

Madrid_Pain_Graphs

September 21, 2020

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 10 | 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 $FILE ; done
! cd ../data/; FILELIST=`seq -w 0 10 | while read i ; do date +%y%m%d -d "$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 10 | 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 $FILE ; done
! cd ../data/; FILELIST=`seq -w 0 10 | while read i ; do date +%Y%m%d -d "$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 2>/dev/null && ls -altr $FILE ; done
#200902_cam_covid19.pdf
```

```
200920_cam_covid19.pdf:~~~~:
200919_cam_covid19.pdf:~~~~:
200913_cam_covid19.pdf:~~~~:
200912_cam_covid19.pdf:~~~~:
200921cam_covid19.pdf:~~~~:
200920cam_covid19.pdf:~~~~:
200919cam_covid19.pdf:~~~~:
200918cam_covid19.pdf:~~~~:
200917cam_covid19.pdf:~~~~:
200916cam_covid19.pdf:~~~~:
200915cam_covid19.pdf:~~~~:
200914cam_covid19.pdf:~~~~:
200913cam_covid19.pdf:~~~~:
200912cam_covid19.pdf:~~~~:
200911cam_covid19.pdf:~~~~:
20200921_cam_covid19.pdf:~~~~:
20200920_cam_covid19.pdf:~~~~:
20200919_cam_covid19.pdf:~~~~:
20200918_cam_covid19.pdf:~~~~:
20200917_cam_covid19.pdf:~~~~:
20200916_cam_covid19.pdf:~~~~:
20200915_cam_covid19.pdf:~~~~:
20200914_cam_covid19.pdf:~~~~:
20200913_cam_covid19.pdf:~~~~:
20200912_cam_covid19.pdf:~~~~:
20200911_cam_covid19.pdf:~~~~:
200920_cam_covid19.pdf:~~~~:
200919_cam_covid19.pdf:~~~~:
200913_cam_covid19.pdf:~~~~:
200912_cam_covid19.pdf:~~~~:
```

```
[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
import os.path

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 get_daily_date_new_format(fecha):

    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=(000, 600, 400, 800) , pages='1')

    #print("1 get_daily_date_new_format")

    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 = {}
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")

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)
return df

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

    if fecha > '200512':

```

```

        return get_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/*_covid19.pdf'),
                     key=os.path.getmtime,
                     reverse=True )

    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].replace('cam_', '').
→replace('_cam_', '').replace('cam_', '')
        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 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')

```

HBox(children=(FloatProgress(value=0.0, description='Procesando pdfs diarios', max=129.0, style=

Got stderr: sep 21, 2020 6:02:39 PM

org.apache.pdfbox.pdmodel.font.PDCIDFontType2 <init>

INFORMACIÓN: OpenType Layout tables used in font CIDFont+F1 are not implemented in PDFBox and will be ignored

sep 21, 2020 6:02:39 PM org.apache.pdfbox.pdmodel.font.PDCIDFontType2 <init>

INFORMACIÓN: OpenType Layout tables used in font CIDFont+F2 are not implemented in PDFBox and will be ignored

sep 21, 2020 6:02:39 PM org.apache.pdfbox.pdmodel.font.PDCIDFontType2 <init>

INFORMACIÓN: OpenType Layout tables used in font CIDFont+F3 are not implemented in PDFBox and will be ignored

sep 21, 2020 6:02:39 PM org.apache.pdfbox.pdmodel.font.PDCIDFontType2 <init>

INFORMACIÓN: OpenType Layout tables used in font CIDFont+F1 are not implemented in PDFBox and will be ignored

sep 21, 2020 6:02:40 PM org.apache.pdfbox.pdmodel.font.PDCIDFontType2 <init>

INFORMACIÓN: OpenType Layout tables used in font CIDFont+F2 are not implemented

in PDFBox and will be ignored
 sep 21, 2020 6:02:40 PM org.apache.pdfbox.pdmodel.font.PDCIDFontType2 <init>
 INFORMACIÓN: OpenType Layout tables used in font CIDFont+F3 are not implemented
 in PDFBox and will be ignored
 sep 21, 2020 6:02:40 PM org.apache.pdfbox.pdmodel.font.PDCIDFontType2 <init>
 INFORMACIÓN: OpenType Layout tables used in font CIDFont+F1 are not implemented
 in PDFBox and will be ignored
 sep 21, 2020 6:02:40 PM org.apache.pdfbox.pdmodel.font.PDCIDFontType2 <init>
 INFORMACIÓN: OpenType Layout tables used in font CIDFont+F2 are not implemented
 in PDFBox and will be ignored
 sep 21, 2020 6:02:40 PM org.apache.pdfbox.pdmodel.font.PDCIDFontType2 <init>
 INFORMACIÓN: OpenType Layout tables used in font CIDFont+F3 are not implemented
 in PDFBox and will be ignored

