Prevalencia

August 12, 2020

[ ]:

1 Prevalencia

Vamos a analizar el influjo de la prevalencia, en el devenir de la enfermedad

En epidemiología, se denomina prevalencia a la proporción de individuos de un grupo o una población (en medicina, persona), que presentan una característica o evento determinado (en medicina, enfermedades).

Se define R0, como el nº de personas que infecta cada persona, en cada ciclo de infección.

Vamos a generar una lista de tablasy gráficas para ver la evolución del nº de infectados, al variar la prevalencia, y el R0.

[1]:
```
import numpy as np
import pandas as pd
import time
from datetime import datetime, date, time, timedelta
from IPython.display import display, HTML
import matplotlib.dates as mdates

SITUACION_INICIAL = 1
prevalencia = 0
R0 = 5.7
DIAS_EN_REINFECTAR=5
```

[2]:
```
### Calculamos la capacidad del sistema sanitario.
####Cuanto se tardaria en copar las camas de uci en cada escenario
"""
"Antes de la crisis sanitaria, España disponía de unas 4.500 camas UCI,␣
 ↪capacidad que aumentó hasta las 8.000"
Madrid cuenta con 1.750 camas
Cataluña tiene 1.722 camas
Andalucía con 1.200 camas.
Canarias cuenta con 595 camas.
Euskadi con capacidad para 550 camas.
Castilla-León tiene 500 camas.
```

```
Aragón con 300 camas.
Castilla-La Mancha cuenta con 300 camas.
Galicia tiene 274 camas.
Comunidad Valenciana con 254 plazas libres.
Navarra con 156 camas.
Murcia tiene 123 camas.
Baleares con 120 camas.
Extremadura cuenta con 100 camas.
Cantabria con 64 camas.
Asturias cuenta con 61 camas.
La Rioja tiene 23 plazas.
TOTAL = 8092
"""

"De los 11.424 pacientes de Covid-19 ingresados en Madrid, según datos del␣
 ↪Ministerio de Sanidad, 1.332 están en la UCI, un 11,7%."
"Si para una prevalencia de 10% (750000 personas para la comunidad de madrid)"

# Calculamos la capacidad del sistema sanitario - el nº de enfermos que puede␣
 ↪haber antes de que colapse
NUMERO_CAMAS_UCI=8092

PORCENTAJE_ENFERMOS_NECESITADOS_HOSPITALIZACION = 0.088 # https://www.
 ↪redaccionmedica.com/secciones/sanidad-hoy/
 ↪coronavirus-en-personal-sanitario-hospitalizacion-en-el-8-8-de-casos-9925
PORCENTAJE_HOSPITALIZADOS_NECESITADOS_UCI = 0.05  #https://www.elperiodico.com/
 ↪es/sociedad/20200316/coronavirus-hospitalizados-graves-contagio-7891866

CAPACIDAD_SISTEMA_SANITARIO = NUMERO_CAMAS_UCI /␣
 ↪PORCENTAJE_ENFERMOS_NECESITADOS_HOSPITALIZACION /␣
 ↪PORCENTAJE_HOSPITALIZADOS_NECESITADOS_UCI
CAPACIDAD_SISTEMA_SANITARIO = int(CAPACIDAD_SISTEMA_SANITARIO)

print ("La estimacion de la capacidad del sistema sanitario es " ,␣
 ↪CAPACIDAD_SISTEMA_SANITARIO )
```

```
La estimacion de la capacidad del sistema sanitario es  1839090
```

[3]:
```
def Get_Header(GENERACIONES,df,FECHA_INICIAL_STR = '2020-02-01'):
    array_fechas = []
    FECHA_INICIAL = datetime.strptime(FECHA_INICIAL_STR, "%Y-%m-%d")
    modified_date = FECHA_INICIAL
    NUM_GENERACIONES = range(1,GENERACIONES)
    for generacion in NUM_GENERACIONES:
        modified_date += timedelta(days=DIAS_EN_REINFECTAR)
        array_fechas.append(datetime.strftime(modified_date, "%Y-%m-%d"))
    df.columns = array_fechas
```

```
    return df

def Calcular_Cuadro_Prevalencias(R0,GENERACIONES,ARRAY_PREVALENCIAS,␣
↪SITUACION_INICIAL=1,FECHA_INICIAL_STR = '2020-02-01'):
    diccionario_prevalencias = {}
    array=[]

    for prevalencia in ARRAY_PREVALENCIAS :
        infectados_en_esta_generacion = SITUACION_INICIAL
        NUM_GENERACIONES = range(1,GENERACIONES)
        array=[]
        for generacion in NUM_GENERACIONES:
            prevalencia_esta_iteracion =  min(45000000,np.sum(array)) / 45000000
            #print␣
↪("infectados_en_esta_generacion",infectados_en_esta_generacion,R0,prevalencia,prevalencia_e
            infectados_en_esta_generacion = int(infectados_en_esta_generacion *␣
↪R0 * max(0,( 1 - (prevalencia + prevalencia_esta_iteracion)) ) )
            #infectados_en_esta_generacion = infectados_en_esta_generacion * R0␣
↪* ( 1 - prevalencia)
            array.append(infectados_en_esta_generacion)
        diccionario_prevalencias['prevalencia ' + str("{:.1f}".
↪format(prevalencia)) + ' y R0 ' + str(R0)] = array
    df = pd.DataFrame.from_dict(diccionario_prevalencias,'index')
    df = Get_Header(GENERACIONES,df,FECHA_INICIAL_STR)
    df = df.astype(np.int64)
    return df.T
```

[4]:
```
# Auxiliary functions
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
```

```
    df.index.name = 'Fecha'
    return df
```

[5]:
```
# first execution
GENERACIONES=8
ARRAY_PREVALENCIAS = np.linspace(0,0.70,8)
ARRAY_PREVALENCIAS
df = ␣
 ↪Calcular_Cuadro_Prevalencias(R0=R0,GENERACIONES=GENERACIONES,ARRAY_PREVALENCIAS=ARRAY_PREVA
df
```

[5]:

	prevalencia 0.0 y R0 5.7	prevalencia 0.1 y R0 5.7	\
2020-02-06	5	5	
2020-02-11	28	25	
2020-02-16	159	128	
2020-02-21	906	656	
2020-02-26	5164	3365	
2020-03-02	29430	17260	
2020-03-07	167617	88496	

	prevalencia 0.2 y R0 5.7	prevalencia 0.3 y R0 5.7	\
2020-02-06	4	3	
2020-02-11	18	11	
2020-02-16	82	43	
2020-02-21	373	171	
2020-02-26	1700	682	
2020-03-02	7751	2721	
2020-03-07	35334	10855	

	prevalencia 0.4 y R0 5.7	prevalencia 0.5 y R0 5.7	\
2020-02-06	3	2	
2020-02-11	10	5	
2020-02-16	34	14	
2020-02-21	116	39	
2020-02-26	396	111	
2020-03-02	1354	316	
2020-03-07	4630	900	

	prevalencia 0.6 y R0 5.7	prevalencia 0.7 y R0 5.7
2020-02-06	2	1
2020-02-11	4	1
2020-02-16	9	1
2020-02-21	20	1
2020-02-26	45	1
2020-03-02	102	1
2020-03-07	232	1

[6]:
```
from matplotlib import pyplot as plt
import pandas as pd
import numpy as np

def Get_Chart(df, title="default"):
    fig = plt.figure(figsize=(8, 6), dpi=80)

    for ca in df.columns:
        plt.plot(df[ca])
        plt.legend(df.columns)
        fig.suptitle(title, fontsize=20)
    plt.gca().xaxis.set_major_formatter(mdates.DateFormatter('%b-%d'))
    plt.gca().xaxis.set_major_locator(mdates.DayLocator(interval=7))
    plt.xticks(rotation=45)
    return plt
```

[7]:
```
from IPython.display import display, HTML

ARRAY_R0S = [2.7,
             3.7 ,
             4.7,
             5.7,
             6.7 ]

for R0 in ARRAY_R0S :
    print("Tabla de como varía el nº de infectados, según varía la prevalencia,␣
 ↪con R0 = " + str(R0))
    df = ␣
 ↪Calcular_Cuadro_Prevalencias(R0=R0,GENERACIONES=GENERACIONES,ARRAY_PREVALENCIAS=ARRAY_PREVA
    display(HTML (df.to_html()))
    plt = Get_Chart(df=interpolate_dataframe(df,'D'), title = 'Numero de␣
 ↪infecciones por semana, con R0 = ' + str(R0))
```

```
Tabla de como varía el nº de infectados, según varía la prevalencia, con R0 =
2.7
```

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

```
Tabla de como varía el nº de infectados, según varía la prevalencia, con R0 =
3.7
```

```
/root/anaconda2/envs/jupyter/lib/python3.6/site-
packages/pandas/plotting/_matplotlib/converter.py:103: FutureWarning: Using an
implicitly registered datetime converter for a matplotlib plotting method. The
converter was registered by pandas on import. Future versions of pandas will
require you to explicitly register matplotlib converters.
```

```
To register the converters:
        >>> from pandas.plotting import register_matplotlib_converters
        >>> register_matplotlib_converters()
  warnings.warn(msg, FutureWarning)
```

<IPython.core.display.HTML object>

Tabla de como varía el nº de infectados, según varía la prevalencia, con R0 = 4.7

