.split('/')[2].split('_')[0].count(".")
    ↪")

    # Hack to fix filename inconsistencies on remote server
    if format_point_occurences > 2 :
        day = pdf_file.split('/')[2].split('_')[0].split('.')[0].zfill(2)
        month = pdf_file.split('/')[2].split('_')[0].split('.')[1].zfill(2)
        year = pdf_file.split('/')[2].split('_')[0].split('.')[2][-2:]
        fecha = year+month+day
        fecha=fecha.replace('.pdf','')
    else :
        fecha = pdf_file.split('/')[2].split('_')[0].replace('cam_','').
    ↪replace('_cam_','').replace('cam','')
        if fecha not in BLACKLIST:
            # query cache, otherwise parse pdf
            df = query_cache(fecha)
            if df.empty:
                df = get_daily_data(fecha,pdf_file)
            list_df.append(df)

df = pd.concat(list_df)
df = df.fillna(0)
df = df.astype(int)
df = df.drop_duplicates()

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

return df

total = get_all_data()

```

```
total.to_csv('/root/kaggle/covid19-madrid/madrid_results.csv')
total
```

HBox(children=(FloatProgress(value=0.0, description='Procesando pdfs diarios', max=246.0, style=

Got stderr: ene 30, 2021 7:44:41 PM

org.apache.pdfbox.pdmodel.font.PDTrueTypeFont <init>

ADVERTENCIA: Using fallback font 'LiberationSans' for 'Arial,Bold'

Cache miss: 20200814

```
[ ]: def get_daily_date_new_format(fecha,filename):

    file_path = '../data/'+fecha+'_cam_covid19.pdf'
    if not os.path.isfile(file_path):
        file_path = '../data/'+fecha+'cam_covid19.pdf'
    if not os.path.isfile(file_path):
        file_path = filename

    #print("Analizando:" + file_path)
    df_pdf = read_pdf(file_path,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:]

    #print("2 get_daily_date_new_format")

    df = pd.DataFrame(data=df)
    df

    dict = {}
    try:
        df2_pdf = read_pdf(file_path,area=(300, 100, 800, 400) , pages='1')
        dict['PACIENTES UCI DIA'] = df2_pdf[0].loc[3:3].values[0][1].
        ↪replace(".", '')
        dict['PACIENTES UCI ACUMULADOS'] = df2_pdf[0].loc[6:6].values[0][1].
        ↪replace(".", '')
    except Exception as e:
        print(f"{fecha} mal parseada: {e}")

    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]
    #print("3 get_daily_date_new_format")

    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]

    #print("4 get_daily_date_new_format")
    df = pd.DataFrame.from_dict(dict, orient='index').T
    #print("4.5 get_daily_date_new_format")

    if '.' in fecha :
        try:
            df['Fecha'] = pd.to_datetime(fecha, format='%d.%m.%Y')
        except :
            df['Fecha'] = pd.to_datetime(fecha, format='%d.%m.%y')
    else:
        try:
            df['Fecha'] = pd.to_datetime(fecha, format='%y%m%d')
        except :
            df['Fecha'] = pd.to_datetime(fecha, format='%Y%m%d')

    #print("5 get_daily_date_new_format")

    df.set_index('Fecha', inplace=True, drop=True)
    #print(df)
    return df

```

```

[4]: """ debug"""
fecha='201210'
filename='/root/scripts/COVID-19/data/201210_cam_covid19.pdf'
file_path=filename
df_pdf = read_pdf(file_path,area=(000, 600, 400, 800) , pages='1')
df_pdf

dict={}
df2_pdf = read_pdf(file_path,area=(300, 000, 800, 300) , pages='1')
df2_pdf
dict['PACIENTES UCI DIA'] = df2_pdf[0].loc[3:3].values[0][1].replace(".",
    ↪",')

```

```
dict['PACIENTES UCI ACUMULADOS']= df2_pdf[0].loc[6:6].values[0][1].replace(".",
↳ ", '')
```

Got stderr: ene 21, 2021 5:46:44 PM

org.apache.pdfbox.pdmodel.font.PDTrueTypeFont <init>

ADVERTENCIA: Using fallback font 'LiberationSans' for 'Arial,Bold'

Got stderr: ene 21, 2021 5:46:45 PM

org.apache.pdfbox.pdmodel.font.PDTrueTypeFont <init>

ADVERTENCIA: Using fallback font 'LiberationSans' for 'Arial,Bold'

```
↳ -----
```

```
AttributeError                                Traceback (most recent call↳
↳ last)
```

```
<ipython-input-4-f6adfd8e8bfe> in <module>
      9 df2_pdf = read_pdf(file_path,area=(300, 000, 800, 300) , pages='1')
     10 df2_pdf
--> 11 dict['PACIENTES UCI DIA']          = df2_pdf[0].loc[3:3].values[0][1].
↳ replace(".", ' ')
     12 dict['PACIENTES UCI ACUMULADOS']= df2_pdf[0].loc[6:6].values[0][1].
↳ replace(".", ' ')
```

AttributeError: 'float' object has no attribute 'replace'