```
[3]: interpolate_dataframe(total, 'D')
```

```
[3]: Unnamed: 0    HOSPITALES  DOMICILIOS  CENTROS SOCIOSSANITARIOS  OTROS LUGARES  \
2020-04-22    7144.000000    761.000000                3932.000000    15.000000
2020-04-23    7271.000000    769.000000                3996.000000    20.000000
2020-04-24    7388.000000    775.000000                4068.000000    21.000000
2020-04-25    7633.000000    788.000000                4170.000000    21.000000
2020-04-26    7800.000000    798.000000                4236.000000    21.000000
...
2020-09-17    10058.000000    987.000000                4846.000000    29.000000
2020-09-18    10117.000000    988.000000                4850.000000    30.000000
2020-09-19    10153.813422    988.986683                4852.656676    30.598745
2020-09-20    10167.813422    990.653350                4853.656676    30.598745
2020-09-21    10159.000000    993.000000                4853.000000    30.000000

Unnamed: 0  FALLECIDOS TOTALES  HOSPITALES hoy  CENTROS SOCIOSSANITARIOS hoy  \
2020-04-22          11852.000000                NaN                NaN
2020-04-23          12056.000000            127.000000            64.000000
2020-04-24          12252.000000            117.000000            72.000000
2020-04-25          12612.000000            245.000000           102.000000
2020-04-26          12855.000000            167.000000            66.000000
...
2020-09-17          15920.000000           144.000000             6.000000
2020-09-18          15985.000000             59.000000             4.000000
2020-09-19          16026.055527              1.253408            2.520070
2020-09-20          16042.722193             -4.413259            2.186736
2020-09-21          16035.000000            42.000000            3.000000

Unnamed: 0  FALLECIDOS TOTALES hoy
2020-04-22                NaN
2020-04-23            204.000000
```

2020-04-24	196.000000
2020-04-25	360.000000
2020-04-26	243.000000
...	...
2020-09-17	161.000000
2020-09-18	65.000000
2020-09-19	0.468118
2020-09-20	-4.531882
2020-09-21	50.000000

[153 rows x 8 columns]

[4]: total

[4]: Unnamed: 0	HOSPITALES	DOMICILIOS	CENTROS SOCIO SANITARIOS	OTROS LUGARES	\
Fecha					
2020-09-21	10159	993		4853	30
2020-09-18	10117	988		4850	30
2020-09-17	10058	987		4846	29
2020-09-15	9914	976		4840	29
2020-09-11	9894	976		4838	29
...	
2020-04-26	7800	798		4236	21
2020-04-25	7633	788		4170	21
2020-04-24	7388	775		4068	21
2020-04-23	7271	769		3996	20
2020-04-22	7144	761		3932	15

Unnamed: 0	FALLECIDOS TOTALES	HOSPITALES hoy	CENTROS SOCIO SANITARIOS hoy	\
Fecha				
2020-09-21	16035	42.0		3.0
2020-09-18	15985	59.0		4.0
2020-09-17	15920	144.0		6.0
2020-09-15	15759	20.0		2.0
2020-09-11	15737	50.0		0.0
...	
2020-04-26	12855	167.0		66.0
2020-04-25	12612	245.0		102.0
2020-04-24	12252	117.0		72.0
2020-04-23	12056	127.0		64.0
2020-04-22	11852	NaN		NaN

Unnamed: 0	FALLECIDOS TOTALES hoy
Fecha	
2020-09-21	50.0
2020-09-18	65.0
2020-09-17	161.0

2020-09-15	22.0
2020-09-11	53.0
...	...
2020-04-26	243.0
2020-04-25	360.0
2020-04-24	196.0
2020-04-23	204.0
2020-04-22	NaN

[71 rows x 8 columns]