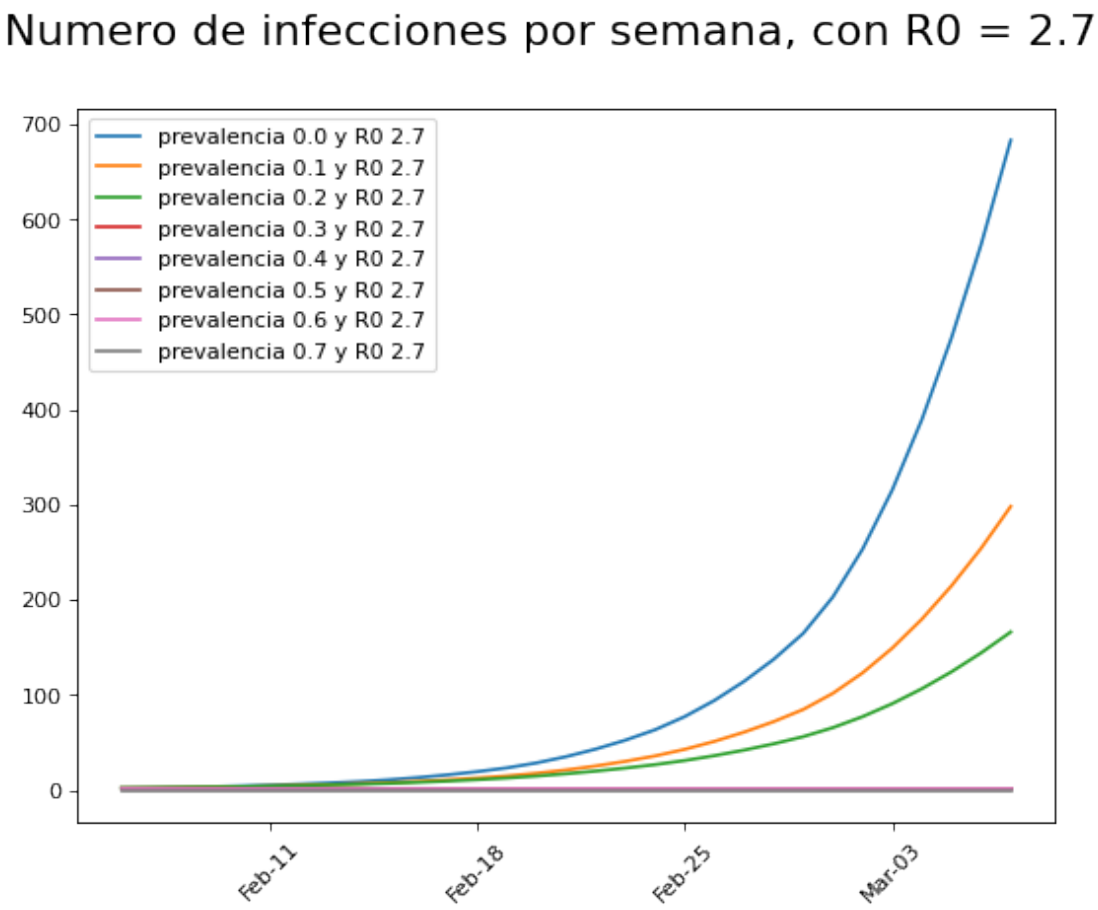
<IPython.core.display.HTML object>

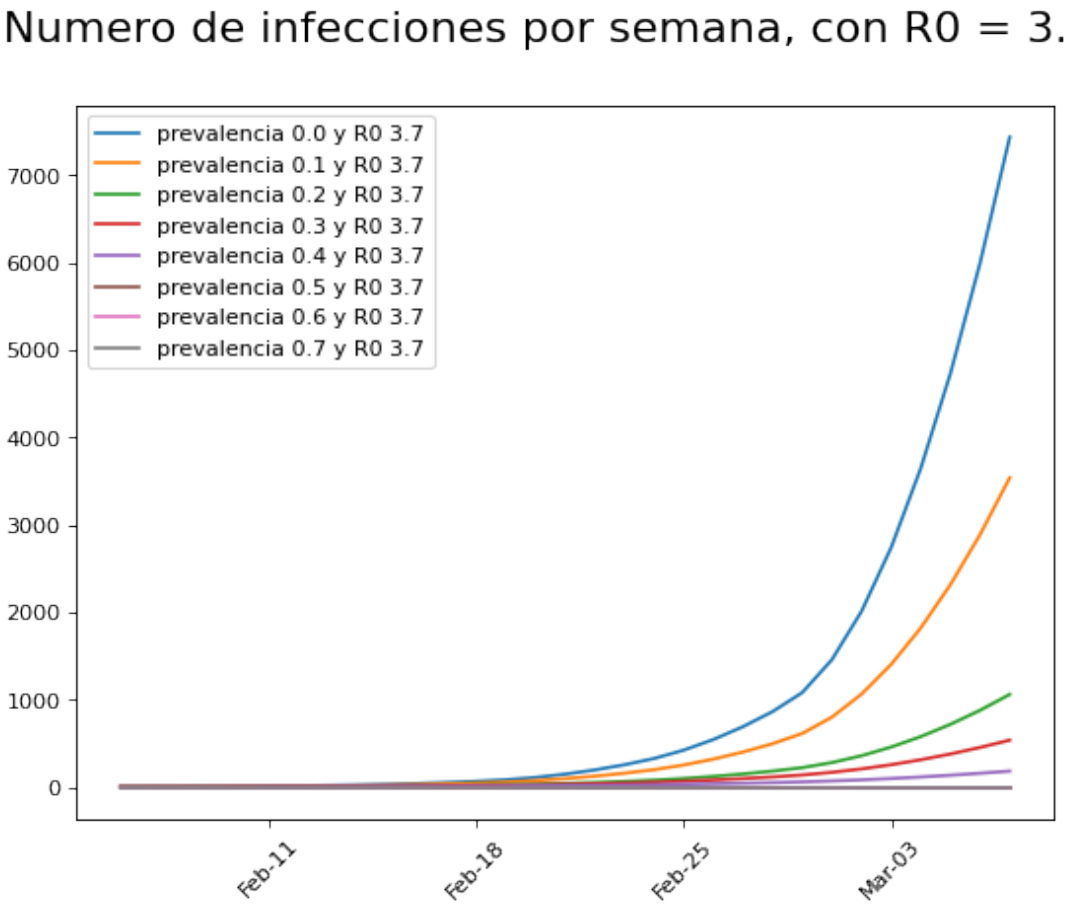
Tabla de como varía el nº de infectados, según varía la prevalencia, con R0 = 5.7

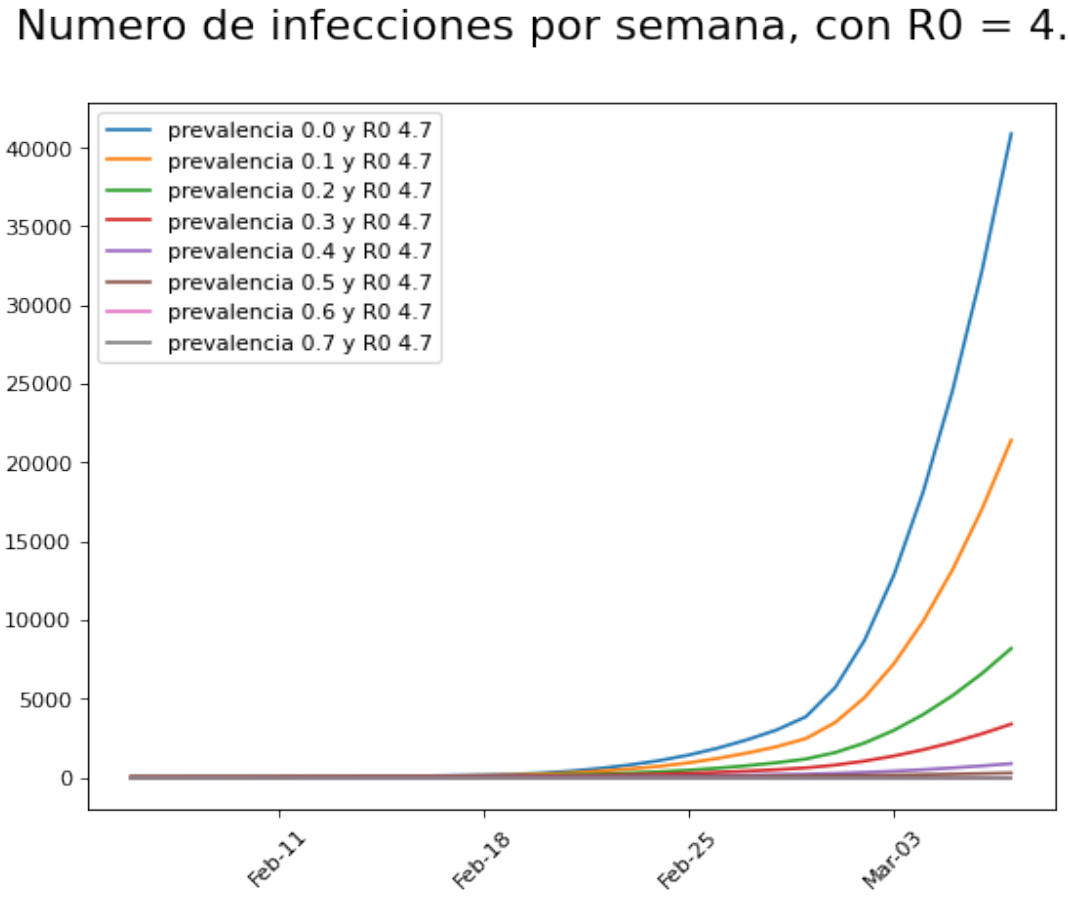
<IPython.core.display.HTML object>

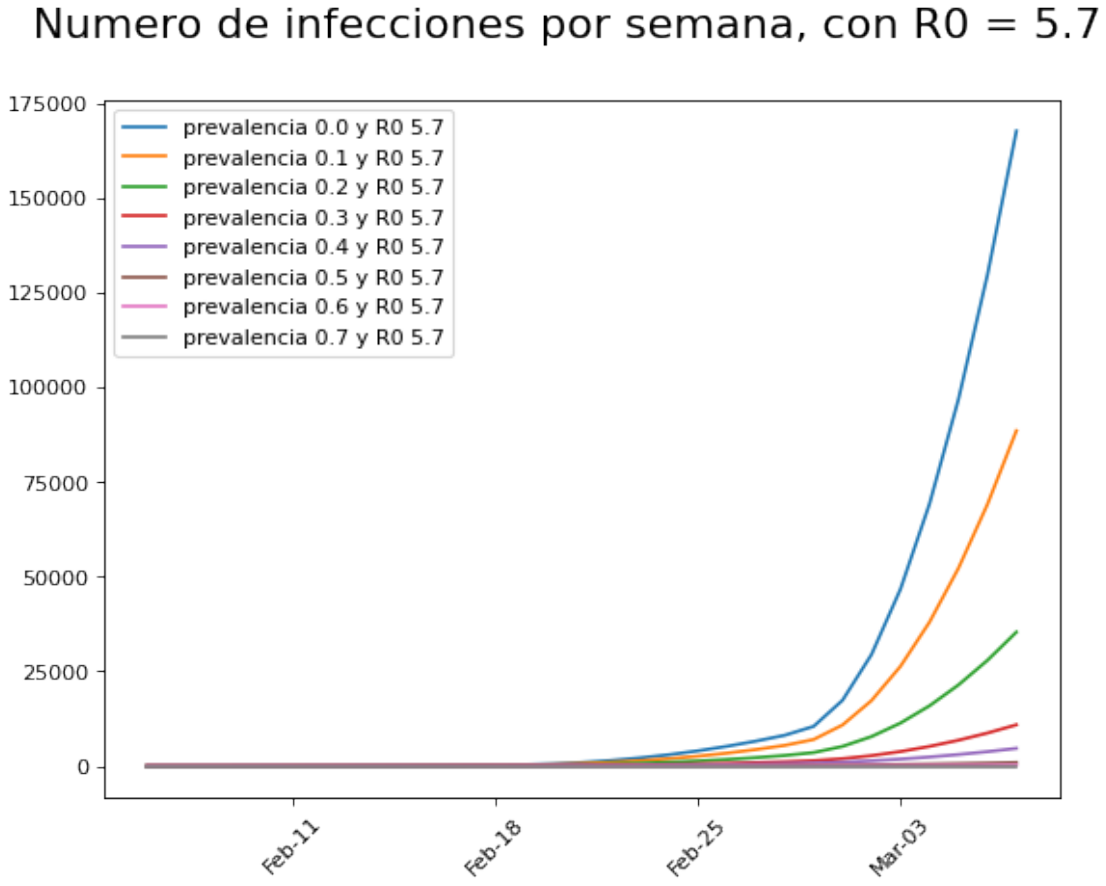
Tabla de como varía el nº de infectados, según varía la prevalencia, con R0 = 6.7