```
[ ]: total
df = total
df = df.fillna(0)
df = df.astype(int)
df
```

```
[ ]: 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()

```

```
[ ]: total.head()
```

```

[ ]: # Hacemos lo contrario
# En lugar de sacar el nº de muertos dado el nº de infectados, como lo primero
    ↳lo sabemos (en madrid), sacamos lo segundo y extrapolamos al conjunto de
    ↳españa
df = df_master

R0_estimada = df['FALLECIDOS TOTALES hoy'].values[0:7].sum() / df['FALLECIDOS_
    ↳TOTALES hoy'].values[7:14].sum()
print(df['FALLECIDOS TOTALES hoy'].values[0:7].sum(), df['FALLECIDOS TOTALES_
    ↳hoy'].values[7:14].sum() )
print(f"R0_estimada = {R0_estimada}")
PROPORCION_ENFERMOS_MUERTOS=750000/15000 # Esta es la proporcion enfermos
    ↳muertos (15.000 muertos para 750.000 afectados)
RATIO_NO_HEMOS_COLAPSADO=2 # La mitad de los muertos se ha calculado del
    ↳colapso. Como ahora no hemos colapsado
PESO_MADRID_MUERTES_TOTALES=1/3
casos_españa_estimados = df['FALLECIDOS TOTALES hoy'].values[0:5].sum() *
    ↳PROPORCION_ENFERMOS_MUERTOS * RATIO_NO_HEMOS_COLAPSADO /
    ↳PESO_MADRID_MUERTES_TOTALES
print(f"casos_españa_estimados = {casos_españa_estimados}")

```

## 1.1 Gráfico estimacion R0

Considerando solo los datos de Madrid, estimamos el R0 a partir del nº de muertos (considerando que el nº de muertos es una combinacion lineal del nº de enfermos), por lo que es posible calcular el ratio igual.

Para calcular el R0, sacamos la suma de muertos de la última semana, entre la suma de muertos de la semana anterior.

```

[ ]: from datetime import datetime, timedelta
import seaborn as sns
from matplotlib import pyplot as plt

```

```

import matplotlib.dates as mdates

df = df_master

def calcular_estimaciones_R0(df):
    def calcular_R0_dia(dia,df):
        dia_semana_anterior = dia - timedelta(days=7)
        return dia,df.loc[dia:dia - timedelta(days=6)][['FALLECIDOS TOTALES_
→hoy']].sum() / df.loc[dia- timedelta(days=7):dia -
→timedelta(days=13)][['FALLECIDOS TOTALES hoy']].sum()

    VENTANA_MEDIA_MOVIL=7

    df_R0_estimada = pd.DataFrame([calcular_R0_dia(dia,df) for dia in df.
→index[0:50]],columns=['Fecha','R0_estimada'])

    df_R0_estimada = df_R0_estimada.sort_values(by=['Fecha'], ascending=True)
    df_R0_estimada['MA_R0_estimada'] = df_R0_estimada['R0_estimada'].
→rolling(window=VENTANA_MEDIA_MOVIL).mean()
    df_R0_estimada = df_R0_estimada.sort_values(by=['Fecha'], ascending=False)
    df_R0_estimada.set_index('Fecha', inplace=True, drop=True)
    return df_R0_estimada

df= calcular_estimaciones_R0(df_master)
#df=df[['R0_estimada']]
df

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.axhline(1, color='r',linestyle = ':' )

ax.set_title("Estimacion R0 Comunidad de Madrid")
ax.set_ylim(ymin=0)

plt.show()

df.style.format ({ c : "{:20,.3f}" for c in df.columns }).
→background_gradient(cmap='Wistia', )

```

```
[ ]: RO_estimada * 1.2
```

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

```
[ ]: 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:]].head(60)
chart_df.plot(legend=True,figsize=(13.5,9), marker='o')

plt.gca().xaxis.set_major_formatter(mdates.DateFormatter('%b-%d'))
plt.gca().xaxis.set_major_locator(mdates.DayLocator(interval=7))
plt.xticks(rotation=45)

ax = plt.gca()
plt.setp(ax.get_xminorticklabels(), visible=False)

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()
```

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

```
[ ]: 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 = df_master

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

```
[ ]: df = df_master
```

```
pd.concat([df.head(1).tail(1) , df.head(8).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:] )
```

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

```
[ ]: 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' )
```

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

```
[ ]: # 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' )
```

```
[ ]:
```

```
[ ]: 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
import os.path
fecha="201005"
import os
```

```
file_path = '../data/'+fecha+'_cam_covid19.pdf'
if not os.path.isfile(file_path):
    file_path = '../data/'+fecha+'cam_covid19.pdf'
#print("Analizando:" + file_path)
```

```
[ ]: df_pdf = read_pdf(file_path,area=(300, 100, 800, 400) , pages='1')
df_pdf
```

```
[ ]: for x,y in enumerate(df_pdf):
    print(x,":",y)

pd.DataFrame(df_pdf)
```

```
[ ]: type(df_pdf)
```

```
[ ]: type(df_pdf[0])
```

```
[ ]: total
```

```
[ ]: get_daily_date_new_format("201005")
```

```
[ ]: total
```

```
[ ]:
```