```
[5]: 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()
```

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

```
[6]: Unnamed: 0  HOSPITALES  DOMICILIOS  CENTROS SOCIO SANITARIOS  OTROS LUGARES  \
Fecha
2020-09-21      10159      993      4853      30
2020-09-18      10117      988      4850      30
2020-09-17      10058      987      4846      29
2020-09-15       9914      976      4840      29
2020-09-11       9894      976      4838      29

Unnamed: 0  FALLECIDOS TOTALES  HOSPITALES hoy  CENTROS SOCIO SANITARIOS hoy  \
Fecha
2020-09-21      16035      42.0      3.0
2020-09-18      15985      59.0      4.0
```

2020-09-17	15920	144.0	6.0
2020-09-15	15759	20.0	2.0
2020-09-11	15737	50.0	0.0

Unnamed: 0 FALLECIDOS TOTALES hoy

Fecha

2020-09-21	50.0
2020-09-18	65.0
2020-09-17	161.0
2020-09-15	22.0
2020-09-11	53.0

```
[7]: # 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}")
```

306.7754779670322 155.22036653628675

R0_estimada = 1.9763867642672752

casos_españa_estimados = 60383.44544910797

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.

```
[8]: 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)

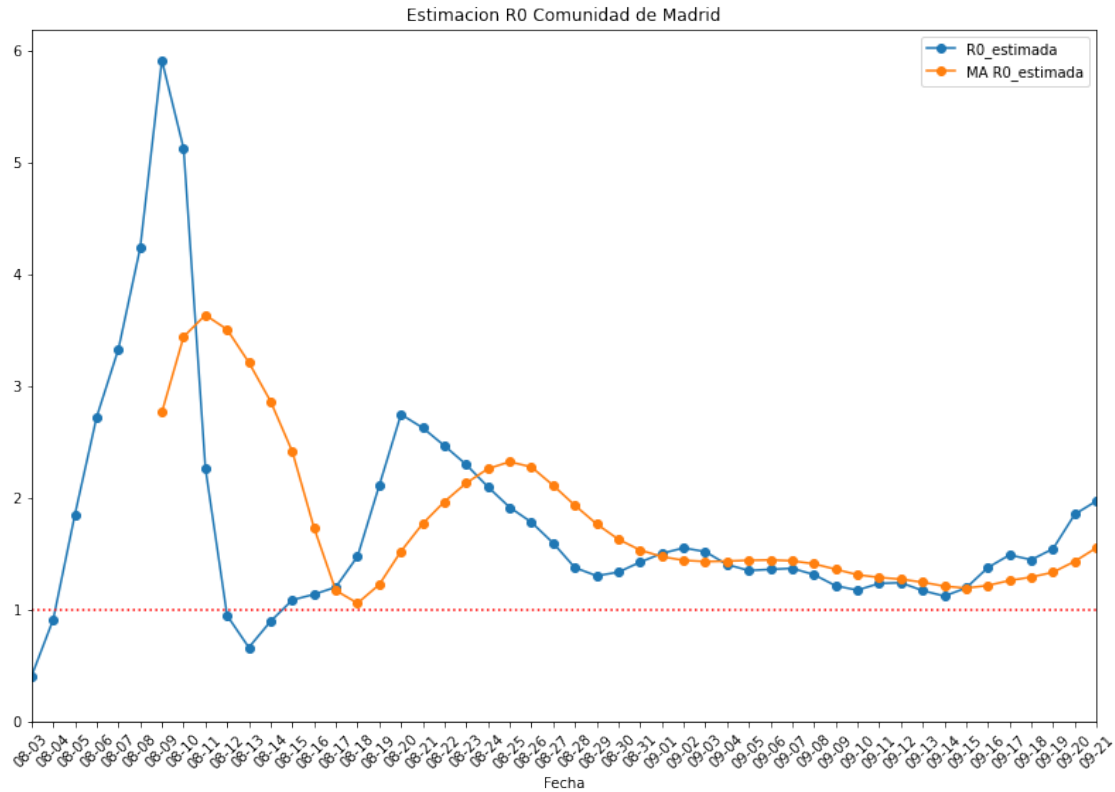
ax = plt.gca()
ax.axhline(1, color='r',linestyle = ':' )