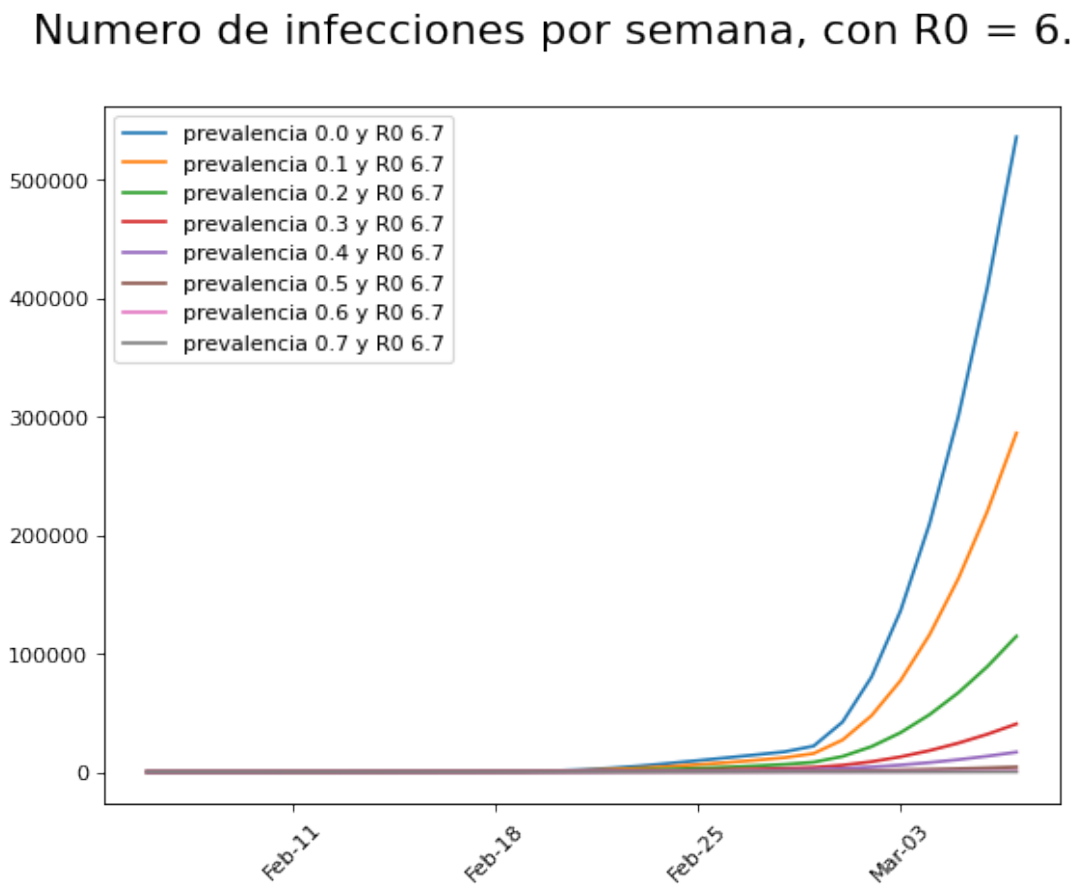
<IPython.core.display.HTML object>




Numero de infecciones por semana, con R0 = 3.7
prevalencia 0.0 y R0 3.7
prevalencia 0.1 y R0 3.7
prevalencia 0.2 y R0 3.7
prevalencia 0.3 y R0 3.7
prevalencia 0.4 y R0 3.7
prevalencia 0.5 y R0 3.7
prevalencia 0.6 y R0 3.7
prevalencia 0.7 y R0 3.7
7000
6000
5000
4000
3000
2000
1000
0
Feb-11
Feb-18
Feb-25
Mar-03


Numero de infecciones por semana, con R0 = 4.7
prevalencia 0.0 y R0 4.7
prevalencia 0.1 y R0 4.7
prevalencia 0.2 y R0 4.7
prevalencia 0.3 y R0 4.7
prevalencia 0.4 y R0 4.7
prevalencia 0.5 y R0 4.7
prevalencia 0.6 y R0 4.7
prevalencia 0.7 y R0 4.7
40000
35000
30000
25000
20000
15000
10000
5000
0
Feb-11
Feb-18
Feb-25
Mar-03


Numero de infecciones por semana, con R0 = 5.7
prevalencia 0.0 y R0 5.7
prevalencia 0.1 y R0 5.7
prevalencia 0.2 y R0 5.7
prevalencia 0.3 y R0 5.7
prevalencia 0.4 y R0 5.7
prevalencia 0.5 y R0 5.7
prevalencia 0.6 y R0 5.7
prevalencia 0.7 y R0 5.7
175000
150000
125000
100000
75000
50000
25000
0
Feb-11
Feb-18
Feb-25
Mar-03



TODO: - Hacer gráfico de la evolución del nº de infectados, en el confinamiento (R0<1)

2 Conclusiones:

- Para un R0 estimado de 5.7 del COVID-19, hace falta un 70% de prevalencia, para que no haya brotes masivos.
- Las medidas de distanciamiento social, influyen para mantener bajo el R0, el cual influye mucho en el número de infectados.
- Con prevalencia bajas, aunque no haya inmunidad de grupo, el nº de infectados desciende significativamente. (En España, que los estudios sugieren una prevalencia menor del 10%, podría haber 1/3 ó 1/2 muertos)
- Además de la prevalencia, tambien es importante, tener controlado ,el nº de personas infectadas en cada momento:
- Hay que detectar los brotes lo mas pronto posible, y reducir el R0 mediante la búsqueda de contactos del infectado cuando el brote es aún pequeño.
- En caso contrario, si no se puede controlar el brote, como el nº de infectados se disparará, se pueden realizar confinamientos parciales intermitentes, para reducir el R0 por debajo de 1,

hasta que el nº de infectados baje.

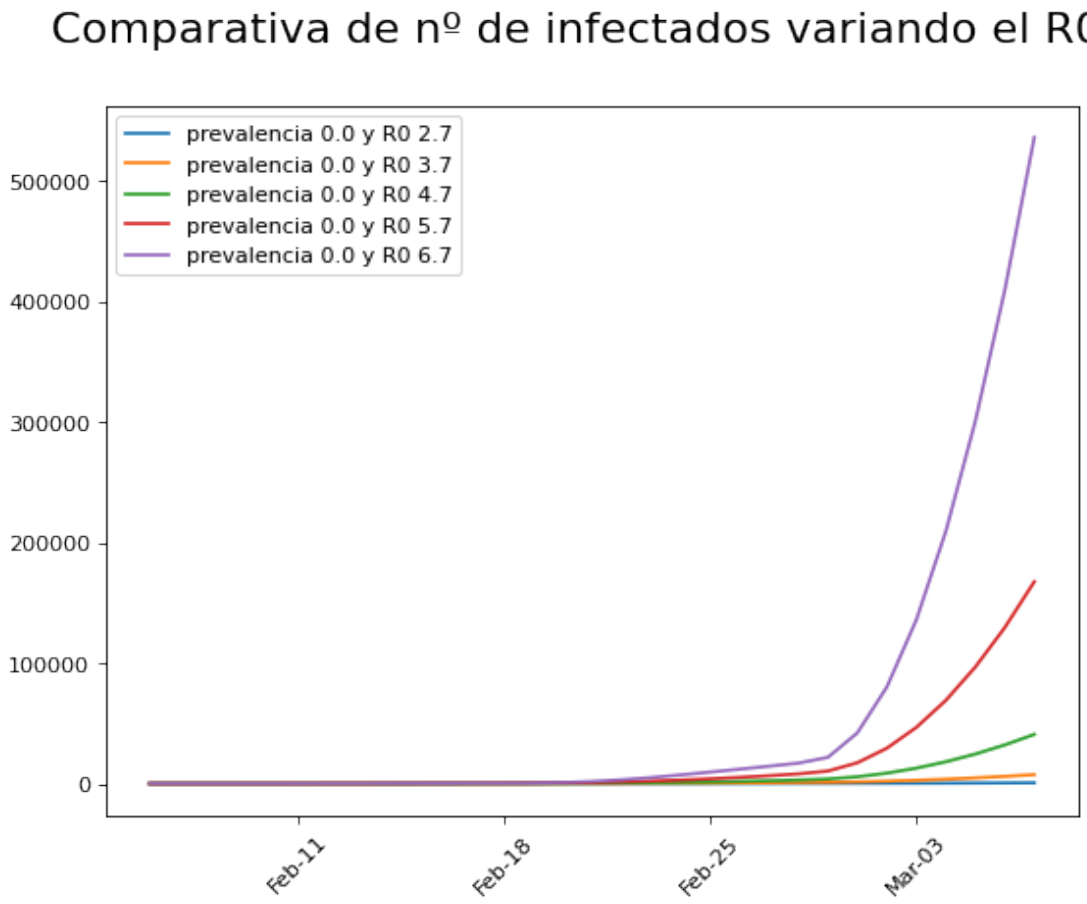
2.1 Bonus : ¿ Sirve de algo quedarse en casa ?

[8]:
```
ARRAY_R0S = [2.7,
             3.7 ,
             4.7,
             5.7,
             6.7 ]

df_R0s = pd.DataFrame()

for MI_R0 in ARRAY_R0S :
    df = ␣
 ↪Calcular_Cuadro_Prevalencias(R0=MI_R0,GENERACIONES=GENERACIONES,ARRAY_PREVALENCIAS=ARRAY_PR
    df_R0s[df.columns[0]] = df[df.columns[0]]


print("Tabla de como varía el nº de infectados, según varía el R0 " )


display(HTML (df_R0s.to_html()))
print("Total de infectados en cada escenario : " )
print( df_R0s.astype(np.int64).sum(axis=0) )


plt = Get_Chart(interpolate_dataframe(df_R0s,'D') ,title= 'Comparativa de nº de␣
 ↪infectados variando el R0')
```

```
Tabla de como varía el nº de infectados, según varía el R0
```

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

```
Total de infectados en cada escenario :
prevalencia 0.0 y R0 2.7      1085
prevalencia 0.0 y R0 3.7     10185
prevalencia 0.0 y R0 4.7     51925
prevalencia 0.0 y R0 5.7    203309
prevalencia 0.0 y R0 6.7    630755
dtype: int64
```



Como se ve en la gráfica, bajar el R0, es muy importante, para detener el nº de infectados, y por ende en número de los fallecidos.

2.2 Re- Bonus : Se me ha hecho larguísimo, ¿ Podría haber estado menos tiempo en casa ?

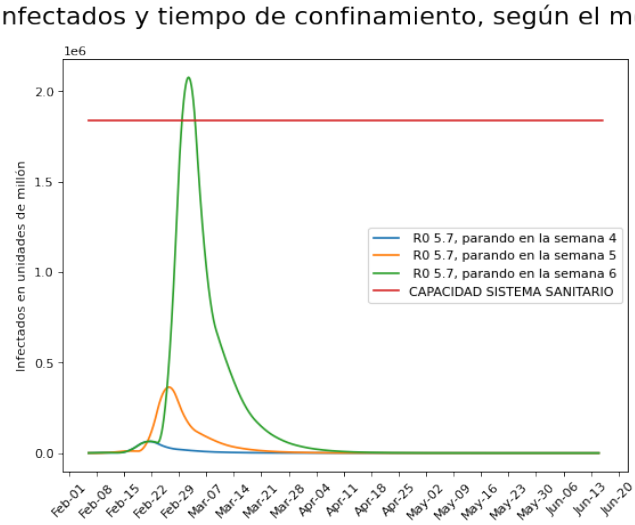
[9]:
```
def comparacion_semanas(SITUACION_INICIAL=1):
    GENERACIONES=8
    prevalencia=0
    diccionario_prevalencias={}
    R0=5.7
    i=1
    for NUM_GENERACIONES in range(5,8) :
        infectados_en_esta_generacion=SITUACION_INICIAL
        array = []
        for generacion in  range(1,NUM_GENERACIONES):
            infectados_en_esta_generacion = infectados_en_esta_generacion * R0␣
↪* (1-prevalencia)
            array.append(infectados_en_esta_generacion)
```

```
        valor_actual = array[-1]
        # Calculamos el R0 con el confinamiento
        # En alemanio confinada el R0 estimado es un 0.7. Nosotros debería ser␣
 ↪mas bajo
        NUEVO_R0 = 0.5
        while infectados_en_esta_generacion > 1 :
            infectados_en_esta_generacion = infectados_en_esta_generacion *␣
 ↪NUEVO_R0 * (1-prevalencia)
            array.append(infectados_en_esta_generacion)
        diccionario_prevalencias[' R0 ' + str(R0) + ', parando en la semana ' +␣
 ↪str(generacion)] = array
        i=1


    df = pd.DataFrame.from_dict(diccionario_prevalencias,'index')
    df = Get_Header(df.shape[1]+1,df)
    df = df.T
    df.index = pd.to_datetime(df.index)

    return df
```

[10]:
```
df = comparacion_semanas(SITUACION_INICIAL=60)
df= df.fillna(0)

print("Total de infectados en cada escenario : " )
print( df.astype(np.int64).sum(axis=0) )
df_interpolate = interpolate_dataframe(df=df,freq='H')
df_interpolate['CAPACIDAD SISTEMA SANITARIO' ] = CAPACIDAD_SISTEMA_SANITARIO

plt = Get_Chart(df = df_interpolate,title= 'Comparativa de infectados y tiempo␣
 ↪de confinamiento, según el momento de empezar')
plt.ylabel('Infectados en unidades de millón', size = 10)



print("Tabla de como varía el nº de infectados, según varía la semana de inicio␣
 ↪del confinamiento " )
df.style.format ({ c : "{:20,.0f}" for c in df.columns }).
 ↪background_gradient(cmap='Wistia',  )
```

```
Total de infectados en cada escenario :
 R0 5.7, parando en la semana 4     140064
 R0 5.7, parando en la semana 5     798759
 R0 5.7, parando en la semana 6    4553312
dtype: int64
Tabla de como varía el nº de infectados, según varía la semana de inicio del
confinamiento
```

[10]: <pandas.io.formats.style.Styler at 0x7f743aeba208>



Como se ve en del anterior gráfico, el tiempo que dura confinamiento, y el nº de infectados varía enormemente.

Sin duda la demora en adoptar las restricciones - como se ve con esta gráfica de datos teóricos- ha influido en el tiempo de confinamiento, y mucho peor, ha costado un gran número de vidas.

2.3 ¿ Se acabará la pandemia en verano ?

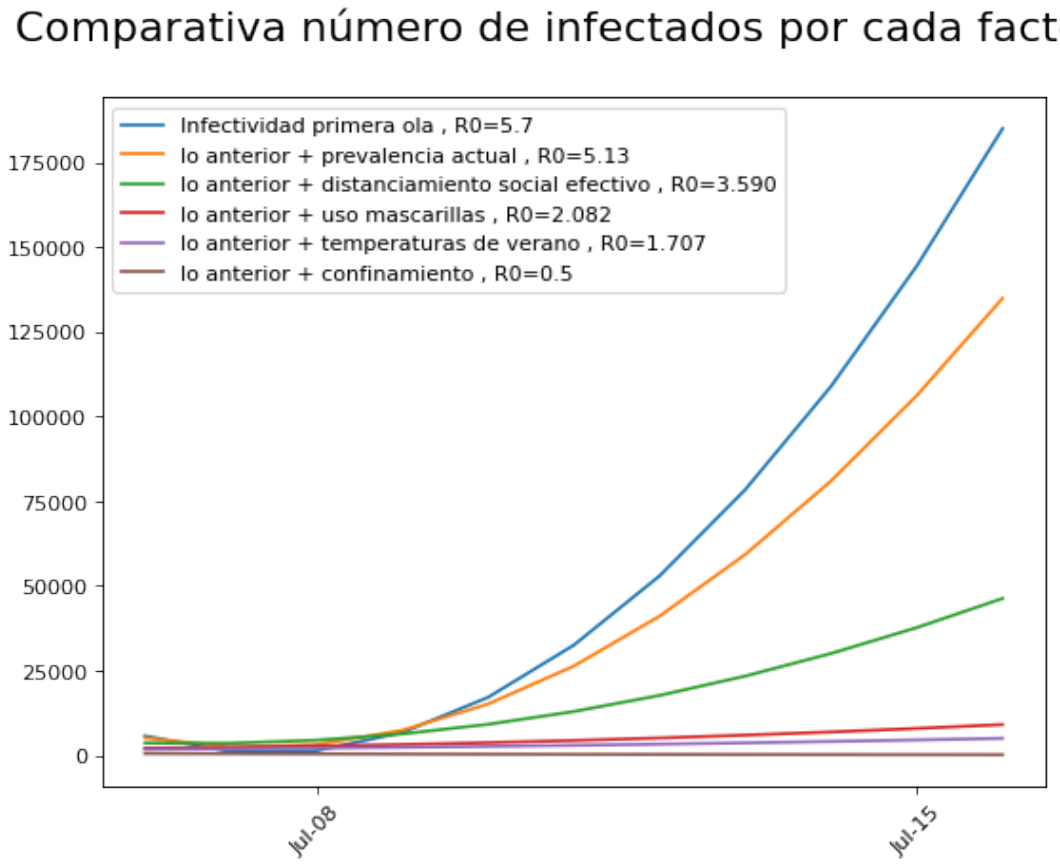
[11]:
```
import numpy as np

### Aqui
INFECTIVIDAD_POR_DEFECTO = 5.7

ARRAY_ROS = [INFECTIVIDAD_POR_DEFECTO]
infectividad = INFECTIVIDAD_POR_DEFECTO
ARRAY_ROS_TITULOS = ["Infectividad primera ola"]



prevalencia = 0.1
infectividad *= 1 - prevalencia
ARRAY_ROS.append(infectividad)
ARRAY_ROS_TITULOS.append("lo anterior + prevalencia actual")

""" We simulated social distancing by reducing R0
by a fixed proportion, which ranged between 0 and 60%, on par with the␣
 ↪reduction in R0
achieved in China through intense social distancing measures
```

```
"""
distanciamiento_social_efectivo = 0.3
infectividad *= 1 - distanciamiento_social_efectivo
ARRAY_R0S.append(infectividad)
ARRAY_R0S_TITULOS.append("lo anterior + distanciamiento social efectivo")
"""
  Uso de mascarillas
  The team investigated the varying effectiveness of facemasks. Previous␣
 ↪research shows that even homemade masks made from cotton t-shirts or␣
 ↪dishcloths can prove 90% effective at preventing transmission.
  The study suggests that an entire population wearing masks of just 75%␣
 ↪effectiveness can bring a very high 'R' number of 4.0-the UK was close to␣
 ↪this before lockdown-all the way down to under 1.0, even without aid of␣
 ↪lockdowns.
  https://medicalxpress.com/news/2020-06-widespread-facemask-covid-.html
"""
porcentaje_efectividad_mascarillas = 0.7
porcentaje_poblacion_usa_mascarillas = 0.6
infectividad *= 1 - (porcentaje_efectividad_mascarillas *␣
 ↪porcentaje_poblacion_usa_mascarillas)
ARRAY_R0S.append(infectividad)
ARRAY_R0S_TITULOS.append("lo anterior + uso mascarillas")

incremento_grados_temperatura_media = 15
infectividad_verano = 1 - ( incremento_grados_temperatura_media * 0.012 )
infectividad *= infectividad_verano
ARRAY_R0S.append(infectividad)
ARRAY_R0S_TITULOS.append("lo anterior + temperaturas de verano")

"""
  Confinamiento :
  One study from France estimated that timely lockdowns pushed R0 down to 0.5␣
 ↪from 3.3
  https://hal-pasteur.archives-ouvertes.fr/pasteur-02548181/document

"""
#efectividad_confinamiento=0.88
#infectividad_confinamiento = 1 - ( efectividad_confinamiento )
#infectividad *= infectividad_confinamiento
infectividad = 0.5
ARRAY_R0S.append(infectividad)
ARRAY_R0S_TITULOS.append("lo anterior + confinamiento")

ARRAY_TITULOS = []
for i,element in enumerate(ARRAY_R0S):
    titulo = str(ARRAY_R0S_TITULOS[i]) + " , R0=" + str(ARRAY_R0S[i])[0:5]
```

```
    ARRAY_TITULOS.append(titulo)


df_ROs = pd.DataFrame()

GENERACIONES=4
SITUACION_INICIAL=1000
FECHA_INICIAL_STR = '2020-07-01'
for MI_R0 in ARRAY_ROS :
    df =  Calcular_Cuadro_Prevalencias(
        SITUACION_INICIAL  = SITUACION_INICIAL,
        R0                 = MI_R0,
        GENERACIONES       = GENERACIONES,
        ARRAY_PREVALENCIAS = ARRAY_PREVALENCIAS,
        FECHA_INICIAL_STR  = FECHA_INICIAL_STR
    )
    df_ROs[df.columns[0]]  = df[df.columns[0]]

df_ROs.columns = ARRAY_TITULOS

print("Total de infectados en cada escenario : " )



print("Ejemplo de si salieramos todos del estado de alarma " )

print( df_ROs.astype(np.int64).sum(axis=0) )

plt = Get_Chart(interpolate_dataframe(df_ROs,'D') ,title= 'Comparativa número␣
 ↪de infectados por cada factor')

df_ROs.style.format ({ c : "{:20,.0f}" for c in df_ROs.columns }).
 ↪background_gradient(cmap='Wistia',  )
```

```
Total de infectados en cada escenario :
Ejemplo de si salieramos todos del estado de alarma
Infectividad primera ola , R0=5.7                              223192
lo anterior + prevalencia actual , R0=5.13                     166334
lo anterior + distanciamiento social efectivo , R0=3.590        62751
lo anterior + uso mascarillas , R0=2.082                        15447
lo anterior + temperaturas de verano , R0=1.707                  9599
lo anterior + confinamiento , R0=0.5                              873
dtype: int64
```