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', )

```



[8]: <pandas.io.formats.style.Styler at 0x7fe4ff04c048>

[9]: R0_estimada * 1.2

[9]: 2.37166411712073

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

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

```
[11]: 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')

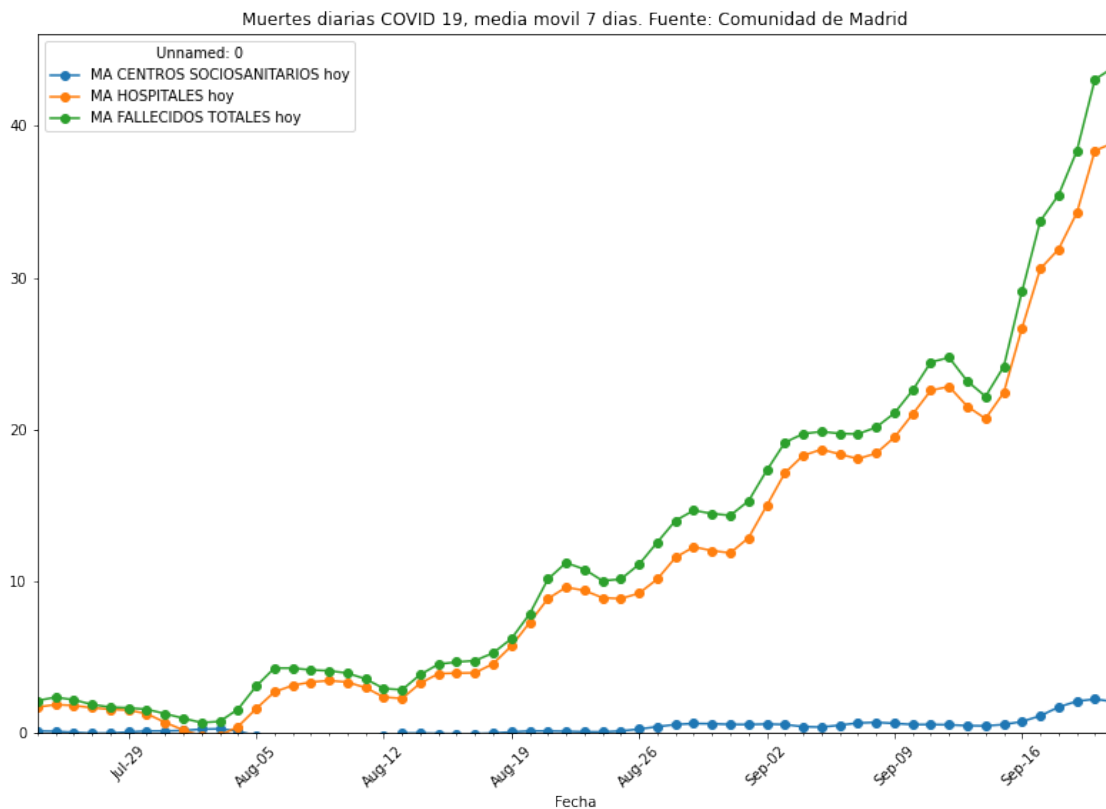
plt.gca().xaxis.set_major_formatter(mdates.DateFormatter('%b-%d'))
plt.gca().xaxis.set_major_locator(mdates.DayLocator(interval=7))
```

```
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()
```



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

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

```
[13]: 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:] )

```

[13]: <pandas.io.formats.style.Styler at 0x7fe50b3ee748>

```

[14]: 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:] )

```

[14]: <pandas.io.formats.style.Styler at 0x7fe4ff71ef98>

```

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

```

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

```

[16]: 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' )

```

[16]: <pandas.io.formats.style.Styler at 0x7fe4fec62470>

```

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

```

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

```
[18]: # 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' )
```

```
[18]: <pandas.io.formats.style.Styler at 0x7fe4fec73550>
```

```
[ ]:
```

```
[ ]:
```