[11]: <pandas.io.formats.style.Styler at 0x7f743aed9b70>



De los datos anteriores parece que la epidemia no se va a poder contener durante el verano, salvo que alguno de los factores esté minusvalorado :

- Que en la nueva normalidad este tan atemorizada que haya un distanciamento social muy efectivo,
- Que use mascarillas la práctica totalidad de la población, o
- Que el incremento de la radiación solar tenga mas incidencia que la estimada.

No obstante, aunque no se acabará en verano, podemos intentar estimar si habrá una segunda ola que necesite confinamiento en verano.

2.4 ¿ Cuándo llegará la segunda ola ?

Para estimar si el sistema sanitario se verá sobrepasado otra vez, intentamos calcular la capacidad del sistema sanitario. Sabieno el nº de camas UCIs, el porcentaje de hospitalizados que necesita una cama UCI, y el porcentaje de enfermos que necesitan hospitalización, estimamos la capacidad del sistema sanitario.

[12]:
```
GENERACIONES=12
SITUACION_INICIAL=1000
```

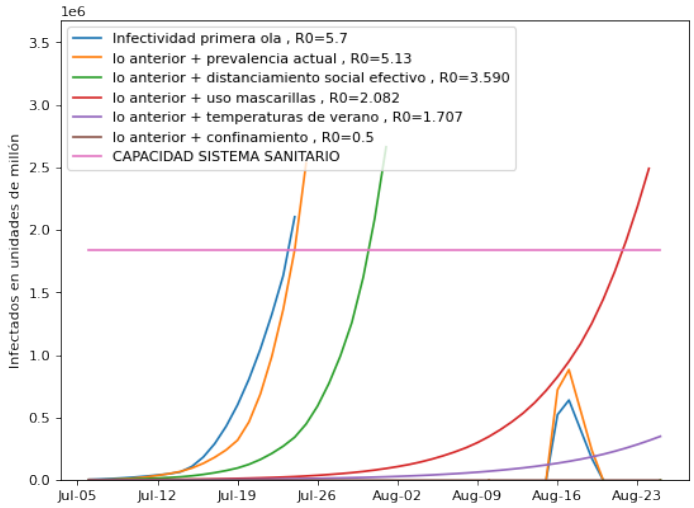
```
FECHA_INICIAL_STR = '2020-07-01'
df_ROs = pd.DataFrame()

for MI_RO in ARRAY_ROS :
    df =  Calcular_Cuadro_Prevalencias(
        SITUACION_INICIAL  = SITUACION_INICIAL,
        RO                 = MI_RO,
        GENERACIONES       = GENERACIONES,
        ARRAY_PREVALENCIAS = ARRAY_PREVALENCIAS,
        FECHA_INICIAL_STR  = FECHA_INICIAL_STR
    )
    df_ROs[df.columns[0]]  = df[df.columns[0]]
df_ROs.columns = ARRAY_TITULOS

df_master = df_ROs.copy()
df_ROs    = interpolate_dataframe(df_ROs,'D')
df_ROs    = df_ROs[df_ROs < (CAPACIDAD_SISTEMA_SANITARIO * 1.5) ]
df_ROs['CAPACIDAD SISTEMA SANITARIO' ] = CAPACIDAD_SISTEMA_SANITARIO

#plt = Get_Chart(df_ROs ,title= 'Comparativa número de infectados por cada␣
 ↪factor')
title = 'Cuando se tarda en superar la capacidad del sistema sanitario, por␣
 ↪escenario.'
df    = df_ROs
fig   = plt.figure(figsize=(8, 6), dpi=80)
ax    = plt.gca()

ax.set_ylim([0,CAPACIDAD_SISTEMA_SANITARIO*2])

for ca in df.columns:
    plt.plot(df[ca])
    plt.legend(df.columns)
    fig.suptitle(title, fontsize=20)
#return plt
ax.legend(df.columns, loc='upper left')
plt.ylabel('Infectados en unidades de millón', size = 10)
plt.gca().xaxis.set_major_formatter(mdates.DateFormatter('%b-%d'))
plt.gca().xaxis.set_major_locator(mdates.DayLocator(interval=7))
plt

print("Estimación de cuando se supera la capacidad del sistema sanitario, en␣
 ↪cada escenario")
print("Presuponiendo una situación inicial de 1.000 infectados")

df_master
df_master = df_master[df_master < (CAPACIDAD_SISTEMA_SANITARIO * 2) ]
```

```
df_master.style.format ({ c : "{:20,.0f}" for c in df_master.columns }).
 ↪background_gradient(cmap='Wistia',  )
```

```
Estimación de cuando se supera la capacidad del sistema sanitario, en cada
escenario
Presuponiendo una situación inicial de 1.000 infectados
```

[12]: <pandas.io.formats.style.Styler at 0x7f743b4e15f8>

Cuando se tarda en superar la capacidad del sistema sanitario, por escenario.



2.4.1 Conclusiones

Estas son estimaciones, no datos reales, pero las tendencias son :

- Parece que las temperaturas de verano pueden ralentizar la infección lo bastante como para que no tengamos otra ola hasta despues del verano.
- Tras el verano, futuras olas parecen inevitables en el plazo de entre un mes y dos de la finalización de las altas temperaturas.
- Habrá que adelantarse, con periodos de confinamientos intermitentes. La cantidad y duración de estos confinamientos aún deben estimarse.
- Futuros cambios de infectividad, o nuevos tratamientos podrían variar este escenario.

2.4.2 Vamos a generar nuevas olas epidemicas, en cada escenario.

Las olas progresaran hasta que alcancen la capacidad del sistema sanitario, en cuyo caso habrá un confinamiento

TO-DO : Estimar mejor cuando se ocupan las camas uci (la duración de una estancia uci está entre 2-11 semanas)

TO-DO : Ajustar mejor los parámetros de mortalidad con saturación del sistema.

[13]:
```
#### PRedicciones a futuro

"""
Crear 30.
Borrar las mayores de umbral superios.
Aplicar hasta que haya menos de x
"""
from datetime import date
RO_CALOR= 1.702
GENERACIONES=9

def calcular_prevision(
    FECHA_FINAL_STR,
    FECHA_INICIAL_STR,
    SITUACION_INICIAL,
    POBLACION_INICIAL_INFECTADA,
    RO_max,
    RO_min,
    Umbral_max,
    Umbral_min):

    print(  FECHA_FINAL_STR,
    FECHA_INICIAL_STR,
    SITUACION_INICIAL,
    POBLACION_INICIAL_INFECTADA,
    RO_max,
    RO_min,
    Umbral_max,
    Umbral_min)
    df_temp = pd.DataFrame()
    df = pd.DataFrame(columns = ['Infectados'])
    #
    while FECHA_INICIAL_STR < FECHA_FINAL_STR :

        df_temp = pd.DataFrame()
        PREVALENCIA = (POBLACION_INICIAL_INFECTADA + df.iloc[:,0].sum()) /
↪45000000
        ARRAY_PREVALENCIAS = []
        ARRAY_PREVALENCIAS.append(PREVALENCIA)

        # Subida
        PERIODO_CALOR = ( FECHA_INICIAL_STR[5:] > '06-15' ) & (␣
↪FECHA_INICIAL_STR[5:] <  '09-15' )
        RO_DESCONTADO_CALOR = RO_CALOR if PERIODO_CALOR else RO_max
        print(f"""SITUACION_INICIAL={SITUACION_INICIAL},
```

```
            RO                  = {RO_DESCONTADO_CALOR} ,
            GENERACIONES        = {GENERACIONES}        ,
            ARRAY_PREVALENCIAS = {ARRAY_PREVALENCIAS}  ,
            FECHA_INICIAL_STR  = {FECHA_INICIAL_STR}""")

        df_temp =  Calcular_Cuadro_Prevalencias( SITUACION_INICIAL  =␣
↪SITUACION_INICIAL   ,
            RO                  = RO_DESCONTADO_CALOR ,
            GENERACIONES        = GENERACIONES        ,
            ARRAY_PREVALENCIAS = ARRAY_PREVALENCIAS  ,
            FECHA_INICIAL_STR  = FECHA_INICIAL_STR )

        df_temp['Infectados'] = df_temp.iloc[:,0]
        df_temp = df_temp[(df_temp['Infectados'] < Umbral_max )]
        df_temp = df_temp[(df_temp['Infectados'] != 0 )]

        df_temp.dropna()
        df_temp = df_temp.loc[~df_temp.index.duplicated(keep='last')]
        df_temp = df_temp['Infectados']


        df_temp = pd.DataFrame(df_temp)
        df = pd.concat([df_temp,df])
        df = df.sort_index()

        # BAJADA TODO

        # Bajada
        PREVALENCIA = (POBLACION_INICIAL_INFECTADA + df.iloc[:,0].sum()) /
↪45000000

        ARRAY_PREVALENCIAS = []
        ARRAY_PREVALENCIAS.append(PREVALENCIA)

        SITUACION_INICIAL = df.iloc[-1]['Infectados']
        FECHA_INICIAL_STR = df.index[-1]
        df_temp = pd.DataFrame()

        df_temp =  Calcular_Cuadro_Prevalencias(
            SITUACION_INICIAL=SITUACION_INICIAL,
            RO=RO_min,
            GENERACIONES=40,
            ARRAY_PREVALENCIAS=ARRAY_PREVALENCIAS,
            FECHA_INICIAL_STR  = FECHA_INICIAL_STR )

        df_temp['Infectados'] = df_temp.iloc[:,0]
        df_temp = df_temp[(df_temp['Infectados'] > Umbral_min)]
```

```
        df_temp = df_temp[(df_temp['Infectados'] != 0 )]
        df_temp.dropna()
        df_temp = df_temp.loc[~df_temp.index.duplicated(keep='last')]
        df_temp = df_temp['Infectados']


        df_temp = pd.DataFrame(df_temp)
        df = pd.concat([df_temp,df])
        df = df.sort_index()

        SITUACION_INICIAL = df.iloc[-1]['Infectados']
        FECHA_INICIAL_STR = df.index[-1]

    df = df.dropna()
    df = df.loc[~df.index.duplicated(keep='last')]

    return df

SITUACION_INICIAL           = 1000
POBLACION_INICIAL_INFECTADA = 4500000
R0_max                      = 5.7
R0_min                      = 0.5
Umbral_max                  = CAPACIDAD_SISTEMA_SANITARIO
Umbral_min                  = 10000
FECHA_INICIAL_STR           = '2020-07-01'
FECHA_FINAL_STR             = '2021-01-01'



df = calcular_prevision(
    FECHA_FINAL_STR,
    FECHA_INICIAL_STR,
    SITUACION_INICIAL,
    POBLACION_INICIAL_INFECTADA,
    R0_max,
    R0_min,
    Umbral_max,
    Umbral_min)
df
```

```
2021-01-01 2020-07-01 1000 4500000 5.7 0.5 1839090 10000
SITUACION_INICIAL=1000,
            R0                = 1.702 ,
            GENERACIONES      = 9       ,
            ARRAY_PREVALENCIAS = [0.1]  ,
            FECHA_INICIAL_STR  = 2020-07-01
```

```
SITUACION_INICIAL=13552,
            R0                = 1.702 ,
            GENERACIONES      = 9       ,
            ARRAY_PREVALENCIAS = [0.1021718888888889]  ,
            FECHA_INICIAL_STR  = 2020-08-15
SITUACION_INICIAL=13769,
            R0                = 5.7 ,
            GENERACIONES      = 9       ,
            ARRAY_PREVALENCIAS = [0.13294224444444444]  ,
            FECHA_INICIAL_STR  = 2020-10-14
SITUACION_INICIAL=17321,
            R0                = 5.7 ,
            GENERACIONES      = 9       ,
            ARRAY_PREVALENCIAS = [0.2032117111111111]  ,
            FECHA_INICIAL_STR  = 2020-11-23
```

[13]:

	Infectados
2020-07-06	1531
2020-07-11	2345
2020-07-16	3591
2020-07-21	5499
2020-07-26	8420
2020-07-31	12890
2020-08-05	19728
2020-08-10	30179
2020-08-15	13552
2020-08-20	20708
2020-08-25	31627
2020-08-30	48266
2020-09-04	73571
2020-09-09	111939
2020-09-14	169843
2020-09-19	256608
2020-09-24	385207
2020-09-29	168226
2020-10-04	73152
2020-10-09	31750
2020-10-14	13769
2020-10-19	68049
2020-10-24	335727
2020-10-29	1642069
2020-11-03	674557
2020-11-08	272050
2020-11-13	108896
2020-11-18	43457
2020-11-23	17321
2020-11-28	78666

```
2020-12-03     356492
2020-12-08    1599427
2020-12-13     601044
2020-12-18     221850
2020-12-23      81340
2020-12-28      29749
2021-01-02      10870
```

[14]:
```
""" diccionario_R0s = {"Infectividad con prevalencia original, R0=5.7" : { R0 :
 ↪5.7, POBLACION_INICIAL_INFECTADA : 4500000} ,
"lo anterior + distanciamiento social efectivo , R0=3.590"  : 3.590,
"lo anterior + uso mascarillas , R0=2.082" : 2.082   ,
"lo anterior + temperaturas de verano , R0=1.707" : 1.702
}

Infectividad primera ola , R0=5.7                          223383
lo anterior + distanciamiento social efectivo , R0=3.989    83430
lo anterior + uso mascarillas , R0=2.314                    20062
lo anterior + temperaturas de verano , R0=1.897             12331
lo anterior + confinamiento , R0=0.5                          875
"""
array_parametros = [
    { "descripcion" : "Infectividad con prevalencia original, R0=5.7"          
 ↪  , 'R0' : 5.7   , 'POBLACION_INICIAL_INFECTADA' : 0} ,
 ]
array_parametros = [
    { "descripcion" : "Infectividad con prevalencia original, R0=5.7"          
 ↪  , 'R0' : 5.7   , 'POBLACION_INICIAL_INFECTADA' : 0} ,
    { "descripcion" : "Infectividad con prevalencia actual  , R0=5.13"         
 ↪  , 'R0' : 5.7   , 'POBLACION_INICIAL_INFECTADA' : 4500000} ,
    { "descripcion" : "lo anterior + distanciamiento social efectivo  , R0=3.
 ↪590" , 'R0' : 3.989 , 'POBLACION_INICIAL_INFECTADA' : 4500000} ,
    { "descripcion" : "lo anterior + uso mascarillas , R0=2.082"               
 ↪  , 'R0' : 2.314 , 'POBLACION_INICIAL_INFECTADA' : 4500000} ,
]


df_array = []

dict_default_values = {
    "SITUACION_INICIAL" : 10000                       ,
    "R0_min"            :     0.5                     ,
    "Umbral_max"        :  CAPACIDAD_SISTEMA_SANITARIO ,
    "Umbral_min"        :  5000                       ,
    "FECHA_INICIAL_STR" : '2020-07-01'                ,
    "FECHA_FINAL_STR"   : '2021-07-01'
```

```
}



for dict_escenario in array_parametros:
    ## Juntamos los valores por defecto, y los que cambian cada vez.
    param = {**dict_escenario, ** dict_default_values}
    #print(param)

    df_temp = pd.DataFrame()
    df_temp = calcular_prevision(
        FECHA_FINAL_STR             = param['FECHA_FINAL_STR'            ],
        FECHA_INICIAL_STR           = param['FECHA_INICIAL_STR'          ],
        SITUACION_INICIAL           = param['SITUACION_INICIAL'          ],
        POBLACION_INICIAL_INFECTADA = param['POBLACION_INICIAL_INFECTADA'],
        R0_max                      = param['R0'                         ],
        R0_min                      = param['R0_min'                     ],
        Umbral_max                  = param['Umbral_max'                 ],
        Umbral_min                  = param['Umbral_min'                 ]
    )
    df_temp = df_temp.astype(np.int64)
    df_temp = df_temp.loc[~df_temp.index.duplicated(keep='last')]
    suma = int(df_temp.sum(axis=0)/1000000)
    #print(param['descripcion'], df_temp.tail(1).index[-1],"suma: " , suma)
    df_array.append(df_temp)
    DIAS_CONFINAMIENTO = df_temp.shape[0] - (df_temp['Infectados'] -␣
 ↪df_temp['Infectados'].shift(1) > 0).sum()
    plt = Get_Chart(df=interpolate_dataframe(df_temp,'D'),
                    title = param['descripcion'] +" \n, "+ str(suma) + "␣
 ↪millones infectados, " +  str(DIAS_CONFINAMIENTO) + ", dias de confinamiento.
 ↪" )
    plt.gca().xaxis.set_major_locator(mdates.DayLocator(interval=14))

    ax = plt.gca()
    ax.set_ylim([0,Umbral_max])
    param = {}

df  =  pd.concat(df_array)
```

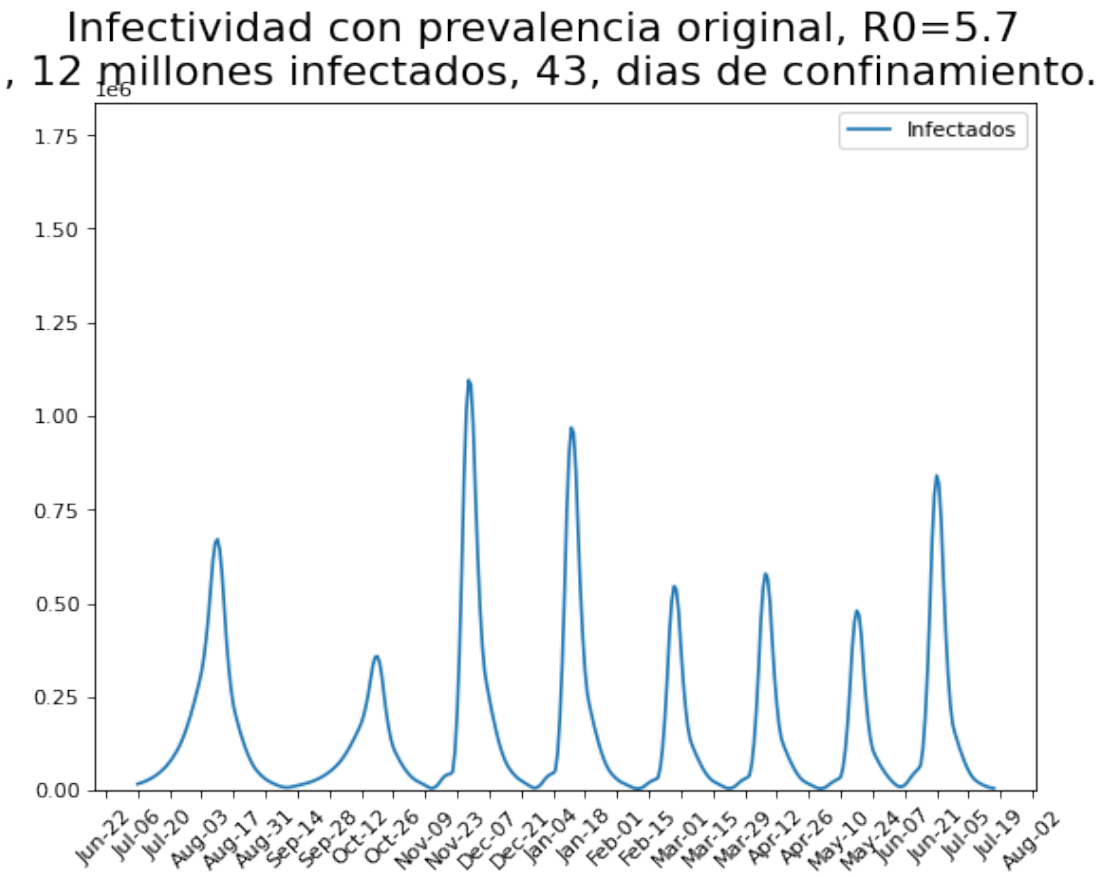
```
2021-07-01 2020-07-01 10000 0 5.7 0.5 1839090 5000
SITUACION_INICIAL=10000,
            R0                = 1.702 ,
            GENERACIONES      = 9       ,
            ARRAY_PREVALENCIAS = [0.0]  ,
            FECHA_INICIAL_STR  = 2020-07-01
SITUACION_INICIAL=7912,
            R0                = 1.702 ,
```

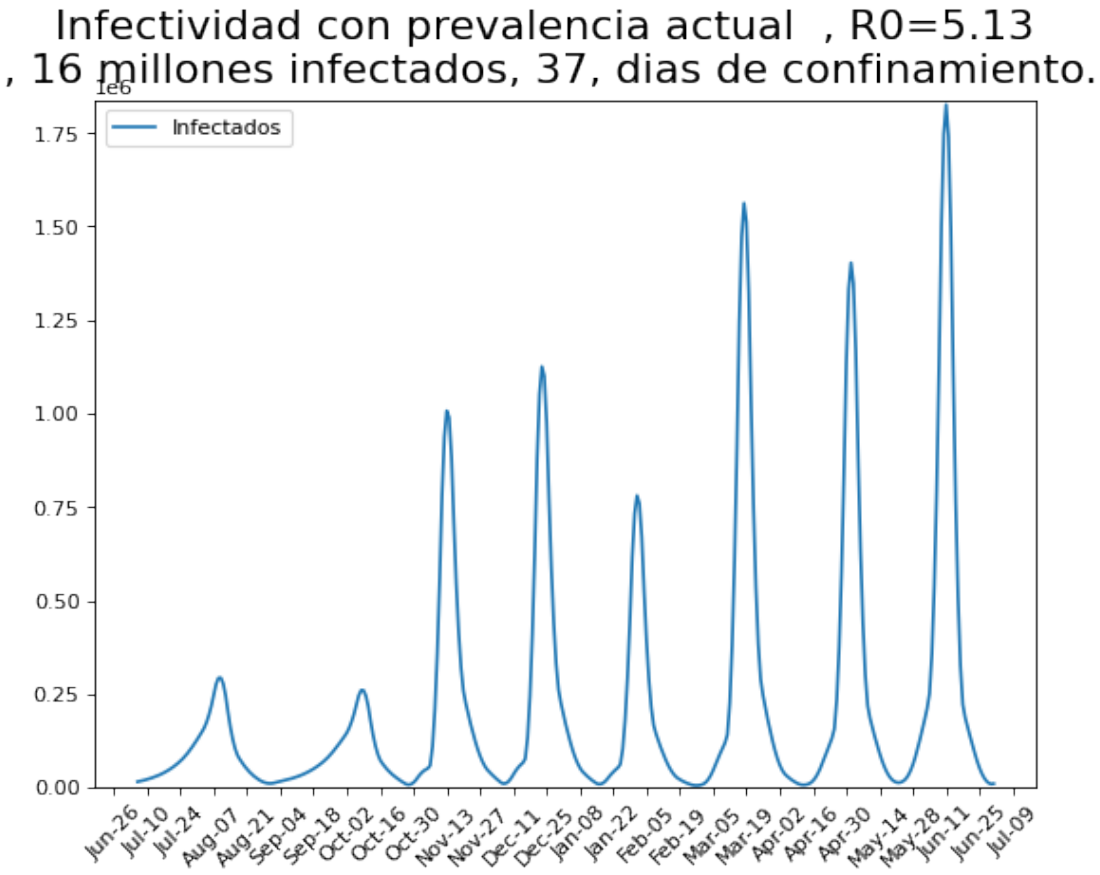
```
            GENERACIONES       = 9        ,
            ARRAY_PREVALENCIAS = [0.049872955555555556]  ,
            FECHA_INICIAL_STR  = 2020-09-09
SITUACION_INICIAL=7585,
            R0                 = 5.7 ,
            GENERACIONES       = 9        ,
            ARRAY_PREVALENCIAS = [0.07729586666666667]  ,
            FECHA_INICIAL_STR  = 2020-11-13
SITUACION_INICIAL=7902,
            R0                 = 5.7 ,
            GENERACIONES       = 9        ,
            ARRAY_PREVALENCIAS = [0.12645424444444445]  ,
            FECHA_INICIAL_STR  = 2020-12-28
SITUACION_INICIAL=5150,
            R0                 = 5.7 ,
            GENERACIONES       = 9        ,
            ARRAY_PREVALENCIAS = [0.16877815555555556]  ,
            FECHA_INICIAL_STR  = 2021-02-11
SITUACION_INICIAL=5957,
            R0                 = 5.7 ,
            GENERACIONES       = 9        ,
            ARRAY_PREVALENCIAS = [0.19222662222222223]  ,
            FECHA_INICIAL_STR  = 2021-03-23
SITUACION_INICIAL=5418,
            R0                 = 5.7 ,
            GENERACIONES       = 9        ,
            ARRAY_PREVALENCIAS = [0.21679804444444445]  ,
            FECHA_INICIAL_STR  = 2021-05-02
SITUACION_INICIAL=10303,
            R0                 = 5.7 ,
            GENERACIONES       = 9        ,
            ARRAY_PREVALENCIAS = [0.2369009111111111]  ,
            FECHA_INICIAL_STR  = 2021-06-06
2021-07-01 2020-07-01 10000 4500000 5.7 0.5 1839090 5000
SITUACION_INICIAL=10000,
            R0                 = 1.702 ,
            GENERACIONES       = 9        ,
            ARRAY_PREVALENCIAS = [0.1]  ,
            FECHA_INICIAL_STR  = 2020-07-01
SITUACION_INICIAL=10920,
            R0                 = 1.702 ,
            GENERACIONES       = 9        ,
            ARRAY_PREVALENCIAS = [0.12333115555555556]  ,
            FECHA_INICIAL_STR  = 2020-08-30
SITUACION_INICIAL=8752,
            R0                 = 5.7 ,
            GENERACIONES       = 9        ,
            ARRAY_PREVALENCIAS = [0.14453366666666667]  ,
```

```
            FECHA_INICIAL_STR  = 2020-10-29
SITUACION_INICIAL=11459,
            R0                 = 5.7 ,
            GENERACIONES       = 9        ,
            ARRAY_PREVALENCIAS = [0.18794984444444446]  ,
            FECHA_INICIAL_STR  = 2020-12-08
SITUACION_INICIAL=9497,
            R0                 = 5.7 ,
            GENERACIONES       = 9        ,
            ARRAY_PREVALENCIAS = [0.23531082222222222]  ,
            FECHA_INICIAL_STR  = 2021-01-17
SITUACION_INICIAL=5246,
            R0                 = 5.7 ,
            GENERACIONES       = 9        ,
            ARRAY_PREVALENCIAS = [0.2676886888888889]  ,
            FECHA_INICIAL_STR  = 2021-02-26
SITUACION_INICIAL=6791,
            R0                 = 5.7 ,
            GENERACIONES       = 9        ,
            ARRAY_PREVALENCIAS = [0.3312570888888889]  ,
            FECHA_INICIAL_STR  = 2021-04-12
SITUACION_INICIAL=12740,
            R0                 = 5.7 ,
            GENERACIONES       = 9        ,
            ARRAY_PREVALENCIAS = [0.38749988888888887]  ,
            FECHA_INICIAL_STR  = 2021-05-22
2021-07-01 2020-07-01 10000 4500000 3.989 0.5 1839090 5000
SITUACION_INICIAL=10000,
            R0                 = 1.702 ,
            GENERACIONES       = 9        ,
            ARRAY_PREVALENCIAS = [0.1]  ,
            FECHA_INICIAL_STR  = 2020-07-01
SITUACION_INICIAL=10920,
            R0                 = 1.702 ,
            GENERACIONES       = 9        ,
            ARRAY_PREVALENCIAS = [0.12333115555555556]  ,
            FECHA_INICIAL_STR  = 2020-08-30
SITUACION_INICIAL=8752,
            R0                 = 3.989 ,
            GENERACIONES       = 9        ,
            ARRAY_PREVALENCIAS = [0.14453366666666667]  ,
            FECHA_INICIAL_STR  = 2020-10-29
SITUACION_INICIAL=11314,
            R0                 = 3.989 ,
            GENERACIONES       = 9        ,
            ARRAY_PREVALENCIAS = [0.19694877777777778]  ,
            FECHA_INICIAL_STR  = 2020-12-28
SITUACION_INICIAL=8930,
```

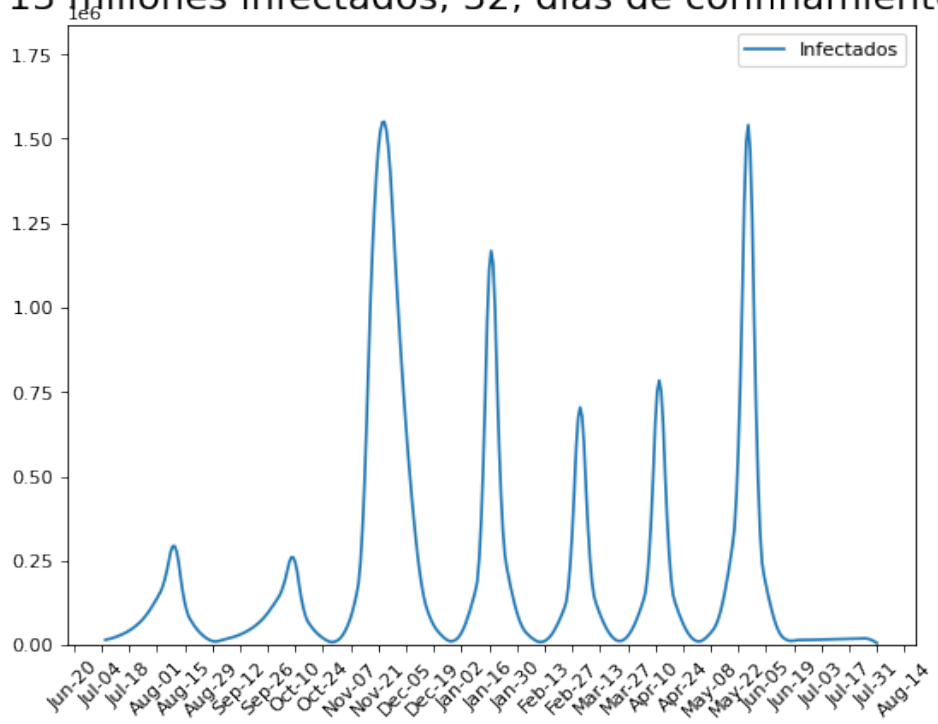
```
            R0                 = 3.989 ,
            GENERACIONES       = 9        ,
            ARRAY_PREVALENCIAS = [0.25033424444444446]  ,
            FECHA_INICIAL_STR  = 2021-02-11
SITUACION_INICIAL=11883,
            R0                 = 3.989 ,
            GENERACIONES       = 9        ,
            ARRAY_PREVALENCIAS = [0.28236104444444443]  ,
            FECHA_INICIAL_STR  = 2021-03-23
SITUACION_INICIAL=10812,
            R0                 = 3.989 ,
            GENERACIONES       = 9        ,
            ARRAY_PREVALENCIAS = [0.31788726666666667]  ,
            FECHA_INICIAL_STR  = 2021-05-02
SITUACION_INICIAL=14016,
            R0                 = 1.702 ,
            GENERACIONES       = 9        ,
            ARRAY_PREVALENCIAS = [0.3877425333333333]  ,
            FECHA_INICIAL_STR  = 2021-06-16
2021-07-01 2020-07-01 10000 4500000 2.314 0.5 1839090 5000
SITUACION_INICIAL=10000,
            R0                 = 1.702 ,
            GENERACIONES       = 9        ,
            ARRAY_PREVALENCIAS = [0.1]  ,
            FECHA_INICIAL_STR  = 2020-07-01
SITUACION_INICIAL=10920,
            R0                 = 1.702 ,
            GENERACIONES       = 9        ,
            ARRAY_PREVALENCIAS = [0.12333115555555556]  ,
            FECHA_INICIAL_STR  = 2020-08-30
SITUACION_INICIAL=8752,
            R0                 = 2.314 ,
            GENERACIONES       = 9        ,
            ARRAY_PREVALENCIAS = [0.14453366666666667]  ,
            FECHA_INICIAL_STR  = 2020-10-29
SITUACION_INICIAL=10171,
            R0                 = 2.314 ,
            GENERACIONES       = 9        ,
            ARRAY_PREVALENCIAS = [0.20411]  ,
            FECHA_INICIAL_STR  = 2020-12-28
SITUACION_INICIAL=7622,
            R0                 = 2.314 ,
            GENERACIONES       = 9        ,
            ARRAY_PREVALENCIAS = [0.28101346666666666]  ,
            FECHA_INICIAL_STR  = 2021-03-03
SITUACION_INICIAL=6121,
            R0                 = 2.314 ,
            GENERACIONES       = 9        ,
```

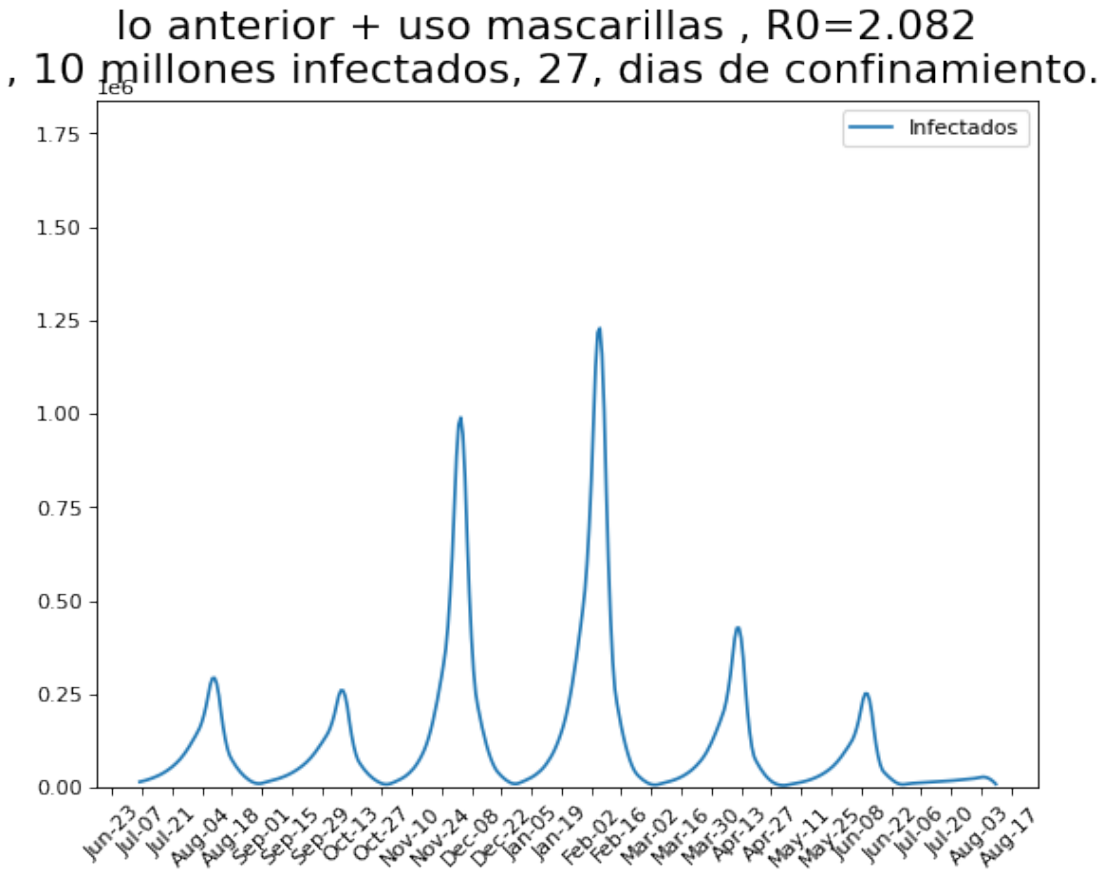
```
            ARRAY_PREVALENCIAS = [0.30979944444444446]  ,
            FECHA_INICIAL_STR  = 2021-05-02
SITUACION_INICIAL=9600,
            R0                 = 1.702 ,
            GENERACIONES       = 9        ,
            ARRAY_PREVALENCIAS = [0.3272810888888889]  ,
            FECHA_INICIAL_STR  = 2021-06-26
```




Infectividad con prevalencia actual , R0=5.13
, 16 millones infectados, 37, dias de confinamiento.
1e6
Infectados
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
Jun-26
Jul-10
Jul-24
Aug-07
Aug-21
Sep-04
Sep-18
Oct-02
Oct-16
Oct-30
Nov-13
Nov-27
Dec-11
Dec-25
Jan-08
Jan-22
Feb-05
Feb-19
Mar-05
Mar-19
Apr-02
Apr-16
Apr-30
May-14
May-28
Jun-11
Jun-25
Jul-09

lo anterior + distanciamiento social efectivo , R0=3.590 , 13 millones infectados, 32, dias de confinamiento.





[15]:
```
"""Infectividad primera ola , R0=5.7                              int64
lo anterior + prevalencia actual , R0=5.13                      int64
lo anterior + distanciamiento social efectivo , R0=3.590        int64
lo anterior + uso mascarillas , R0=2.082                        int64
lo anterior + temperaturas de verano , R0=1.707                 int64
lo anterior + confinamiento , R0=0.5                            int64
dtype: object
"""
```

[15]:
```
'Infectividad primera ola , R0=5.7                              int64\nlo anterior
+ prevalencia actual , R0=5.13                    int64\nlo anterior +
distanciamiento social efectivo , R0=3.590    int64\nlo anterior + uso
mascarillas , R0=2.082                      int64\nlo anterior + temperaturas de
verano , R0=1.707             int64\nlo anterior + confinamiento , R0=0.5
int64\ndtype: object\n'
```

[16]:
```
##### de aqui a abajo, solo debug

SITUACION_INICIAL=829339
```

```
R0                 = 5.7
GENERACIONES       = 15
ARRAY_PREVALENCIAS = [0.10]
FECHA_INICIAL_STR  = '2020-10-04'

Calcular_Cuadro_Prevalencias( SITUACION_INICIAL  = SITUACION_INICIAL   ,
            R0                 = R0 ,
            GENERACIONES       = GENERACIONES        ,
            ARRAY_PREVALENCIAS = ARRAY_PREVALENCIAS  ,
            FECHA_INICIAL_STR  = FECHA_INICIAL_STR )
```

[16]:

	prevalencia 0.1 y R0 5.7
2020-10-09	4254509
2020-10-14	19532857
2020-10-19	41349762
2020-10-24	0
2020-10-29	0
2020-11-03	0
2020-11-08	0
2020-11-13	0
2020-11-18	0
2020-11-23	0
2020-11-28	0
2020-12-03	0
2020-12-08	0
2020-12-13	0

[17]:
```
df['incremento'] = df['Infectados'] - df['Infectados'].shift(1) > 0
df['incremento'].count()
```

[17]: 306

[18]:
```
kk = df.head(20)
kk['incremento'].sum()
```

[18]: 13

[ ]:

[ ]:

[ ]: