# Olas-Epidemicas

May 25, 2021

## 1 Olas epidémicas

Vamos a analizar diferentes escenarios, de olas epidémicas.

Cada hipotesis tiene distintos escenarios. Y cada escenario un juego de parámetros diferentes.

En cada hipotesis, intentaremos encontrar aquellos umbrales de confinamiento/desconfinamiento que minimicen los valores de olas, días de confinamiento, e infectados totales, por cada escenario de cada hipotesis

Tabla de contenidos:

- Cálculo de capacidad del sistema sanitario
- Cálculos de prevalencias
- Cálculos de infectados, sin umbrales de confinamiento

Hipótesis a estudiar :

- Impacto de usar mascarillas.
- Airborne. ¿ Es aerotransportado el virus ?
- ¿ Habrá una segunda ola en vacaciones ?
- Datos de Madrid.
- Casos importados - ¿ Puede haber apertura de aeropuertos ?
- Datos agregados
- Conclusiones

### 1.1 Parametros de la simulación

| Parámetro | Significado |
|---|---|
| FECHA_INICIAL_STR | Fecha de inicio de la simulación |

| Parámetro | Significado |
|---|---|
| FECHA_FINAL_STR | Fecha de fin de la simulación. |
| SITUACION_INICIAL | N° de personas con anticuerpos al comienzo de la simulación. |
| POBLACION_INICIAL_INFECTADA | N° de personas infectadas al comienzo la simulación. |
| R0_max | R0 de la simulación cuando no hay confinamiento. |

| Parámetro | Significado |
|---|---|
| R0_min | R0 de la simulación cuando si hay confinamiento. |
| R0_calor | R0 de la simulación durante el periodo estival |
| Umbral_max | Umbral máximo de personas infectadas a la vez, que si se sobrepasa se inicia el confinamiento. |

| Parámetro | Significado |
|---|---|
| Umbral_min | Umbral mínimo de personas infectadas a la vez, que si se sobrepasase finaliza el confinamiento. |
| UMBRAL_DETECCION_BROTES | Umbral de personas infectadas a la vez, a la que podemos hacer el seguimiento de contactos. |

| Parámetro | Significado |
|---|---|
| PORCENTAJE_DETECCION_BROTES | Porcentaje de contactos infectados de un infectado al que se la hace el seguimiento, que el sistema de detección de brotes puede detectar. |

| Parámetro | Significado |
|---|---|
| INCREMENTOS_SEMANAL_CONFINADOS | Casos importados del extranjero a la semana, cuando si hay confinamiento. |
| INCREMENTOS_SEMANAL_DESCONFINADOS | Casos importados del extranjero a la semana, cuando no hay confinamiento. |

Otros parámetros no configurables - Fecha Inicio y Fecha de Fin periodo estival. - Población total del pais. - Capacidad hospitalaria del pais en camas UCI - Tiempo en días que tarda un infectado en infectar a otros.

[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
from sklearn.model_selection import GridSearchCV,ParameterGrid
from IPython.display import display, HTML
```

```
import janitor
import warnings
warnings.filterwarnings('ignore')
pd.set_option('display.max_rows',100)

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

### 1.2 Cálculo de capacidad del sistema sanitario

[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( RO,
                                  GENERACIONES,
                                  ARRAY_PREVALENCIAS,
                                  SITUACION_INICIAL=1,
                                  FECHA_INICIAL_STR = '2020-02-01',
                                  UMBRAL_DETECCION_BROTES = 0  ,
                                  PORCENTAJE_DETECCION_BROTES = 0,
                                  INCREMENTO_SEMANAL = 0 ,
                                  DEBUG=False):
    if DEBUG :
        print (f""" Calcular_Cuadro_Prevalencias infectados_en_esta_generacion␣
 ↪{RO},{GENERACIONES},{ARRAY_PREVALENCIAS}, {SITUACION_INICIAL},␣
 ↪{FECHA_INICIAL_STR} ,{INCREMENTO_SEMANAL}""")
    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
            if DEBUG : print␣
↪("infectados_en_esta_generacion",infectados_en_esta_generacion,R0,prevalencia,prevalencia_e
            # Si estamos por debajo del umbral de deteccion de brotes,␣
↪aplicamos el PORCENTAJE_DETECCION_BROTES,
            # que disminuye efectivamente el R0
            R0_esta_iteracion = R0 * ( 1 - PORCENTAJE_DETECCION_BROTES) if␣
↪infectados_en_esta_generacion < UMBRAL_DETECCION_BROTES else R0
            infectados_en_esta_generacion = int(infectados_en_esta_generacion *␣
↪R0_esta_iteracion * max(0,( 1 - (prevalencia + prevalencia_esta_iteracion))␣
↪) )
            # Incrementos importados
            infectados_en_esta_generacion += int (INCREMENTO_SEMANAL *␣
↪DIAS_EN_REINFECTAR / 7 )
            # A esta cantidad sumamos el incremento en esta generacion
            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

# 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
```

```
# first execution
GENERACIONES=8
ARRAY_PREVALENCIAS = np.linspace(0,0.70,8)
ARRAY_PREVALENCIAS

from matplotlib import pyplot as plt
import pandas as pd
import numpy as np

def Get_Chart(df, title="default", xlabel="" ):
    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)
    plt.gca().set_xlabel(xlabel)
    return plt

def debug_Calcular_Cuadro_Prevalencias():
# DEBUG
    SITUACION_INICIAL  = 1000
    R0_min             = 1.702
    GENERACIONES_BAJADA= 30
    ARRAY_PREVALENCIAS = [0.1]
    FECHA_INICIAL_STR  = '2020-07-01'
    INCREMENTO_SEMANAL = 50

    df_temp =  Calcular_Cuadro_Prevalencias(
        R0                = R0_min,
        GENERACIONES      = GENERACIONES_BAJADA,
        ARRAY_PREVALENCIAS = ARRAY_PREVALENCIAS,
        SITUACION_INICIAL  = SITUACION_INICIAL,
        FECHA_INICIAL_STR  = FECHA_INICIAL_STR    ,
        DEBUG              = False,
        INCREMENTO_SEMANAL = INCREMENTO_SEMANAL
    )
    return df_temp

##debug_Calcular_Cuadro_Prevalencias()
```

[4]:
```
#### PRedicciones a futuro

"""
Crear 30.
Borrar las mayores de umbral superios.
Aplicar hasta que haya menos de x
"""
from datetime import date
GENERACIONES=15

def calcular_prevision(**kwargs) :
    """Calcula la prevision en ciclos de subida y bajada"""
    """FECHA_FINAL_STR,
    FECHA_INICIAL_STR,
    SITUACION_INICIAL,
    POBLACION_INICIAL_INFECTADA,
    RO_max,
    RO_min,
    Umbral_max,
    Umbral_min):
"""
    FECHA_FINAL_STR                  = kwargs.pop('FECHA_FINAL_STR')
    FECHA_INICIAL_STR                = kwargs.pop('FECHA_INICIAL_STR')
    SITUACION_INICIAL                = kwargs.pop('SITUACION_INICIAL')
    POBLACION_INICIAL_INFECTADA      = kwargs.
 ↪pop('POBLACION_INICIAL_INFECTADA')
    RO_max                           = kwargs.pop('RO_max')
    RO_min                           = kwargs.pop('RO_min')
    RO_calor                         = kwargs.pop('RO_calor',1.702 )
    Umbral_max                       = kwargs.pop('Umbral_max')
    Umbral_min                       = kwargs.pop('Umbral_min')
    GENERACIONES_SUBIDA              = kwargs.pop('GENERACIONES_SUBIDA',16)
    GENERACIONES_BAJADA              = kwargs.pop('GENERACIONES_BAJADA',22)
    UMBRAL_DETECCION_BROTES          = kwargs.
 ↪pop('UMBRAL_DETECCION_BROTES',7200)
    PORCENTAJE_DETECCION_BROTES      = kwargs.
 ↪pop('PORCENTAJE_DETECCION_BROTES',.25)
    INCREMENTOS_SEMANAL_CONFINADOS   = kwargs.
 ↪pop('INCREMENTOS_SEMANAL_CONFINADOS',20)
    INCREMENTOS_SEMANAL_DESCONFINADOS = kwargs.
 ↪pop('INCREMENTOS_SEMANAL_DESCONFINADOS',50)
    DEBUG                            = kwargs.pop('DEBUG',False)



    #DEBUG=True
```

```
   if DEBUG:  print(  f"""FECHA_FINAL_STR={FECHA_FINAL_STR},
       FECHA_INICIAL_STR={FECHA_INICIAL_STR},
       SITUACION_INICIAL={SITUACION_INICIAL},
       POBLACION_INICIAL_INFECTADA={POBLACION_INICIAL_INFECTADA},
       R0_max={R0_max},
       R0_min={R0_min},
       Umbral_max={Umbral_max},
       Umbral_min={Umbral_min},
       INCREMENTOS_SEMANAL_CONFINADOS={INCREMENTOS_SEMANAL_CONFINADOS},
      ␣
↪INCREMENTOS_SEMANAL_DESCONFINADOS={INCREMENTOS_SEMANAL_DESCONFINADOS}""")
   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
       # En verano cambian los parámetros
       PERIODO_CALOR = ( FECHA_INICIAL_STR[5:] > '06-15' ) & (␣
↪FECHA_INICIAL_STR[5:] <  '09-15' )
       R0_DESCONTADO_CALOR = min(R0_max,R0_calor) if PERIODO_CALOR else R0_max
       #GENERACIONES_SUBIDA = 30       if PERIODO_CALOR else kwargs.
↪pop('GENERACIONES_SUBIDA',16)

       if DEBUG:  print(f""" subida SITUACION_INICIAL={SITUACION_INICIAL},
           R0                 = {R0_DESCONTADO_CALOR} ,
           GENERACIONES       = {GENERACIONES_SUBIDA} ,
           ARRAY_PREVALENCIAS = {ARRAY_PREVALENCIAS}  ,
           FECHA_INICIAL_STR  = {FECHA_INICIAL_STR} ,
           df_shape           = {df.shape[0]}   """)
       df_temp =  Calcular_Cuadro_Prevalencias(
           DEBUG = False,
           SITUACION_INICIAL                = SITUACION_INICIAL              ␣
↪  ,
           R0                               = R0_DESCONTADO_CALOR            ␣
↪  ,
           GENERACIONES                     = GENERACIONES_SUBIDA            ␣
↪  ,
```

```
            ARRAY_PREVALENCIAS                = ARRAY_PREVALENCIAS                ␣
↪   ,
            FECHA_INICIAL_STR                 = FECHA_INICIAL_STR                 ␣
↪   ,
            UMBRAL_DETECCION_BROTES           = UMBRAL_DETECCION_BROTES           ␣
↪   ,
            PORCENTAJE_DETECCION_BROTES       = PORCENTAJE_DETECCION_BROTES       ␣
↪   ,
            INCREMENTO_SEMANAL                =␣
↪INCREMENTOS_SEMANAL_DESCONFINADOS  )

        # Borramos las siguientes

        df_temp['Infectados'] = df_temp.iloc[:,0]
        df_temp = df_temp[['Infectados']]

        HACEMOS_LA_BAJADA =df_temp['Infectados'].max() > Umbral_max
        if  df_temp[(df_temp['Infectados'] > (Umbral_max*1) )].size > 0  :
            FECHA_DESDE_LA_QUE_BORRAR = df_temp[(df_temp['Infectados'] >␣
↪(Umbral_max*1) )].index[0]
            otro_index = [fecha for fecha in df_temp.index if fecha <␣
↪FECHA_DESDE_LA_QUE_BORRAR ]
            df_temp = df_temp.head(len(otro_index))

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

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

        SITUACION_INICIAL = df.iloc[-1]['Infectados']
        #df.Infectados.iat[-1] += INCREMENTOS_SEMANAL_DESCONFINADOS

        # Bajada
        if HACEMOS_LA_BAJADA :
            PREVALENCIA = (POBLACION_INICIAL_INFECTADA + df.iloc[:,0].sum()) /
↪45000000
            ARRAY_PREVALENCIAS = []
            ARRAY_PREVALENCIAS.append(PREVALENCIA)
            FECHA_INICIAL_STR = df.index[-1]
            df_temp = pd.DataFrame()

            if DEBUG:  print(f""" bajada SITUACION_INICIAL={SITUACION_INICIAL},
                RO                 = {RO_min} ,
                GENERACIONES       = {GENERACIONES_BAJADA} ,
```

```
                ARRAY_PREVALENCIAS = {ARRAY_PREVALENCIAS}  ,
                FECHA_INICIAL_STR  = {FECHA_INICIAL_STR} ,
                df_shape           = {df.shape[0]}   """)
            df_temp =  Calcular_Cuadro_Prevalencias(
                DEBUG                          = False,
                SITUACION_INICIAL              = SITUACION_INICIAL,
                R0                             = R0_min,
                GENERACIONES                   = GENERACIONES_BAJADA,
                ARRAY_PREVALENCIAS             = ARRAY_PREVALENCIAS,
                FECHA_INICIAL_STR              = FECHA_INICIAL_STR ,
                UMBRAL_DETECCION_BROTES        = UMBRAL_DETECCION_BROTES ,
                PORCENTAJE_DETECCION_BROTES    = PORCENTAJE_DETECCION_BROTES,
                INCREMENTO_SEMANAL             = INCREMENTOS_SEMANAL_CONFINADOS  )

            # Borramos las siguientes
            df_temp['Infectados'] = df_temp.iloc[:,0]
            df_temp = df_temp[['Infectados']]
            if  df_temp[(df_temp['Infectados'] < (Umbral_min*1) )].size > 0  :
                FECHA_DESDE_LA_QUE_BORRAR = df_temp[(df_temp['Infectados'] <␣
↪(Umbral_min*1) )].index[0]
                otro_index = [fecha for fecha in df_temp.index if fecha <␣
↪FECHA_DESDE_LA_QUE_BORRAR ]
                df_temp = df_temp.head(len(otro_index))


            df_temp = pd.DataFrame(df_temp)
            df = pd.concat([df_temp,df])
            df = df.sort_index()
            SITUACION_INICIAL = df.iloc[-1]['Infectados']
            #return df,df_temp
        FECHA_INICIAL_STR = df.index[-1]
        #df.Infectados.iat[-1] += INCREMENTOS_SEMANAL_DESCONFINADOS
        if DEBUG:  print(f""" iteracion :  df_shape= {df.shape[0]} ␣
↪,FECHA_INICIAL_STR={FECHA_INICIAL_STR} ,␣
↪FECHA_FINAL_STR={FECHA_FINAL_STR}""")



    FECHA_DESDE_LA_QUE_BORRAR = FECHA_FINAL_STR
    otro_index = [fecha for fecha in df.index if fecha <=␣
↪FECHA_DESDE_LA_QUE_BORRAR ]
    df = df.head(len(otro_index))

    df = df[df > 0].dropna()
    df = df.dropna()
    df = df.loc[~df.index.duplicated(keep='last')]
```

```
    if DEBUG:  print(f""" final :  df_shape= {df.shape[0]} ␣
 ↪,FECHA_INICIAL_STR={FECHA_INICIAL_STR} ,␣
 ↪FECHA_FINAL_STR={FECHA_FINAL_STR}""")
    return df


def debug_calcular_prevision():
    """ Solo para depurar calcular_prevision()"""
    """
    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'
    """
    SITUACION_INICIAL =10000
    POBLACION_INICIAL_INFECTADA=4500000
    R0_max=5.7
    R0_min=0.5
    Umbral_max=30000
    Umbral_min=5000
    FECHA_FINAL_STR   ='2020-07-25'
    FECHA_INICIAL_STR ='2020-07-01'
    INCREMENTOS_SEMANAL_CONFINADOS=30
    INCREMENTOS_SEMANAL_DESCONFINADOS=60

    df = calcular_prevision(
        FECHA_FINAL_STR = FECHA_FINAL_STR,
        FECHA_INICIAL_STR = FECHA_INICIAL_STR,
        SITUACION_INICIAL = SITUACION_INICIAL,
        POBLACION_INICIAL_INFECTADA =  POBLACION_INICIAL_INFECTADA,
        R0_max = R0_max,
        R0_min =  R0_min,
        Umbral_max =  Umbral_max,
        Umbral_min =  Umbral_min,
        INCREMENTOS_SEMANAL_CONFINADOS    = INCREMENTOS_SEMANAL_CONFINADOS,
        INCREMENTOS_SEMANAL_DESCONFINADOS = INCREMENTOS_SEMANAL_DESCONFINADOS,
        DEBUG=False)
    return df
Umbral_min=10000
```

Hipótesis Impacto de usar mascarillas..

Que pasaría si la infectividad del virus es de una R0 de 5.7. Tenemos varios escenario de R0, con/sin mascarilla, con/sin distanciamiento.

Estos son los peores escenarios.

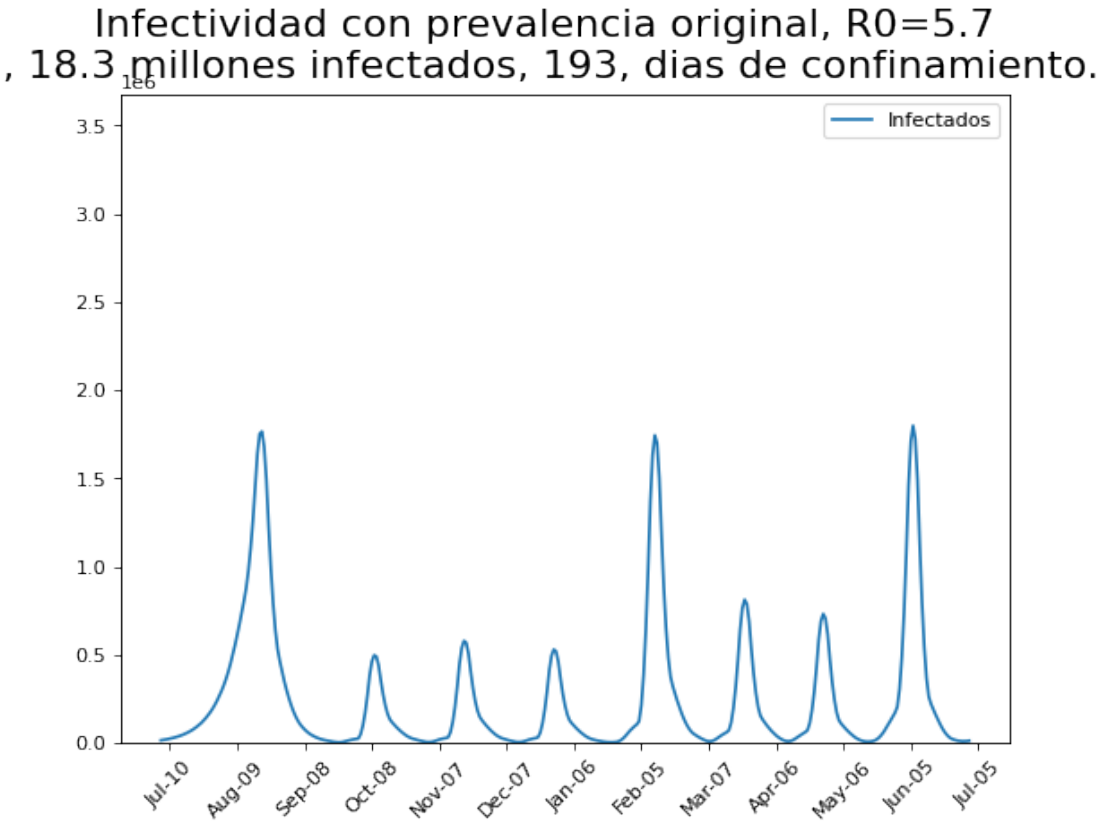
Solo confinamos cuando superamos la capacidad hospitalaria, desconfinamos hasta que llegamos a 5000 nuevas infecciones en 5 días

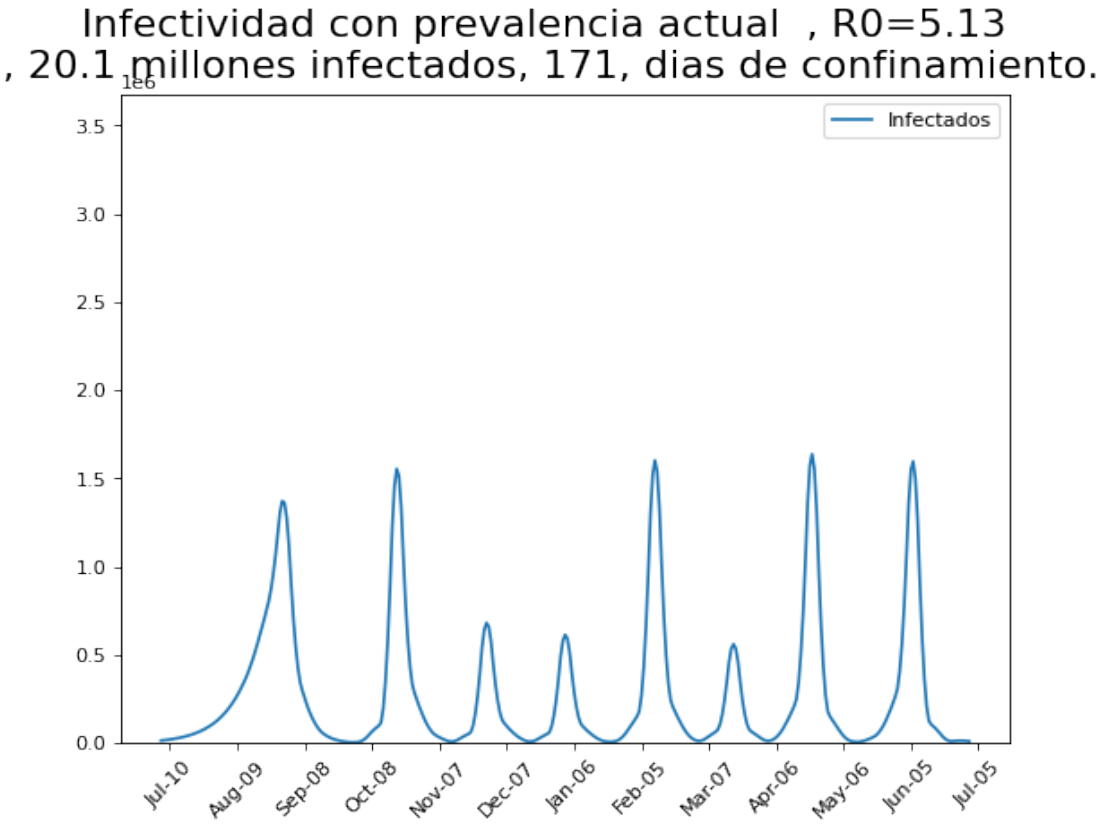
Estos escenarios son solo una linea base, para ver cuanto optimizamos si no tomamos decisiones de confinamiento/desconfinamiento.

[5]:
```
import pickle
from tqdm.notebook import tqdm

""" 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 actual  , R0=5.13"          ␣
 ↪  , 'R0' : 5.7   , 'POBLACION_INICIAL_INFECTADA' : 4500000}
 ]
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} ,
]

default_grid_thresolds ={"Umbral_max": [1800000,
                                        1000000,
                                         750000,
                                         500000,
                                         250000,
                                         125000,
                                         62000,
                                        ],
                         "Umbral_min": [5000, 10000,20000,40000,50000] }
```

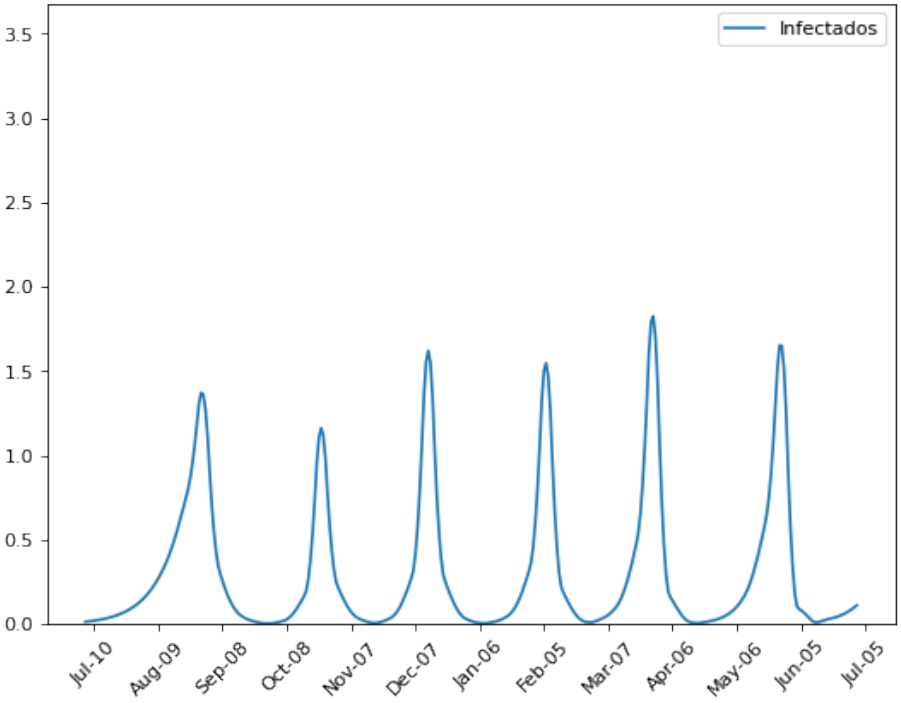
```
df_array = []

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'                   ,
    "UMBRAL_DETECCION_BROTES"          :  7200                          ,
    "PORCENTAJE_DETECCION_BROTES"      :  .25                           ,
    "INCREMENTOS_SEMANAL_DESCONFINADOS" :  50                           ,
    "INCREMENTOS_SEMANAL_CONFINADOS"   :  20
}


def get_lockdown_days(df_temp,Umbral_min):
    df_interpolated=interpolate_dataframe(df_temp,'D')
    #return df_interpolated.shape[0] - (df_interpolated['Infectados'] -␣
 ↪df_interpolated['Infectados'].shift(1) > 0).sum()
    return ((df_interpolated['Infectados'] - df_interpolated['Infectados'].
 ↪shift(1) < 0) & (df_interpolated['Infectados'] > Umbral_min)).sum()
Umbral_max = CAPACIDAD_SISTEMA_SANITARIO

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'                    ␣
 ↪      ],
```

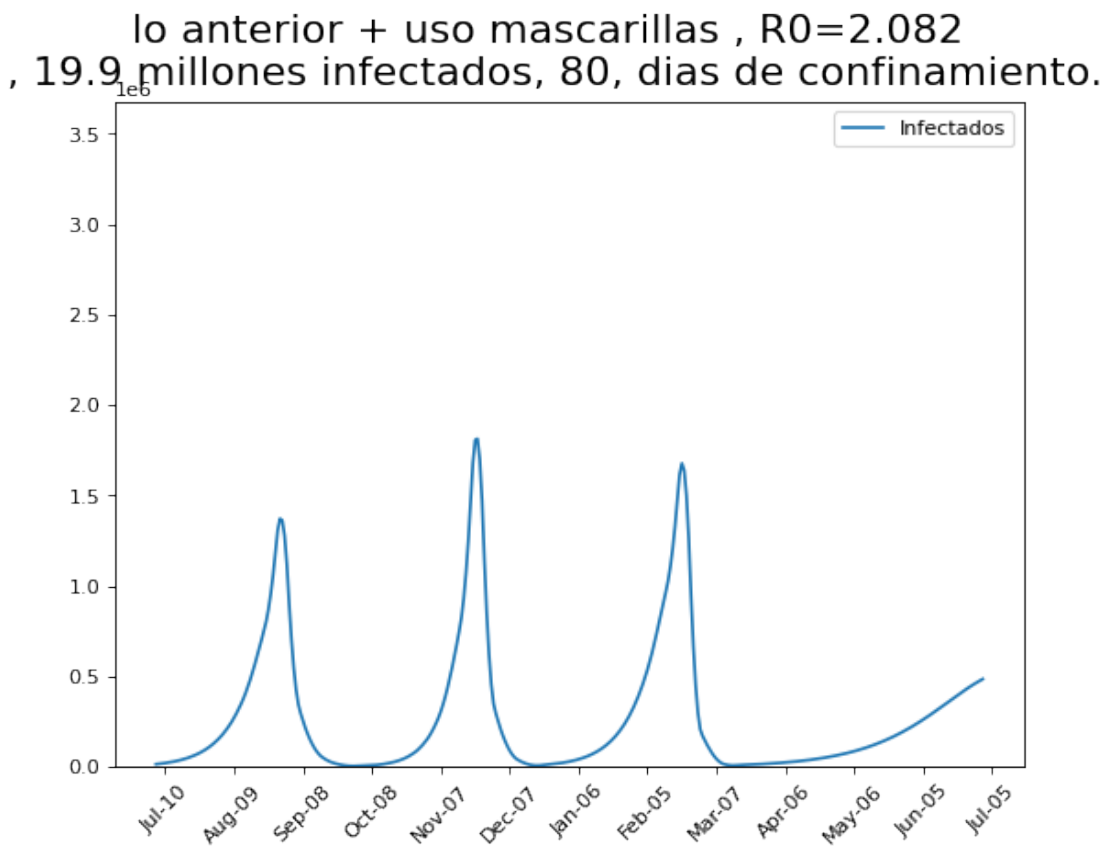
```
        Umbral_max                          = param['Umbral_max'                ␣
↪       ],
        Umbral_min                          = param['Umbral_min'                ␣
↪       ],
        INCREMENTOS_SEMANAL_CONFINADOS    =␣
↪param['INCREMENTOS_SEMANAL_CONFINADOS'    ],
        INCREMENTOS_SEMANAL_DESCONFINADOS =␣
↪param['INCREMENTOS_SEMANAL_DESCONFINADOS' ],
        DEBUG = False
    )
    df_temp

    df_temp = df_temp.astype(np.int64)
    df_temp = df_temp.loc[~df_temp.index.duplicated(keep='last')]
    infectados_totales = int(df_temp.astype(int).sum(axis=0))
    #print(param['descripcion'], df_temp.tail(1).index[-1],"suma: " , suma)
    df_array.append(df_temp)
    DIAS_CONFINAMIENTO = get_lockdown_days(df_temp,param['Umbral_min'])

    plt = Get_Chart(df=interpolate_dataframe(df_temp,'D'),
                    title = param['descripcion'] +" \n, "+␣
↪str(infectados_totales/1000000)[:4] + " millones infectados, " + ␣
↪str(DIAS_CONFINAMIENTO) + ", dias de confinamiento." )
    plt.gca().xaxis.set_major_locator(mdates.DayLocator(interval=30))

    ax = plt.gca()
    ax.set_ylim([0,Umbral_max*2])
    param = {}

df  =  pd.concat(df_array)
#df_temp
```


Infectividad con prevalencia original, R0=5.7
, 18.3 millones infectados, 193, dias de confinamiento.
1e6
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Infectados
Jul-10
Aug-09
Sep-08
Oct-08
Nov-07
Dec-07
Jan-06
Feb-05
Mar-07
Apr-06
May-06
Jun-05
Jul-05


Infectividad con prevalencia actual , R0=5.13
, 20.1 millones infectados, 171, dias de confinamiento.
1e6
Infectados
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Jul-10
Aug-09
Sep-08
Oct-08
Nov-07
Dec-07
Jan-06
Feb-05
Mar-07
Apr-06
May-06
Jun-05
Jul-05

lo anterior + distanciamiento social efectivo , R0=3.590
, 22.0 millones infectados, 139, dias de confinamiento.






[6]:
```
import pickle

def simulation_save_to_file(objeto,nombre_fichero):
    pickle.dump(objeto, open(nombre_fichero, 'wb'))

def simulation_get_from_file(nombre_fichero):
    return pickle.load(open(nombre_fichero, 'rb'))
```

[7]:
```
def␣
 ↪get_results_singlechart_simulacion(results_array,criterio,puesto,mejores_resultados=True):
 ↪
    df_results = pd.DataFrame(results_array)
    df_results.sort_values(criterio, inplace=True,ascending=mejores_resultados)
    row      = df_results.iloc[puesto]

    ### Get Parameters from simulation
    df_temp             = pd.DataFrame(row.df)
    suma                = row.infectados_totales
    dias_confinamiento = row.dias_confinamiento
    infectados_totales = '%.2f' % ( row.infectados_totales /1000000)
```

```
    numero_de_olas      = row['numero_de_olas']
    descripcion         = row['descripcion']
    umbral_maximo       = row['Umbral_max']
    umbral_minimo       = row['Umbral_min']

    lugar = int(puesto) + int(1)
    df_array.append(df_temp)
    #print (f"""criterio: {criterio}, puesto: {lugar}, descripcion␣
↪{descripcion}, umbral_maximo: {umbral_maximo},umbral_minimo:␣
↪{umbral_minimo},
    #infectados_totales: {infectados_totales}, dias_confinamiento:␣
↪{dias_confinamiento}, numero_de_olas: {numero_de_olas})
    #""" )
    plt = Get_Chart(df=interpolate_dataframe(df_temp,'D'),
                title  = f"""{descripcion}, {lugar}º en {criterio}""",
                xlabel = f"""infectados: {infectados_totales}, dias de␣
↪confinamiento: {dias_confinamiento},Umbrales:␣
↪({umbral_maximo},{umbral_minimo}), olas {numero_de_olas}""")

    #plt.gca().xaxis.set_major_locator(mdates.DayLocator(interval=14))
    return plt
```

[8]:
```
import matplotlib.pyplot as plt
import pandas as pd
import matplotlib.dates as mdates

def Get_Chart2(df, title="default", xlabel="", ax=None,f=None, color='r'):
    if ax is None:
        ax = plt.gca()
    plt.axes(ax)
    for ca in df.columns:
        plt.plot(df[ca],color=color)
        plt.legend(df.columns)
        f.suptitle(title, fontsize=20)
    ax.xaxis.set_major_formatter(mdates.DateFormatter('%b-%d'))
    ax.xaxis.set_major_locator(mdates.DayLocator(interval=30))
    plt.xticks(rotation=45)
    plt.gca().set_xlabel(xlabel,horizontalalignment='center')
    return plt


def get_results_multichart_simulacion2(f,␣
↪axes,results_array,criterio,puesto,mejores_resultados=True,color='r'):
    """ Obtiene los n puestos mejores por un criterio.
    Selecciona los df y genera el array de graficos aosciado"""
    df_results = pd.DataFrame(results_array)
    for puesto in range(puesto):
```

```
        df_results.sort_values(criterio,␣
↪inplace=True,ascending=mejores_resultados)
        row     = df_results.iloc[puesto]

        ### Get Parameters from simulation
        df_temp            = pd.DataFrame(row.df)
        suma               = row.infectados_totales
        dias_confinamiento = row.dias_confinamiento
        infectados_totales = '%.2f' % ( row.infectados_totales /1000000)
        numero_de_olas     = row['numero_de_olas']
        descripcion        = row['descripcion']
        umbral_maximo      = row['Umbral_max']
        umbral_minimo      = row['Umbral_min']

        lugar = int(puesto) + int(1)


        plt = Get_Chart2( ax=axes[puesto],f=f,␣
↪df=interpolate_dataframe(df_temp,'D'), color=color,
                    title  = f"""{descripcion}, {lugar} mejores resultados en:␣
↪{criterio}""",
                    xlabel = f"""infectados: {infectados_totales} mill.,  olas␣
↪{numero_de_olas}
dias de confinamiento: {dias_confinamiento},
Umbrales confinar/desconfinar:
({umbral_maximo},{umbral_minimo})""")

    return plt

def debug_get_results_multichart_simulacion2():
    for criterio in␣
↪['dias_confinamiento','numero_de_olas','infectados_totales']:
        f, axes = plt.subplots(1 , 4 ,figsize=(16, 4), sharex=True)

        get_results_multichart_simulacion2(f=f,␣
↪axes=axes,results_array=results_array,
                                criterio =criterio,
                               puesto=4
                               )
```

[9]:
```
import pickle
from tqdm.notebook import tqdm

default_grid_thresolds ={"Umbral_max": [1800000,
                                        1000000,
                                         750000,
```

```
                                        500000,
                                        250000,
                                        125000,
                                        62000,
                                       ],
                          "Umbral_min": [5000, 10000,20000,40000,50000] }

def generate_results_simlation(dict_escenario,␣
↪dict_default_values,parameter_grid = default_grid_thresolds) :
    df_array = []



    grid = ParameterGrid(parameter_grid)



    results_array = [ ]

    for grid_params in  tqdm(grid, desc= dict_escenario['descripcion']) :
        params = {**grid_params, **dict_escenario, ** dict_default_values}

        if  params['Umbral_max'] <  params['Umbral_min']          : continue
        if (params['Umbral_max'] / params['Umbral_min']) <= 2.50  : continue
        if (params['Umbral_max'] - params['Umbral_min']) < 25000  : continue
        # Si hay una situacion donde no puedas ir ni hacia arriba ni hacia abajo
        if (params['Umbral_max'] / params['Umbral_min']) <  params['R0'] /␣
↪params['R0_min']   : continue
        ## Juntamos los valores por defecto, y los que cambian cada vez.
        #print(grid_params,dict_escenario["descripcion"])
        this_iteration_result = {}
        df_temp = pd.DataFrame()
        df_temp = calcular_prevision(
            FECHA_INICIAL_STR                = params['FECHA_INICIAL_STR'     ␣
↪           ],
            FECHA_FINAL_STR                  = params['FECHA_FINAL_STR'       ␣
↪           ],
            SITUACION_INICIAL                = params['SITUACION_INICIAL'     ␣
↪           ],
            POBLACION_INICIAL_INFECTADA      =␣
↪params['POBLACION_INICIAL_INFECTADA'      ],
            R0_max                           = params['R0'                    ␣
↪           ],
            R0_min                           = params['R0_min'                ␣
↪           ],
```

```
            Umbral_max                          = params['Umbral_max'           ␣
↪           ],
            Umbral_min                          = params['Umbral_min'           ␣
↪           ],
            INCREMENTOS_SEMANAL_CONFINADOS    =␣
↪params['INCREMENTOS_SEMANAL_CONFINADOS'   ],
            INCREMENTOS_SEMANAL_DESCONFINADOS =␣
↪params['INCREMENTOS_SEMANAL_DESCONFINADOS']

        )
        # Prepare results
        # Dataframe level
        df_temp = df_temp.astype(np.int64)
        df_temp = df_temp.loc[~df_temp.index.duplicated(keep='last')]
        # Calculate results
        infectados_totales = int(df_temp.astype(int).sum(axis=0))
        df_temp['Diferencia'] =  df_temp['Infectados'] - df_temp['Infectados'].
↪shift(1)

        dias_confinamiento = get_lockdown_days(df_temp,params['Umbral_min'])

        numero_de_olas = (( df_temp['Diferencia'] < 0) & (df_temp['Diferencia'].
↪shift(1) > 0 )  ).sum()
        # Store results
        this_iteration_result['escenario']                          =␣
↪dict_escenario["escenario"]
        this_iteration_result['descripcion']                        =␣
↪dict_escenario["descripcion"]
        this_iteration_result['Umbral_max']                         =␣
↪params['Umbral_max']
        this_iteration_result['Umbral_min']                         =␣
↪params['Umbral_min']
        this_iteration_result['dias_confinamiento']                 =␣
↪dias_confinamiento
        this_iteration_result['numero_de_olas'    ]                 =␣
↪numero_de_olas
        this_iteration_result['infectados_totales']                 =␣
↪infectados_totales
        this_iteration_result['R0']                                 =␣
↪params['R0']
        this_iteration_result['R0_min']                             =␣
↪params['R0_min']
        this_iteration_result['FECHA_INICIAL_STR']                  =␣
↪params['FECHA_INICIAL_STR']
        this_iteration_result['FECHA_FINAL_STR']                    =␣
↪params['FECHA_FINAL_STR']
```

```
        this_iteration_result['SITUACION_INICIAL']                =␣
↪params['SITUACION_INICIAL']
        this_iteration_result['POBLACION_INICIAL_INFECTADA']      =␣
↪params['POBLACION_INICIAL_INFECTADA']
        this_iteration_result['INCREMENTOS_SEMANAL_CONFINADOS']   =␣
↪params['INCREMENTOS_SEMANAL_CONFINADOS']
        this_iteration_result['INCREMENTOS_SEMANAL_DESCONFINADOS'] =␣
↪params['INCREMENTOS_SEMANAL_DESCONFINADOS']
        this_iteration_result['df']                               =␣
↪df_temp['Infectados']

        #print(this_iteration_result)
        results_array.append(this_iteration_result)
    return results_array

def simulation_save_to_file(objeto,nombre_fichero):
    pickle.dump(objeto, open(nombre_fichero, 'wb'))



def␣
↪Generar_Datos_Conjunto_Escenarios(array_parametros,dict_default_values,title):
↪
    display(HTML(f"""<h1 id='{title}'>{title}</h1>"""))

    all_results_array=[]
    # Por cada escenario, generamos las respuestas, y obtenemos las mejores por␣
↪    dias_confinamiento','numero_de_olas','infectados_totales'
    for i,escenario in  tqdm(enumerate(array_parametros), desc="Procesando␣
↪escenarios") :
        results_array = []
        results_array =␣
↪generate_results_simlation(escenario,dict_default_values)
        all_results_array += results_array
        #print(" Escenario: "+ escenario['descripcion'])
        simulation_save_to_file(results_array,"/tmp/simulacion_"+str(i)+".
↪pickle")
    simulation_save_to_file(all_results_array,"/tmp/"+title+"last_scenary_all.
↪pickle")


    return all_results_array


def Generar_Graficos(results_array,escenario,criterio,color):
    f, axes = plt.subplots(1 , 4 ,figsize=(16, 4), sharex=True)
```

```
    get_results_multichart_simulacion2(f=f,␣
 ↪axes=axes,results_array=results_array,
                              criterio =criterio,
                             puesto=4,color=color
                             )

def␣
 ↪Extraer_Escenario_From_Resultados(all_results_array,escenario,criterio,elementos=5):
 ↪
    """Extraer los resultados de un escenario y un criterio"""
    all_results_df   = pd.DataFrame(all_results_array)
    df_escenario     = all_results_df.filter_on(f"""descripcion ==␣
 ↪'{escenario}'"""  )
    array_criterios  = [ ]
    criterios_order  = [ ]
    if criterio == 'infectados_totales' :
        array_criterios      =  [ 'infectados_totales', 'numero_de_olas']
        array_criterio_order =  [ True, True ]
    elif  criterio == 'dias_confinamiento' :
        array_criterios      =  [ 'dias_confinamiento' ,'infectados_totales' ]
        array_criterio_order =  [ True, True]
    elif criterio == 'numero_de_olas' :
        array_criterios      =  [ 'numero_de_olas', 'infectados_totales' ]
        array_criterio_order =  [ True, True]
    #print("ordenamos ",escenario, array_criterios,array_criterio_order)
    df_escenario.sort_values(by =␣
 ↪array_criterios,ascending=array_criterio_order ,inplace=True)
    return df_escenario.head(elementos)

from matplotlib import cm,colors
def Mostrar_Resultados_Conjunto_Escenarios(all_results_array,␣
 ↪ordenar='criterio'):
    # Extraemos resultados
    cmap = colors.ListedColormap(['tab:blue','tab:green','tab:brown'])
    lista_escenarios = pd.DataFrame(all_results_array).descripcion.unique()
    if  ordenar == 'criterio':
       ␣
 ↪ARRAY_CRITERIO=['infectados_totales','dias_confinamiento','numero_de_olas']
        for i,criterio in enumerate(ARRAY_CRITERIO):
            color = cmap(i)
            for escenario in  lista_escenarios:
                escenario_results_array =␣
 ↪Extraer_Escenario_From_Resultados(all_results_array,escenario,criterio,elementos=5)
               ␣
 ↪Generar_Graficos(escenario_results_array,escenario,criterio,color=color)
```

```
    else :
        for escenario in  lista_escenarios:
            ␣
↪ARRAY_CRITERIO=['infectados_totales','dias_confinamiento','numero_de_olas']
            for i,criterio in enumerate( ARRAY_CRITERIO):
                color = cmap(i)
                escenario_results_array =␣
↪Extraer_Escenario_From_Resultados(all_results_array,escenario,criterio,elementos=5)
                ␣
↪Generar_Graficos(escenario_results_array,escenario,criterio,color=color)
```

Generador de resultados.

Por cada escenario, lo probamos con un grid de parámetros configurables. Es decir, que calculamos cada escenario múltiples veces usando distintos parámetros de confinamiento y desconfinamiento, generando un array de resultados posibles por cada escenario.

A continuación escogeremos por cada escenario, aquellos juegos de parámetros que optimicen los resultados de nº de infectados, nº de días confinados, nº de olas.

[10]:
```
array_parametros_estudio_mascarillas = [
  #  { "descripcion" : "Infectividad con prevalencia original, R0=5.7"          ␣
↪    , 'R0' : 5.7   , 'POBLACION_INICIAL_INFECTADA' : 0} ,
    { "escenario": "mascarillas" , "descripcion" : "Infectividad con␣
↪prevalencia actual  , R0=5.13"            , 'R0' : 5.7     } ,
    { "escenario": "mascarillas" , "descripcion" : "lo anterior +␣
↪distanciamiento social efectivo  , R0=3.590" , 'R0' : 3.989   } ,
    { "escenario": "mascarillas" , "descripcion" : "lo anterior +␣
↪distanciamiento social lejania   , R0=2.836" , 'R0' : 3.1515  } ,
    { "escenario": "mascarillas" , "descripcion" : "lo anterior + uso␣
↪mascarillas , R0=2.082"                  , 'R0' : 2.314   } ,
]

dict_default_values ={
    'FECHA_FINAL_STR'                  : '2021-07-01',
    'FECHA_INICIAL_STR'                : '2020-07-01',
    'POBLACION_INICIAL_INFECTADA'      :  45000000*.3 ,
    "R0_min"                           :  0.43        ,
    "SITUACION_INICIAL"                :  25000       ,
    "UMBRAL_DETECCION_BROTES"          :  7200        ,
    "PORCENTAJE_DETECCION_BROTES"      :  .25         ,
    "INCREMENTOS_SEMANAL_CONFINADOS"   :  25          ,
    "INCREMENTOS_SEMANAL_DESCONFINADOS" :  50
    }

mascarillas =␣
↪Generar_Datos_Conjunto_Escenarios(array_parametros_estudio_mascarillas,dict_default_values,
↪de Mascarillas")
```

```
Mostrar_Resultados_Conjunto_Escenarios(mascarillas)
```

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

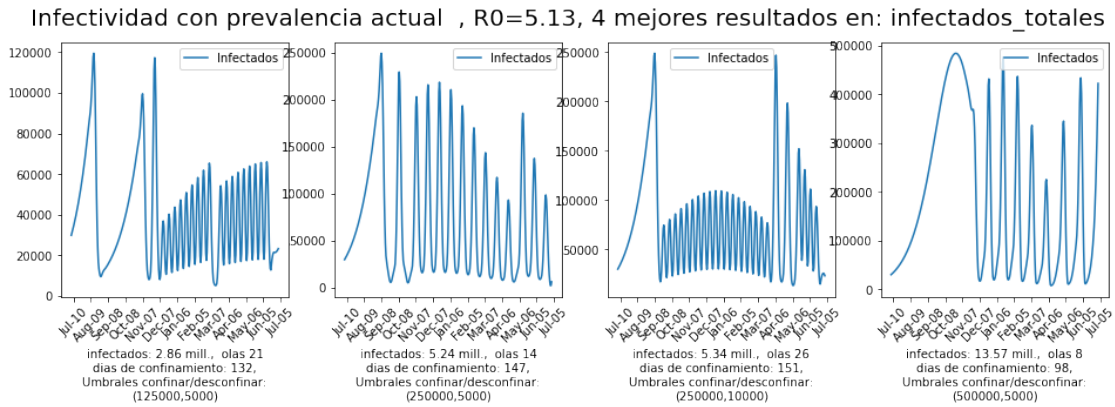
```
HBox(children=(FloatProgress(value=1.0, bar_style='info', description='Procesando escenarios',
```

```
HBox(children=(FloatProgress(value=0.0, description='Infectividad con prevalencia actual  , R0=
```

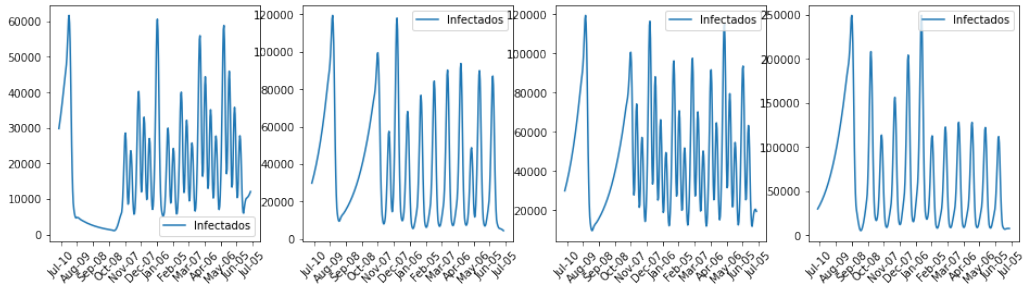
```
HBox(children=(FloatProgress(value=0.0, description='lo anterior + distanciamiento social efect
```

```
HBox(children=(FloatProgress(value=0.0, description='lo anterior + distanciamiento social lejar
```

```
HBox(children=(FloatProgress(value=0.0, description='lo anterior + uso mascarillas , R0=2.082'
```



lo anterior + distanciamiento social efectivo , R0=3.590, 4 mejores resultados en: infectados_totales



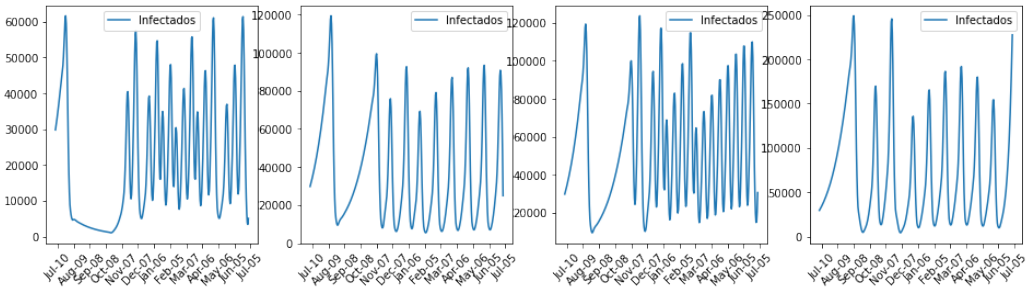

infectados: 1.34 mill., olas 20
dias de confinamiento: 119,
Umbrales confinar/desconfinar:
(62000,5000)

infectados: 2.78 mill., olas 12
dias de confinamiento: 122,
Umbrales confinar/desconfinar:
(125000,5000)

infectados: 3.41 mill., olas 21
dias de confinamiento: 122,
Umbrales confinar/desconfinar:
(125000,10000)

infectados: 4.73 mill., olas 12
dias de confinamiento: 131,
Umbrales confinar/desconfinar:
(250000,5000)

lo anterior + distanciamiento social lejania , R0=2.836, 4 mejores resultados en: infectados_totales



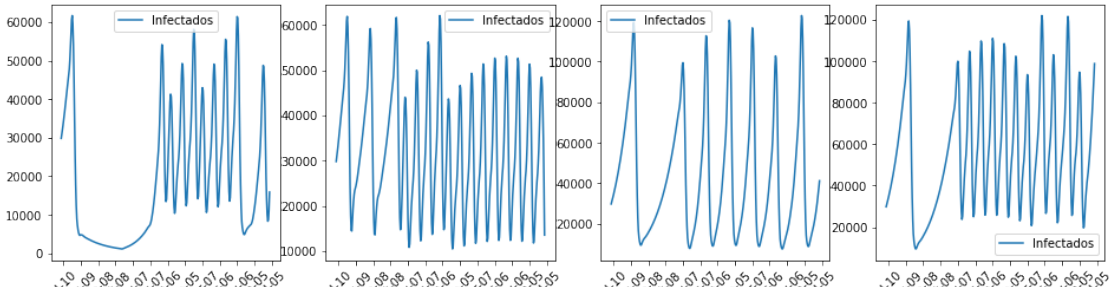

infectados: 1.50 mill., olas 16
dias de confinamiento: 112,
Umbrales confinar/desconfinar:
(62000,5000)

infectados: 2.76 mill., olas 10
dias de confinamiento: 104,
Umbrales confinar/desconfinar:
(125000,5000)

infectados: 3.81 mill., olas 17
dias de confinamiento: 113,
Umbrales confinar/desconfinar:
(125000,10000)

infectados: 5.28 mill., olas 9
dias de confinamiento: 102,
Umbrales confinar/desconfinar:
(250000,5000)

lo anterior + uso mascarillas , R0=2.082, 4 mejores resultados en: infectados_totales



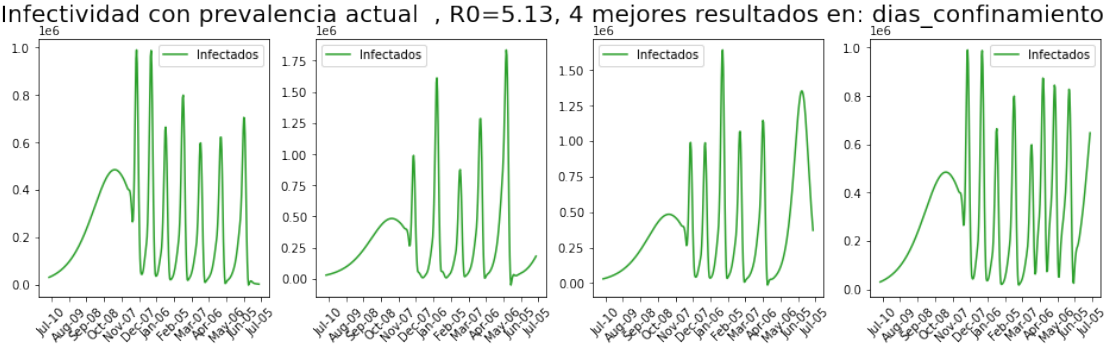

infectados: 1.41 mill., olas 10
dias de confinamiento: 79,
Umbrales confinar/desconfinar:
(62000,5000)

infectados: 2.32 mill., olas 16
dias de confinamiento: 112,
Umbrales confinar/desconfinar:
(62000,10000)

infectados: 3.15 mill., olas 7
dias de confinamiento: 84,
Umbrales confinar/desconfinar:
(125000,5000)

infectados: 4.14 mill., olas 12
dias de confinamiento: 88,
Umbrales confinar/desconfinar:
(125000,10000)

Infectividad con prevalencia actual , R0=5.13, 4 mejores resultados en: dias_confinamiento



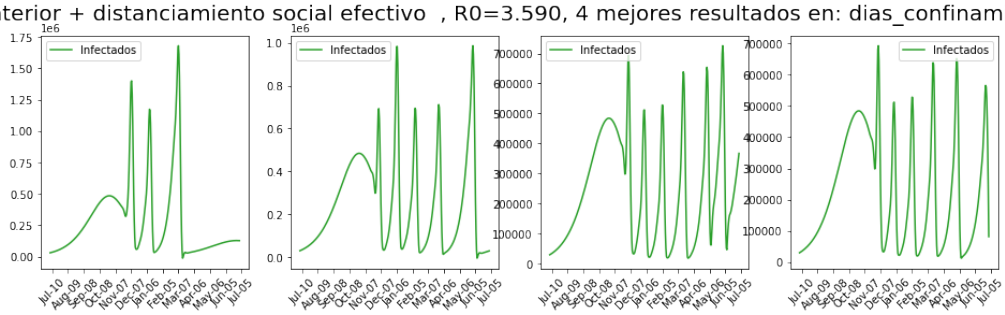

infectados: 17.62 mill., olas 8
dias de confinamiento: 85,
Umbrales confinar/desconfinar:
(1000000,10000)

infectados: 21.53 mill., olas 6
dias de confinamiento: 87,
Umbrales confinar/desconfinar:
(1800000,5000)

infectados: 27.35 mill., olas 7
dias de confinamiento: 88,
Umbrales confinar/desconfinar:
(1800000,10000)

infectados: 22.06 mill., olas 9
dias de confinamiento: 90,
Umbrales confinar/desconfinar:
(1000000,20000)

lo anterior + distanciamiento social efectivo , R0=3.590, 4 mejores resultados en: dias_confinamiento



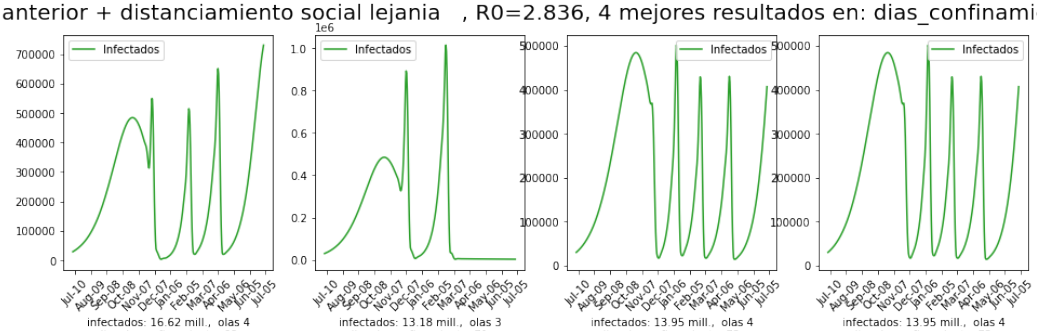

infectados: 20.42 mill., olas 5
dias de confinamiento: 65,
Umbrales confinar/desconfinar:
(1800000,20000)

infectados: 18.34 mill., olas 6
dias de confinamiento: 77,
Umbrales confinar/desconfinar:
(1000000,10000)

infectados: 17.73 mill., olas 7
dias de confinamiento: 82,
Umbrales confinar/desconfinar:
(750000,20000)

infectados: 16.35 mill., olas 7
dias de confinamiento: 83,
Umbrales confinar/desconfinar:
(750000,10000)

lo anterior + distanciamiento social lejania , R0=2.836, 4 mejores resultados en: dias_confinamiento



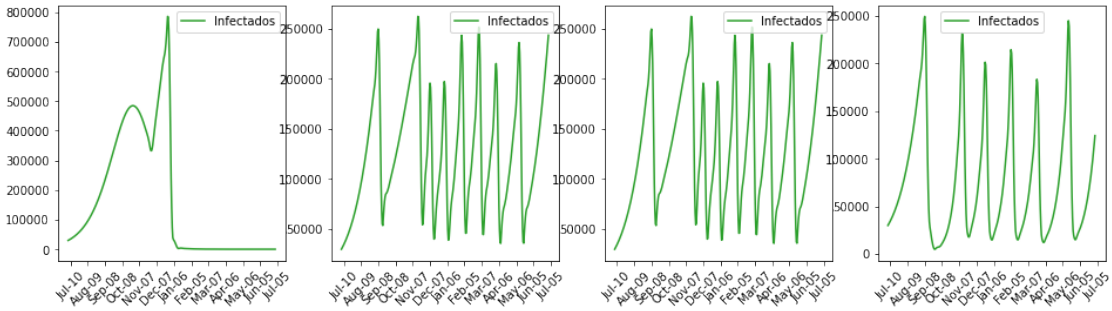

infectados: 16.62 mill., olas 4
dias de confinamiento: 68,
Umbrales confinar/desconfinar:
(750000,5000)

infectados: 13.18 mill., olas 3
dias de confinamiento: 71,
Umbrales confinar/desconfinar:
(1000000,5000)

infectados: 13.95 mill., olas 4
dias de confinamiento: 72,
Umbrales confinar/desconfinar:
(500000,5000)

infectados: 13.95 mill., olas 4
dias de confinamiento: 72,
Umbrales confinar/desconfinar:
(500000,10000)

lo anterior + uso mascarillas , R0=2.082, 4 mejores resultados en: dias_confinamiento



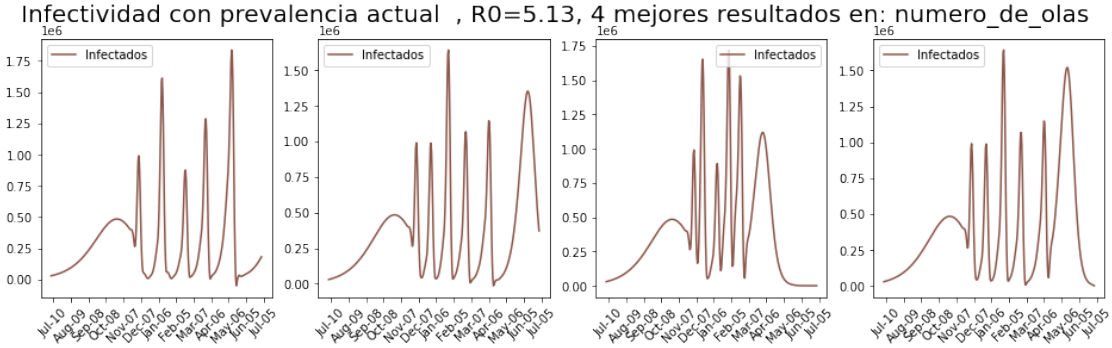

infectados: 11.70 mill., olas 2
dias de confinamiento: 48,
Umbrales confinar/desconfinar:
(750000,5000)

infectados: 9.22 mill., olas 8
dias de confinamiento: 55,
Umbrales confinar/desconfinar:
(250000,40000)

infectados: 9.22 mill., olas 8
dias de confinamiento: 61,
Umbrales confinar/desconfinar:
(250000,20000)

infectados: 5.58 mill., olas 6
dias de confinamiento: 76,
Umbrales confinar/desconfinar:
(250000,5000)

Infectividad con prevalencia actual , R0=5.13, 4 mejores resultados en: numero_de_olas



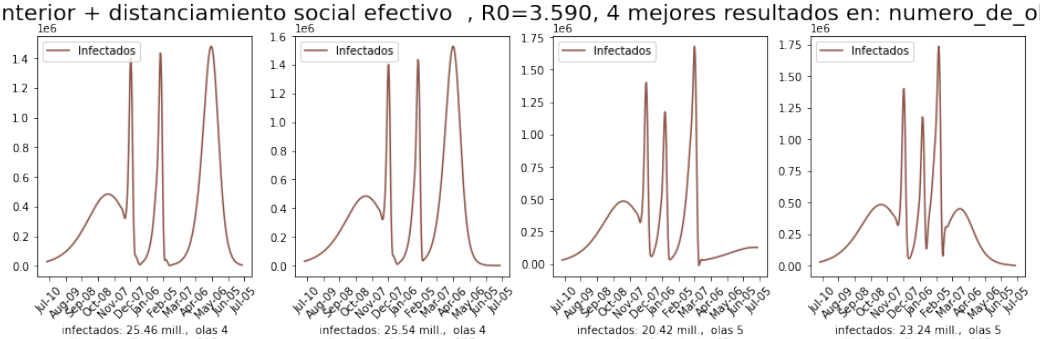

infectados: 21.53 mill., olas 6
dias de confinamiento: 87,
Umbrales confinar/desconfinar:
(1800000,5000)

infectados: 27.35 mill., olas 7
dias de confinamiento: 88,
Umbrales confinar/desconfinar:
(1800000,10000)

infectados: 27.40 mill., olas 7
dias de confinamiento: 100,
Umbrales confinar/desconfinar:
(1800000,50000)

infectados: 27.94 mill., olas 7
dias de confinamiento: 107,
Umbrales confinar/desconfinar:
(1800000,20000)

lo anterior + distanciamiento social efectivo , R0=3.590, 4 mejores resultados en: numero_de_olas

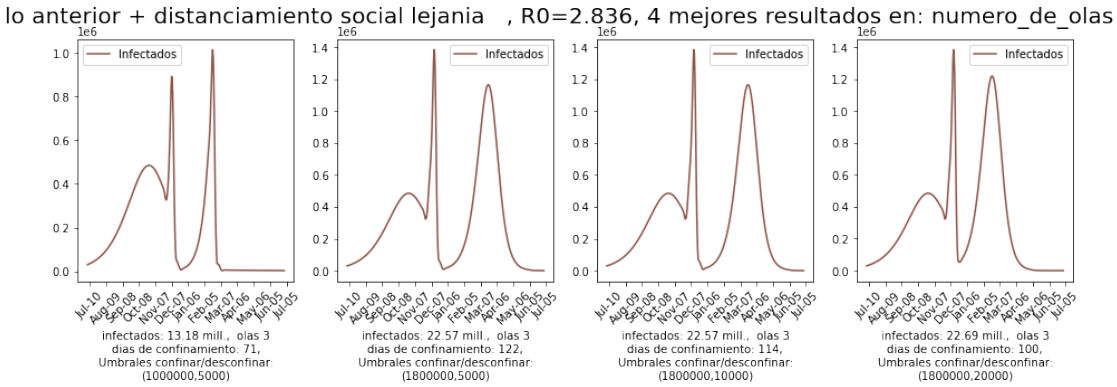


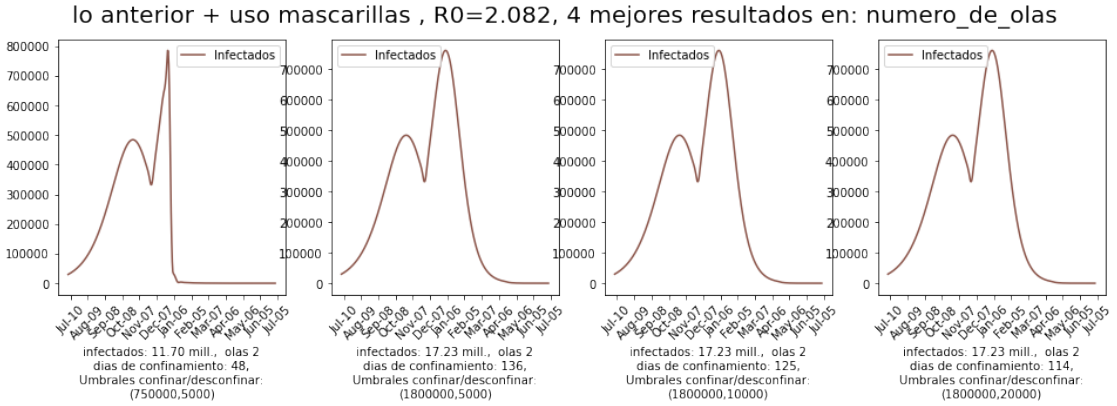

infectados: 25.46 mill., olas 4
dias de confinamiento: 119,
Umbrales confinar/desconfinar:
(1800000,5000)

infectados: 25.54 mill., olas 4
dias de confinamiento: 107,
Umbrales confinar/desconfinar:
(1800000,10000)

infectados: 20.42 mill., olas 5
dias de confinamiento: 65,
Umbrales confinar/desconfinar:
(1800000,20000)

infectados: 23.24 mill., olas 5
dias de confinamiento: 119,
Umbrales confinar/desconfinar:
(1800000,40000)




## 1.3 Segunda hipotesis : Airborne

La hipotesis a estudiar aquí, es que el virus sea aerotransportado

Es decir, que la capacidad principal de propagación del virus sea a traves del aire.

De ser esto cierto, las mascarillas jugarían un papel fundamental.

Esta hipotesis sigue este estudio https://www.pnas.org/content/early/2020/06/10/2009637117, y las R0 son calculadas a partir del mismo

[11]:
```
array_parametros_airborne = [
    { "escenario": "airborne" , "descripcion" : "Airborne, 33% poblacion usa␣
 ↪mascarillas, R0=1.84" , 'R0' : 1.84 } ,
    { "escenario": "airborne" , "descripcion" : "Airborne, 50% poblacion usa␣
 ↪mascarillas, R0=1.35" , 'R0' : 1.35 } ,
    { "escenario": "airborne" , "descripcion" : "Airborne, 66% poblacion usa␣
 ↪mascarillas, R0=1.07" , 'R0' : 1.07 } ,

]
```

```
dict_default_values ={
 'FECHA_FINAL_STR'                   : '2021-07-01' ,
 'FECHA_INICIAL_STR'                 : '2020-07-01' ,
 'POBLACION_INICIAL_INFECTADA'       : 4500000      ,
 'R0_min'                            : 0.43         ,
 'SITUACION_INICIAL'                 : 25000        ,
 'GENERACIONES_SUBIDA'               : 1           ,
 'GENERACIONES_BAJADA'               : 1           ,
 "UMBRAL_DETECCION_BROTES"           :  7200        ,
 "PORCENTAJE_DETECCION_BROTES"       :  .25        ,
 "INCREMENTOS_SEMANAL_CONFINADOS"    :  25          ,
 "INCREMENTOS_SEMANAL_DESCONFINADOS" :  50 }

airborne =␣
 ↪Generar_Datos_Conjunto_Escenarios(array_parametros_airborne,dict_default_values,"Analisis␣
 ↪airborne medio de transporte")
Mostrar_Resultados_Conjunto_Escenarios(airborne)
```

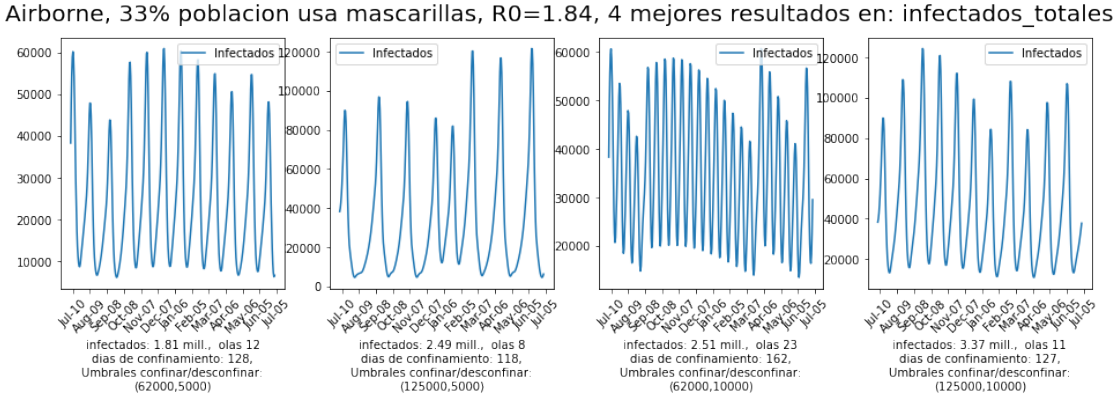
```
<IPython.core.display.HTML object>
```

```
HBox(children=(FloatProgress(value=1.0, bar_style='info', description='Procesando escenarios',
```

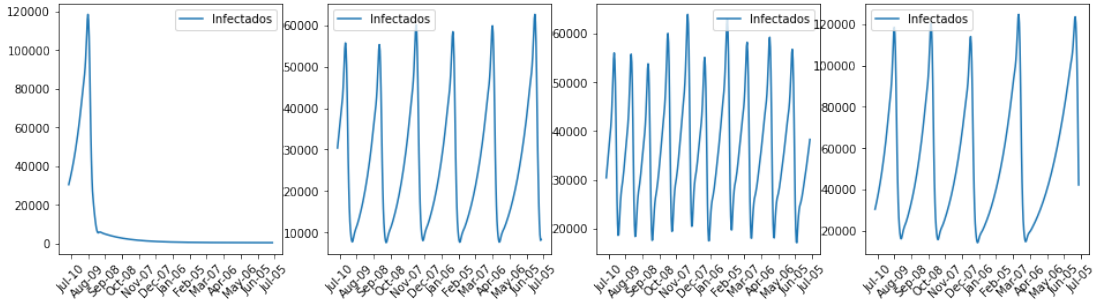
```
HBox(children=(FloatProgress(value=0.0, description='Airborne, 33% poblacion usa mascarillas, I
```

```
HBox(children=(FloatProgress(value=0.0, description='Airborne, 50% poblacion usa mascarillas, I
```

```
HBox(children=(FloatProgress(value=0.0, description='Airborne, 66% poblacion usa mascarillas, I
```



Airborne, 50% poblacion usa mascarillas, R0=1.35, 4 mejores resultados en: infectados_totales



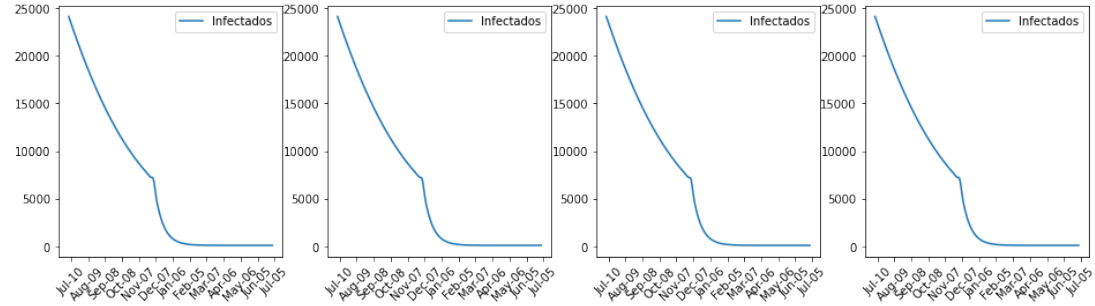

infectados: 0.66 mill., olas 1
dias de confinamiento: 25,
Umbrales confinar/desconfinar:
(125000,5000)

infectados: 1.93 mill., olas 6
dias de confinamiento: 70,
Umbrales confinar/desconfinar:
(62000,5000)

infectados: 2.64 mill., olas 10
dias de confinamiento: 74,
Umbrales confinar/desconfinar:
(62000,10000)

infectados: 3.93 mill., olas 5
dias de confinamiento: 54,
Umbrales confinar/desconfinar:
(125000,10000)

Airborne, 66% poblacion usa mascarillas, R0=1.07, 4 mejores resultados en: infectados_totales



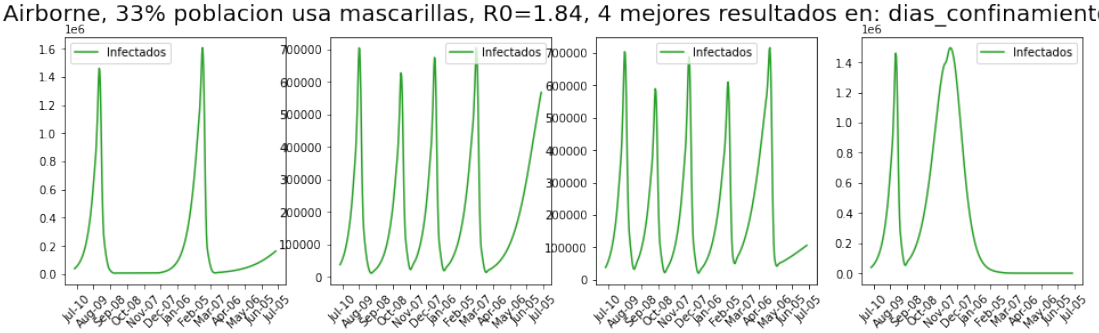

infectados: 0.46 mill., olas 0
dias de confinamiento: 155,
Umbrales confinar/desconfinar:
(1800000,5000)

infectados: 0.46 mill., olas 0
dias de confinamiento: 108,
Umbrales confinar/desconfinar:
(1800000,10000)

infectados: 0.46 mill., olas 0
dias de confinamiento: 24,
Umbrales confinar/desconfinar:
(1800000,20000)

infectados: 0.46 mill., olas 0
dias de confinamiento: 0,
Umbrales confinar/desconfinar:
(1800000,40000)

Airborne, 33% poblacion usa mascarillas, R0=1.84, 4 mejores resultados en: dias_confinamiento



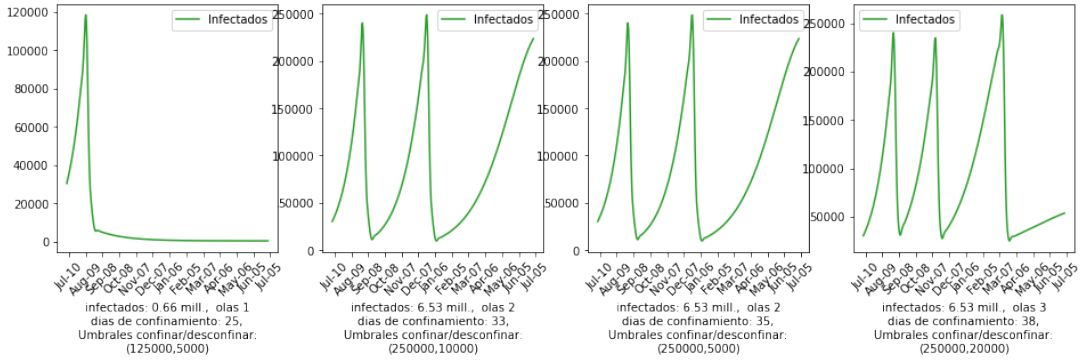

infectados: 13.54 mill., olas 2
dias de confinamiento: 54,
Umbrales confinar/desconfinar:
(1800000,5000)

infectados: 14.42 mill., olas 4
dias de confinamiento: 73,
Umbrales confinar/desconfinar:
(750000,10000)

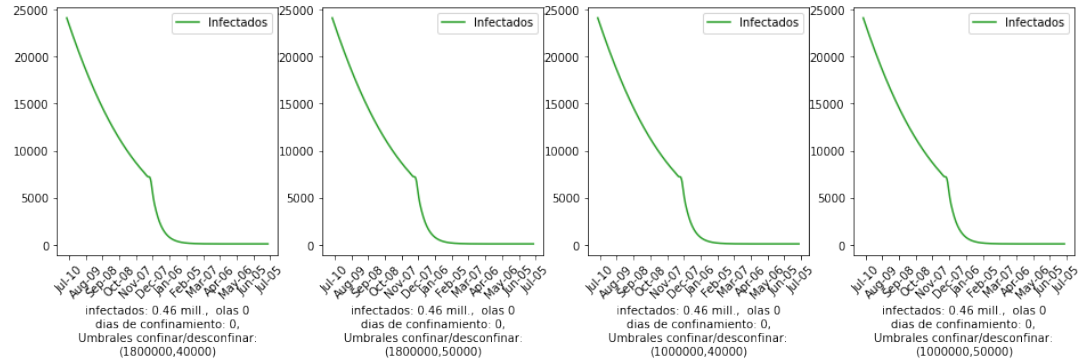
infectados: 14.60 mill., olas 5
dias de confinamiento: 75,
Umbrales confinar/desconfinar:
(750000,20000)

infectados: 24.07 mill., olas 2
dias de confinamiento: 79,
Umbrales confinar/desconfinar:
(1800000,50000)

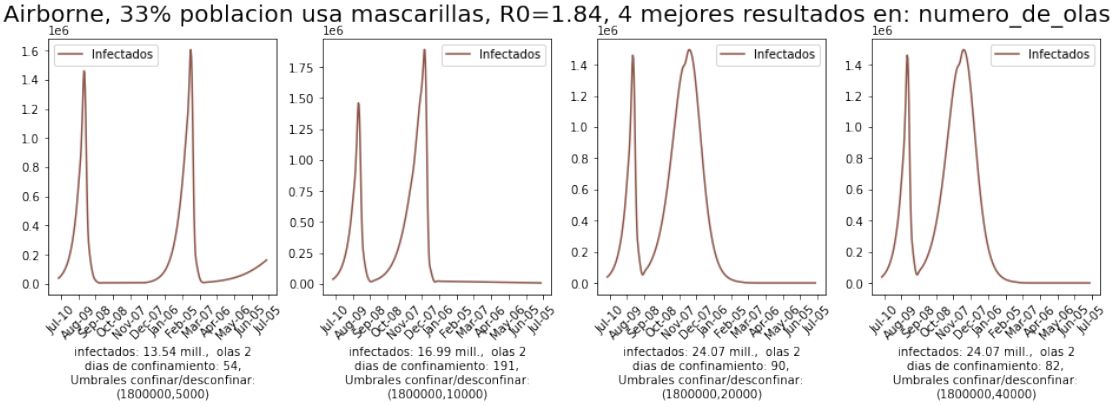
Airborne, 50% poblacion usa mascarillas, R0=1.35, 4 mejores resultados en: dias_confinamiento

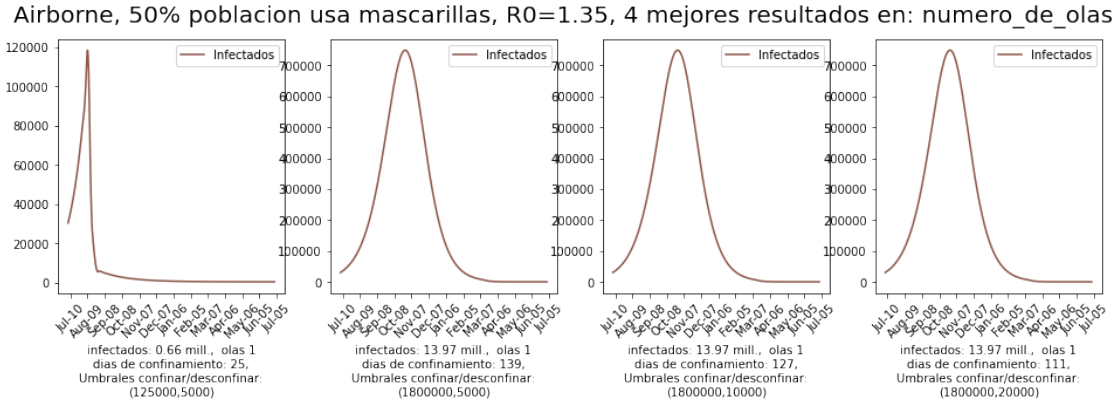


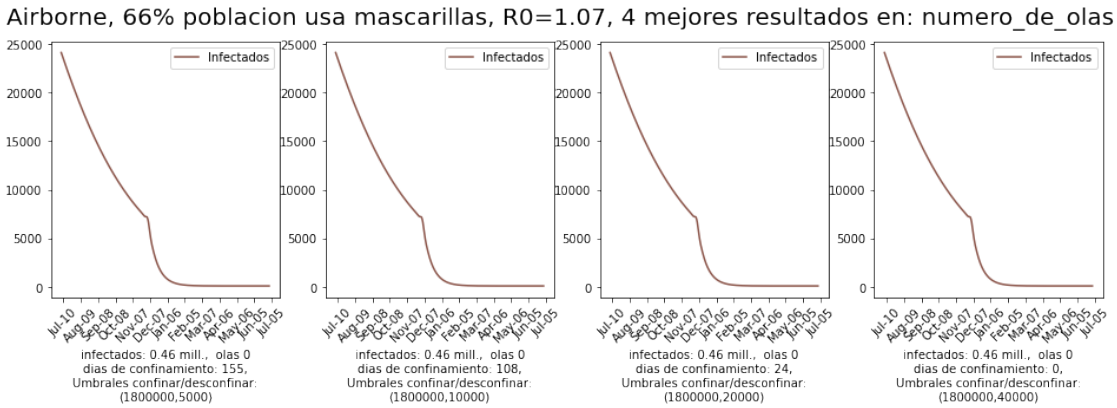

Airborne, 66% poblacion usa mascarillas, R0=1.07, 4 mejores resultados en: dias_confinamiento




Airborne, 33% poblacion usa mascarillas, R0=1.84, 4 mejores resultados en: numero_de_olas






## 1.4 Conclusiones del estudio de la hipotesis "Airbone", que el virus es aerotransportado.

- Las cifras de R0 de este estudio son mas bajas que las de otros, y abundan en la importancia de las mascarillas frente al confinamiento.
- De ser veraz el estudio, basta con que dos de cada 3 personas llevaran mascarillas, para que la epidemia se autoextinguiera - pero si baja del 66% al 50 %, ya no se extingue y habría al menos 2 millones de infectados.
- En este escenario, un 13% de personas que lleven mascarilla es la diferencia entre que vivan o mueran unas 5.000 personas.

## 1.5 Hipotesis vacaciones: ¿ Habrá ola durante el verano. ?

Vamos a extrapolar los datos de muertes en Madrid, para estimar el nº de infectados, y probar con distintos valores de R0 durante el verano para saber si habrá segunda ola en verano.

[12]:
```
# Ver los datos del notebook Madrid_Pain_Graphs
MADRID_MUERTES=15000
```

```
MADRID_CASOS_PCR=323000
MADRID_CASOS_POR_PREVALENCIA=7000000*0.1

MADRID_CASOS_TOTALES_POR_MUERTE=MADRID_CASOS_POR_PREVALENCIA/MADRID_MUERTES
#MADRID_CASOS_TOTALES_POR_MUERTE
#46.666666666666664
MADRID_MUERTES_DIARIAS_MA=13
ESTIMACION_CASOS_MADRID=MADRID_MUERTES_DIARIAS_MA*MADRID_CASOS_TOTALES_POR_MUERTE
#ESTIMACION_CASOS_MADRID
#606.6666666666666
# MADRID REPRESENTA 1/3 CASOS TOTALES. ASI QUE

ESTIMACION_CASOS_ESPAÑA=ESTIMACION_CASOS_MADRID*3
ESTIMACION_CASOS_ESPAÑA
#1820
ESTIMACION_CASOS_ESPAÑA_DIARIA=ESTIMACION_CASOS_MADRID*3
ESTIMACION_CASOS_ESPAÑA_EN_UNA_GENERACION_INFECCION=ESTIMACION_CASOS_ESPAÑA_DIARIA*5
ESTIMACION_CASOS_ESPAÑA_EN_UNA_GENERACION_INFECCION
#9100
ESTIMACION_CASOS_ESPAÑA_EN_UNA_GENERACION_INFECCION=9100
ESTIMACION_CASOS_ESPAÑA_EN_UNA_GENERACION_INFECCION
```

[12]:
```
9100
```

[13]:
```
ESTIMACION_CASOS_ESPAÑA_EN_UNA_GENERACION_INFECCION=26400
array_parametros_estudio_vacaciones = [
 { "escenario": "vacaciones" , "descripcion" : "vacaciones R0=1.1" ,"R0_calor" :
 ↪ 1.10 ,  "R0" : 1.1} ,
 { "escenario": "vacaciones" , "descripcion" : "vacaciones R0=1.2" ,"R0_calor" :
 ↪ 1.20  , "R0" : 1.2} ,
 { "escenario": "vacaciones" , "descripcion" : "vacaciones R0=1.3" ,"R0_calor" :
 ↪ 1.30  , "R0" : 1.3} ,
 { "escenario": "vacaciones" , "descripcion" : "vacaciones R0=1.4" ,"R0_calor" :
 ↪ 1.40  , "R0" : 1.4} ,
 { "escenario": "vacaciones" , "descripcion" : "vacaciones R0=1.5" ,"R0_calor" :
 ↪ 1.50  , "R0" : 1.5} ,
 { "escenario": "vacaciones" , "descripcion" : "vacaciones R0=1.6" ,"R0_calor" :
 ↪ 1.60  , "R0" : 1.6} ,
 { "escenario": "vacaciones" , "descripcion" : "vacaciones R0=1.7" ,"R0_calor" :
 ↪ 1.70  , "R0" : 1.7} ,
 { "escenario": "vacaciones" , "descripcion" : "vacaciones R0=1.8" ,"R0_calor" :
 ↪ 1.80  , "R0" : 1.8} ,
 { "escenario": "vacaciones" , "descripcion" : "vacaciones R0=1.9" ,"R0_calor" :
 ↪ 1.90  , "R0" : 1.9}
]
```

```
dict_default_values = {
    "R0_min" : 0.5,
    "SITUACION_INICIAL" : ESTIMACION_CASOS_ESPAÑA_EN_UNA_GENERACION_INFECCION  ,
    "FECHA_INICIAL_STR" : '2020-06-20'  ,
    "FECHA_FINAL_STR"   : '2020-09-20'  ,
    'GENERACIONES_SUBIDA' : 1 , 'GENERACIONES_BAJADA' : 1,
    'POBLACION_INICIAL_INFECTADA' : 4500000  ,
    'INCREMENTOS_SEMANAL_DESCONFINADOS' : 50,
     "INCREMENTOS_SEMANAL_CONFINADOS"    :  20     ,
    "UMBRAL_DETECCION_BROTES"           :  7200                          ,
    "PORCENTAJE_DETECCION_BROTES"      :  .25
    }

vacaciones =␣
 ↪Generar_Datos_Conjunto_Escenarios(array_parametros_estudio_vacaciones,dict_default_values,"
Mostrar_Resultados_Conjunto_Escenarios(vacaciones)
```

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

```
HBox(children=(FloatProgress(value=1.0, bar_style='info', description='Procesando escenarios',
```

```
HBox(children=(FloatProgress(value=0.0, description='vacaciones R0=1.1', max=35.0, style=Progre
```

```
HBox(children=(FloatProgress(value=0.0, description='vacaciones R0=1.2', max=35.0, style=Progre
```

```
HBox(children=(FloatProgress(value=0.0, description='vacaciones R0=1.3', max=35.0, style=Progre
```

```
HBox(children=(FloatProgress(value=0.0, description='vacaciones R0=1.4', max=35.0, style=Progre
```

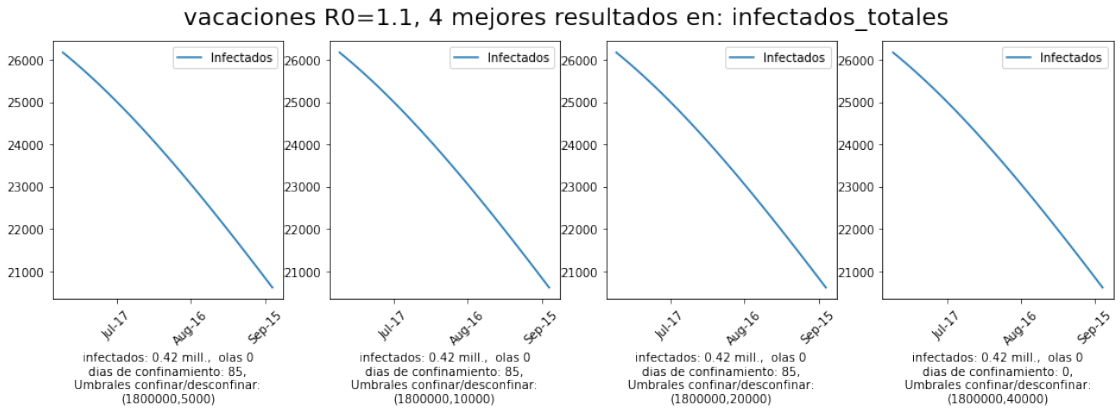
```
HBox(children=(FloatProgress(value=0.0, description='vacaciones R0=1.5', max=35.0, style=Progre
```

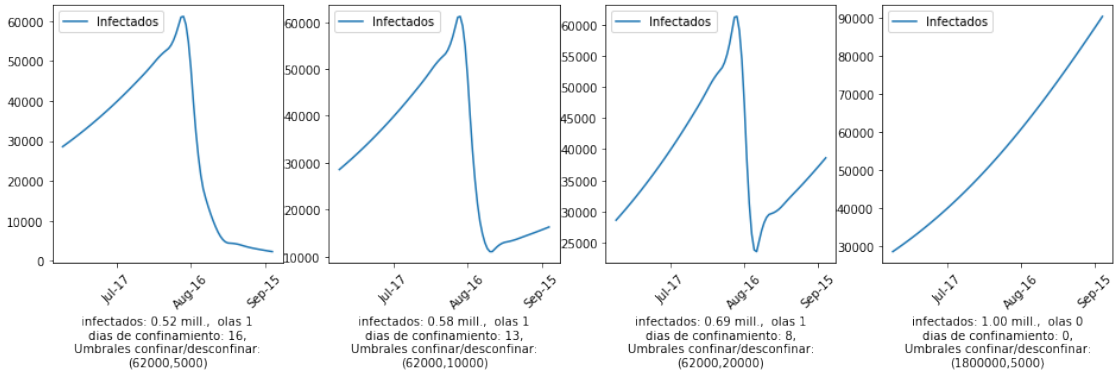
```
HBox(children=(FloatProgress(value=0.0, description='vacaciones R0=1.6', max=35.0, style=Progre
```

HBox(children=(FloatProgress(value=0.0, description='vacaciones R0=1.7', max=35.0, style=Progre

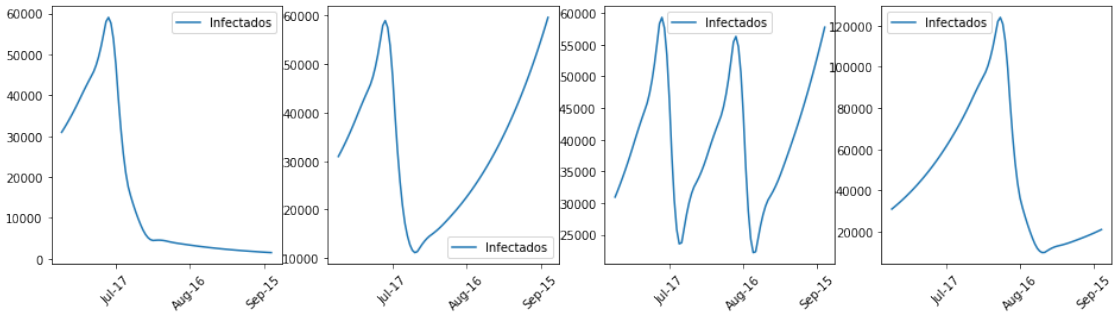
HBox(children=(FloatProgress(value=0.0, description='vacaciones R0=1.8', max=35.0, style=Progre

HBox(children=(FloatProgress(value=0.0, description='vacaciones R0=1.9', max=35.0, style=Progre



vacaciones R0=1.3, 4 mejores resultados en: infectados_totales



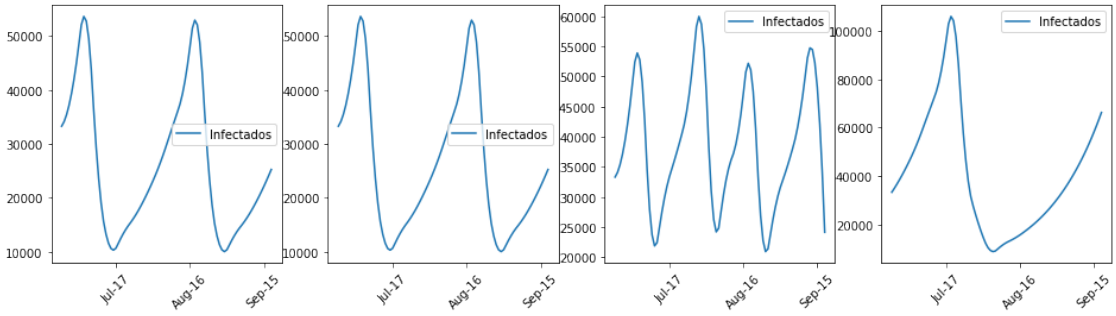

infectados: 0.29 mill., olas 1
dias de confinamiento: 16,
Umbrales confinar/desconfinar:
(62000,5000)

infectados: 0.60 mill., olas 1
dias de confinamiento: 12,
Umbrales confinar/desconfinar:
(62000,10000)

infectados: 0.72 mill., olas 2
dias de confinamiento: 14,
Umbrales confinar/desconfinar:
(62000,20000)

infectados: 0.85 mill., olas 1
dias de confinamiento: 17,
Umbrales confinar/desconfinar:
(125000,5000)

vacaciones R0=1.4, 4 mejores resultados en: infectados_totales



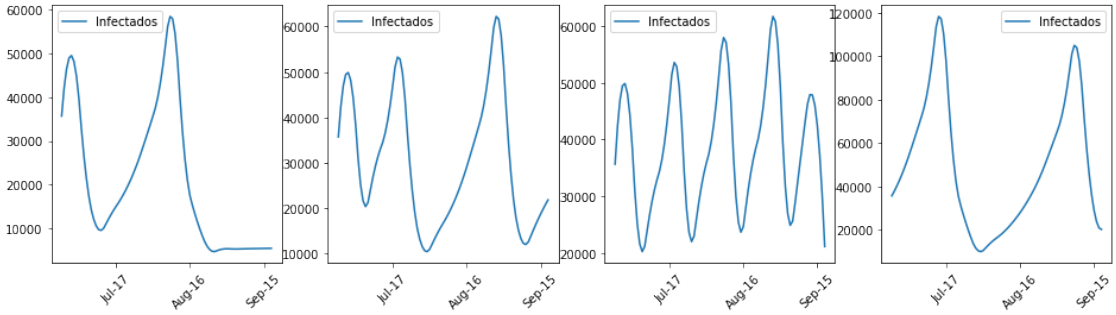

infectados: 0.48 mill., olas 2
dias de confinamiento: 24,
Umbrales confinar/desconfinar:
(62000,5000)

infectados: 0.48 mill., olas 2
dias de confinamiento: 24,
Umbrales confinar/desconfinar:
(62000,10000)

infectados: 0.68 mill., olas 4
dias de confinamiento: 27,
Umbrales confinar/desconfinar:
(62000,20000)

infectados: 0.75 mill., olas 1
dias de confinamiento: 17,
Umbrales confinar/desconfinar:
(125000,5000)

vacaciones R0=1.5, 4 mejores resultados en: infectados_totales



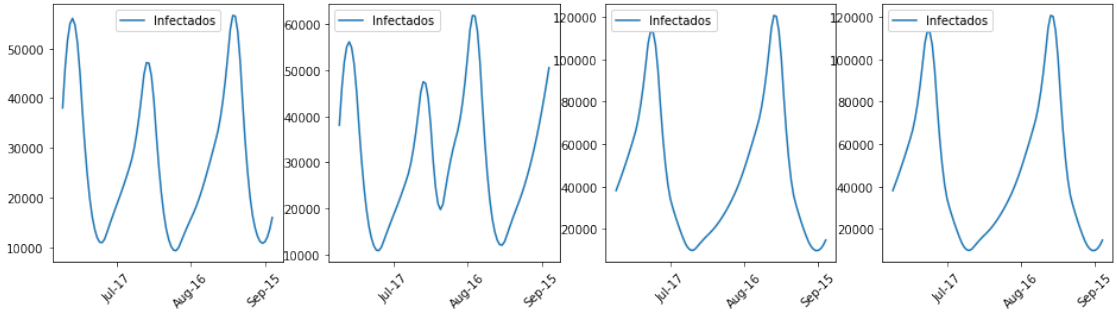

infectados: 0.37 mill., olas 2
dias de confinamiento: 32,
Umbrales confinar/desconfinar:
(62000,5000)

infectados: 0.54 mill., olas 3
dias de confinamiento: 31,
Umbrales confinar/desconfinar:
(62000,10000)

infectados: 0.68 mill., olas 5
dias de confinamiento: 34,
Umbrales confinar/desconfinar:
(62000,20000)

infectados: 0.88 mill., olas 2
dias de confinamiento: 28,
Umbrales confinar/desconfinar:
(125000,5000)

vacaciones R0=1.6, 4 mejores resultados en: infectados_totales



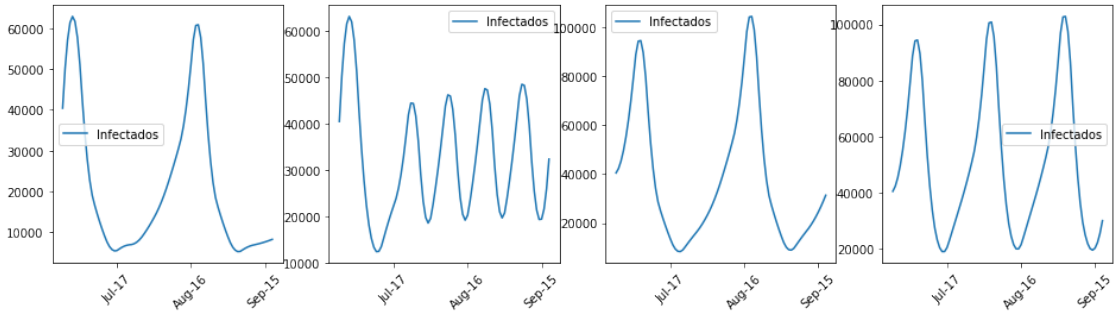

infectados: 0.49 mill., olas 3
dias de confinamiento: 36,
Umbrales confinar/desconfinar:
(62000,5000)

infectados: 0.57 mill., olas 3
dias de confinamiento: 31,
Umbrales confinar/desconfinar:
(62000,10000)

infectados: 0.84 mill., olas 2
dias de confinamiento: 34,
Umbrales confinar/desconfinar:
(125000,5000)

infectados: 0.84 mill., olas 2
dias de confinamiento: 32,
Umbrales confinar/desconfinar:
(125000,10000)

vacaciones R0=1.7, 4 mejores resultados en: infectados_totales



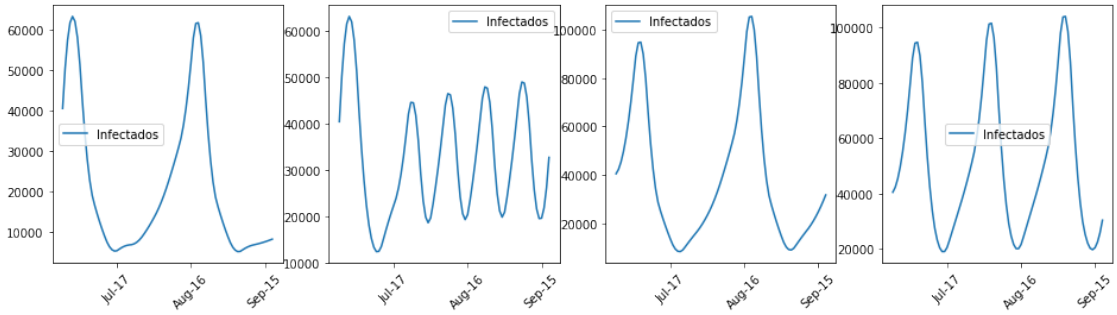

infectados: 0.38 mill., olas 2
dias de confinamiento: 33,
Umbrales confinar/desconfinar:
(62000,5000)

infectados: 0.58 mill., olas 5
dias de confinamiento: 39,
Umbrales confinar/desconfinar:
(62000,10000)

infectados: 0.69 mill., olas 2
dias de confinamiento: 32,
Umbrales confinar/desconfinar:
(125000,5000)

infectados: 0.90 mill., olas 3
dias de confinamiento: 32,
Umbrales confinar/desconfinar:
(125000,10000)

vacaciones R0=1.8, 4 mejores resultados en: infectados_totales



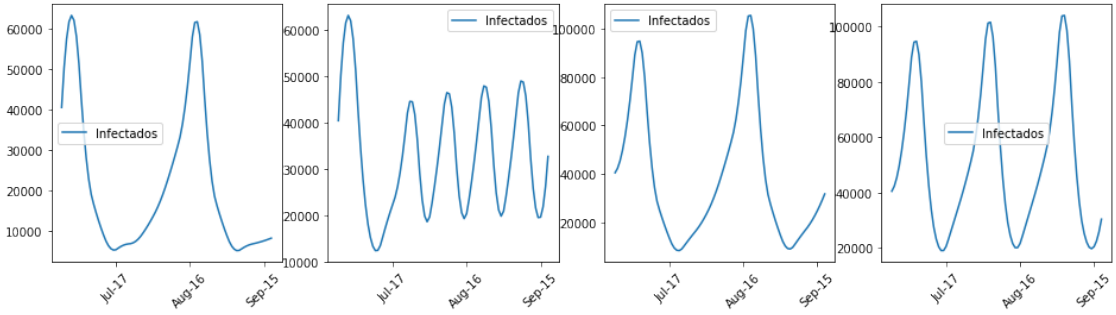

infectados: 0.39 mill., olas 2
dias de confinamiento: 33,
Umbrales confinar/desconfinar:
(62000,5000)

infectados: 0.59 mill., olas 5
dias de confinamiento: 39,
Umbrales confinar/desconfinar:
(62000,10000)

infectados: 0.69 mill., olas 2
dias de confinamiento: 32,
Umbrales confinar/desconfinar:
(125000,5000)

infectados: 0.91 mill., olas 3
dias de confinamiento: 32,
Umbrales confinar/desconfinar:
(125000,10000)

vacaciones R0=1.9, 4 mejores resultados en: infectados_totales



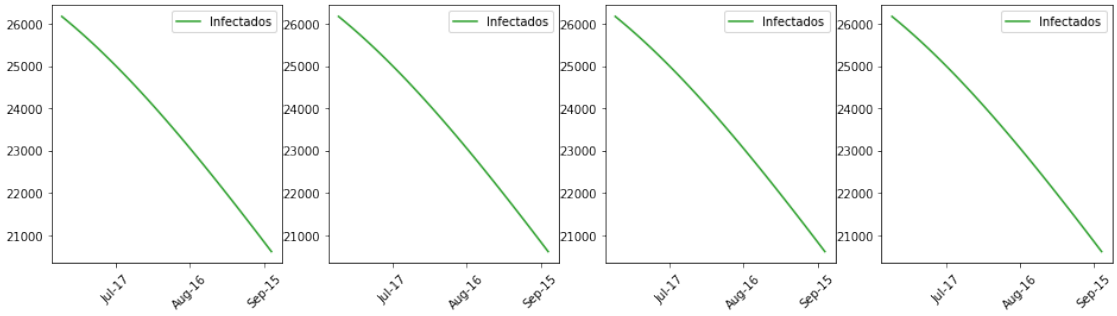

infectados: 0.39 mill., olas 2
dias de confinamiento: 33,
Umbrales confinar/desconfinar:
(62000,5000)

infectados: 0.59 mill., olas 5
dias de confinamiento: 39,
Umbrales confinar/desconfinar:
(62000,10000)

infectados: 0.69 mill., olas 2
dias de confinamiento: 32,
Umbrales confinar/desconfinar:
(125000,5000)

infectados: 0.91 mill., olas 3
dias de confinamiento: 32,
Umbrales confinar/desconfinar:
(125000,10000)

vacaciones R0=1.1, 4 mejores resultados en: dias_confinamiento



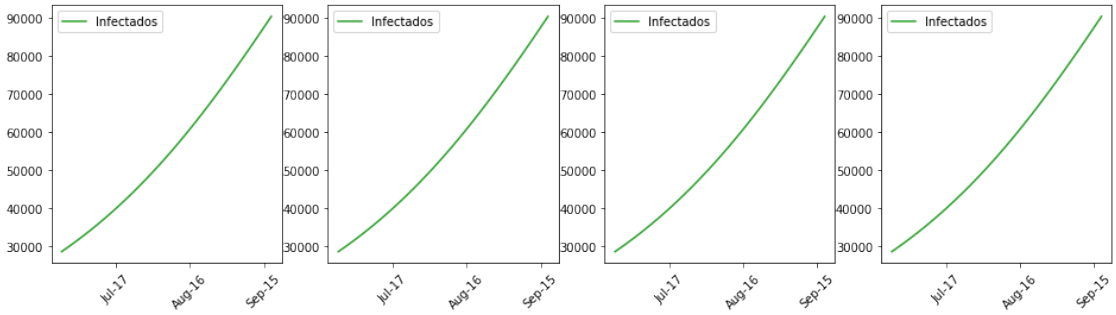

infectados: 0.42 mill., olas 0
dias de confinamiento: 0,
Umbrales confinar/desconfinar:
(1800000,40000)

infectados: 0.42 mill., olas 0
dias de confinamiento: 0,
Umbrales confinar/desconfinar:
(1800000,50000)

infectados: 0.42 mill., olas 0
dias de confinamiento: 0,
Umbrales confinar/desconfinar:
(1000000,40000)

infectados: 0.42 mill., olas 0
dias de confinamiento: 0,
Umbrales confinar/desconfinar:
(1000000,50000)

vacaciones R0=1.2, 4 mejores resultados en: dias_confinamiento



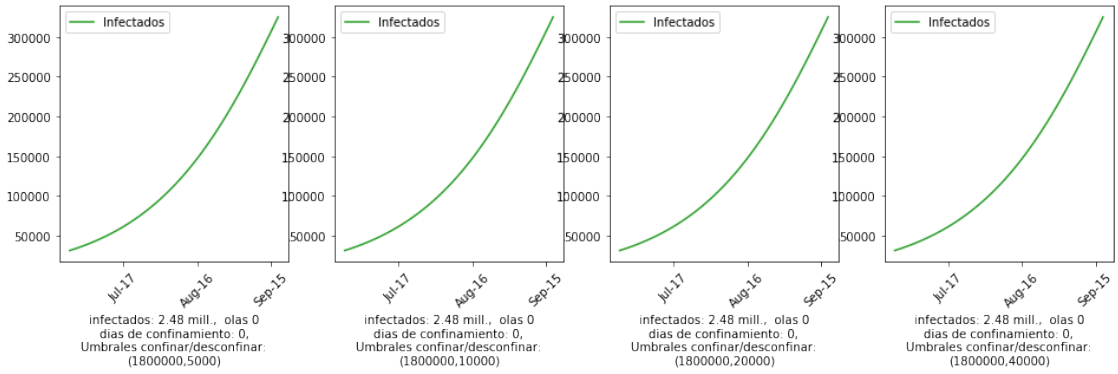

infectados: 1.00 mill., olas 0
dias de confinamiento: 0,
Umbrales confinar/desconfinar:
(1800000,5000)

infectados: 1.00 mill., olas 0
dias de confinamiento: 0,
Umbrales confinar/desconfinar:
(1800000,10000)

infectados: 1.00 mill., olas 0
dias de confinamiento: 0,
Umbrales confinar/desconfinar:
(1800000,20000)

infectados: 1.00 mill., olas 0
dias de confinamiento: 0,
Umbrales confinar/desconfinar:
(1800000,40000)

vacaciones R0=1.3, 4 mejores resultados en: dias_confinamiento



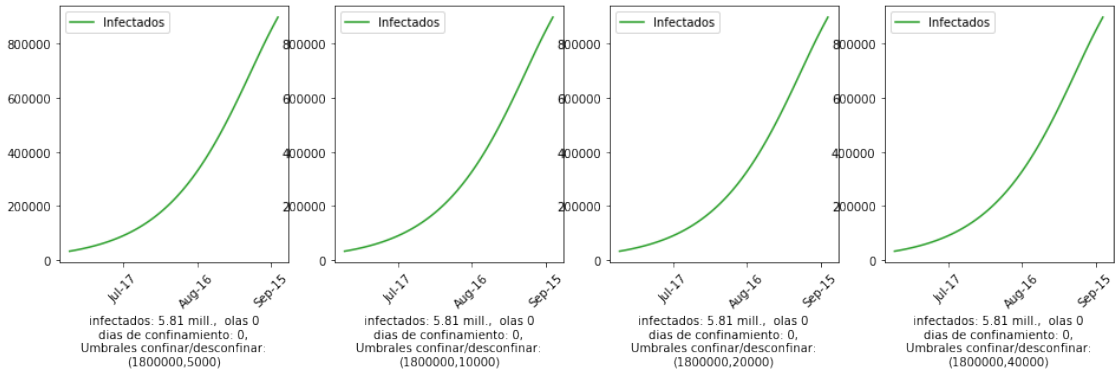

infectados: 2.48 mill., olas 0 dias de confinamiento: 0, Umbrales confinar/desconfinar: (1800000,5000)

infectados: 2.48 mill., olas 0 dias de confinamiento: 0, Umbrales confinar/desconfinar: (1800000,10000)

infectados: 2.48 mill., olas 0 dias de confinamiento: 0, Umbrales confinar/desconfinar: (1800000,20000)

infectados: 2.48 mill., olas 0 dias de confinamiento: 0, Umbrales confinar/desconfinar: (1800000,40000)

vacaciones R0=1.4, 4 mejores resultados en: dias_confinamiento



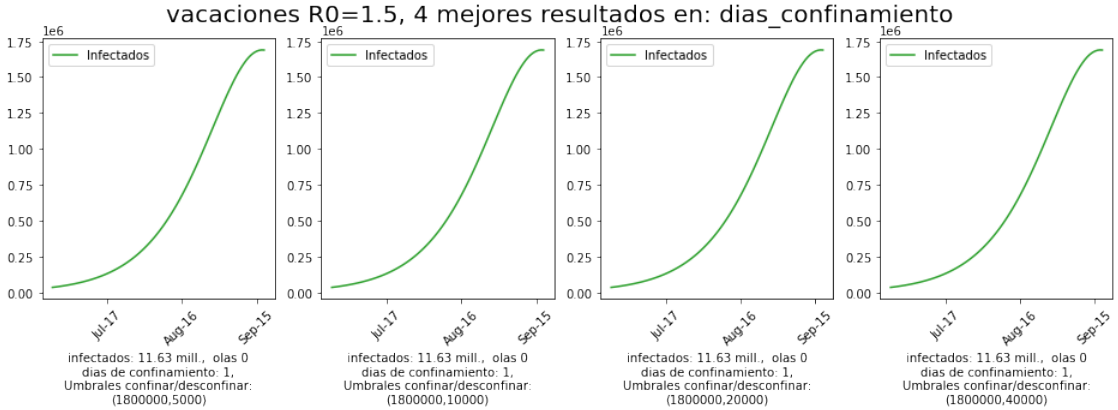

infectados: 5.81 mill., olas 0 dias de confinamiento: 0, Umbrales confinar/desconfinar: (1800000,5000)

infectados: 5.81 mill., olas 0 dias de confinamiento: 0, Umbrales confinar/desconfinar: (1800000,10000)

infectados: 5.81 mill., olas 0 dias de confinamiento: 0, Umbrales confinar/desconfinar: (1800000,20000)

infectados: 5.81 mill., olas 0 dias de confinamiento: 0, Umbrales confinar/desconfinar: (1800000,40000)

vacaciones R0=1.5, 4 mejores resultados en: dias_confinamiento



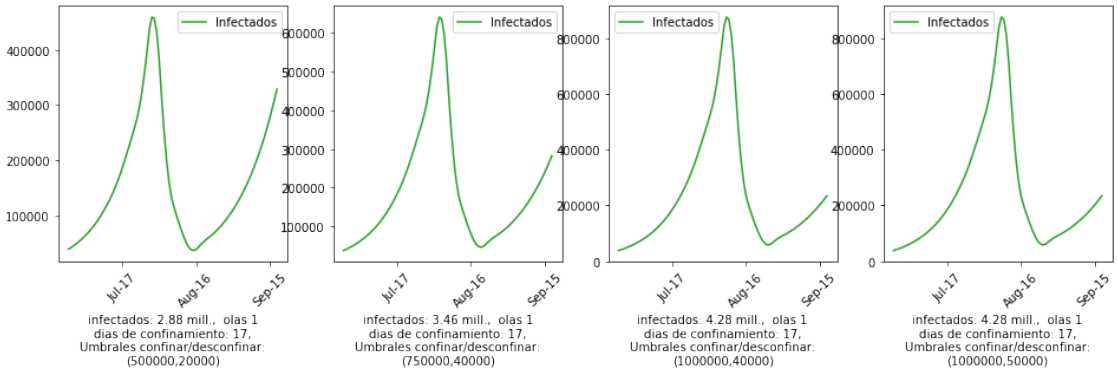

infectados: 11.63 mill., olas 0 dias de confinamiento: 1, Umbrales confinar/desconfinar: (1800000,5000)

infectados: 11.63 mill., olas 0 dias de confinamiento: 1, Umbrales confinar/desconfinar: (1800000,10000)

infectados: 11.63 mill., olas 0 dias de confinamiento: 1, Umbrales confinar/desconfinar: (1800000,20000)

infectados: 11.63 mill., olas 0 dias de confinamiento: 1, Umbrales confinar/desconfinar: (1800000,40000)

vacaciones R0=1.6, 4 mejores resultados en: dias_confinamiento



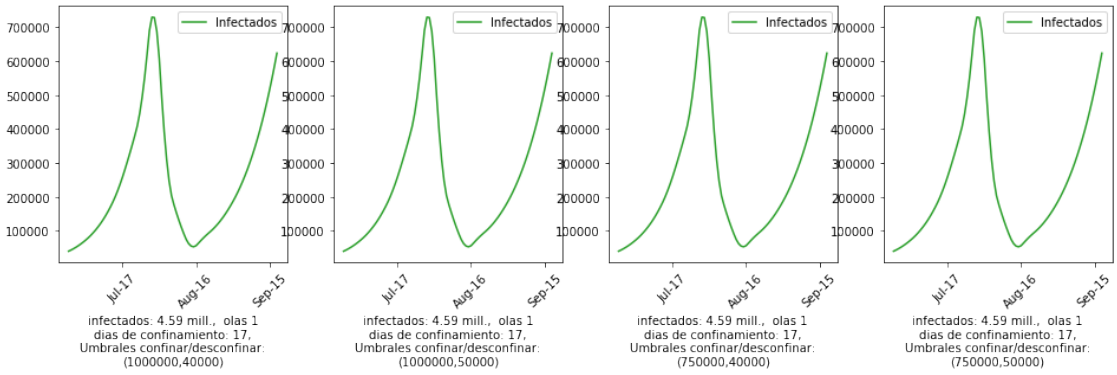

infectados: 2.88 mill., olas 1 dias de confinamiento: 17, Umbrales confinar/desconfinar: (500000,20000)

infectados: 3.46 mill., olas 1 dias de confinamiento: 17, Umbrales confinar/desconfinar: (750000,40000)

infectados: 4.28 mill., olas 1 dias de confinamiento: 17, Umbrales confinar/desconfinar: (1000000,40000)

infectados: 4.28 mill., olas 1 dias de confinamiento: 17, Umbrales confinar/desconfinar: (1000000,50000)

vacaciones R0=1.7, 4 mejores resultados en: dias_confinamiento



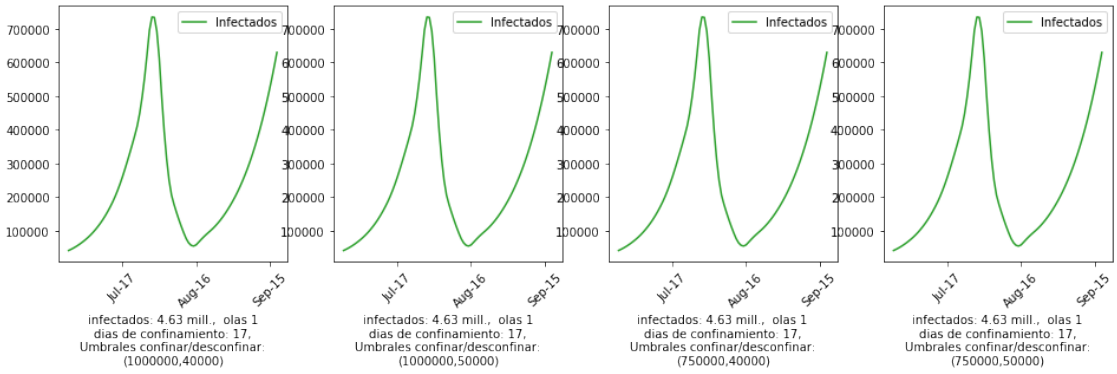

infectados: 4.59 mill., olas 1 dias de confinamiento: 17, Umbrales confinar/desconfinar: (1000000,40000)

infectados: 4.59 mill., olas 1 dias de confinamiento: 17, Umbrales confinar/desconfinar: (1000000,50000)

infectados: 4.59 mill., olas 1 dias de confinamiento: 17, Umbrales confinar/desconfinar: (750000,40000)

infectados: 4.59 mill., olas 1 dias de confinamiento: 17, Umbrales confinar/desconfinar: (750000,50000)

vacaciones R0=1.8, 4 mejores resultados en: dias_confinamiento



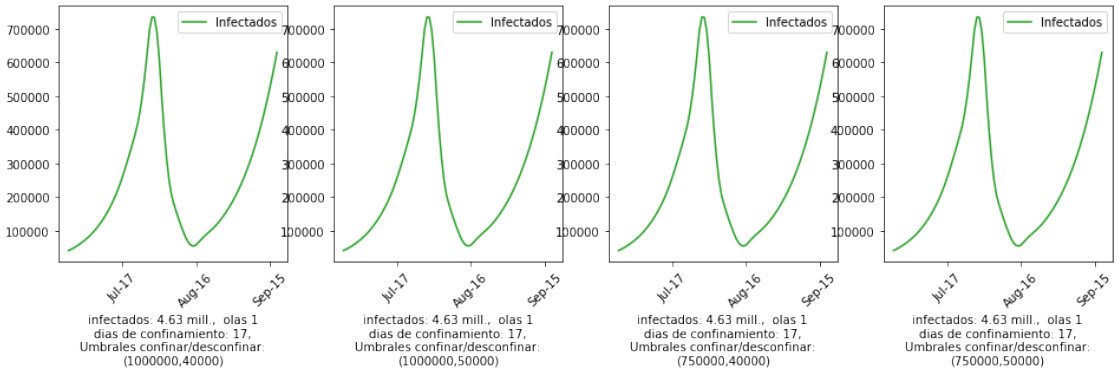

infectados: 4.63 mill., olas 1 dias de confinamiento: 17, Umbrales confinar/desconfinar: (1000000,40000)

infectados: 4.63 mill., olas 1 dias de confinamiento: 17, Umbrales confinar/desconfinar: (1000000,50000)

infectados: 4.63 mill., olas 1 dias de confinamiento: 17, Umbrales confinar/desconfinar: (750000,40000)

infectados: 4.63 mill., olas 1 dias de confinamiento: 17, Umbrales confinar/desconfinar: (750000,50000)

vacaciones R0=1.9, 4 mejores resultados en: dias_confinamiento



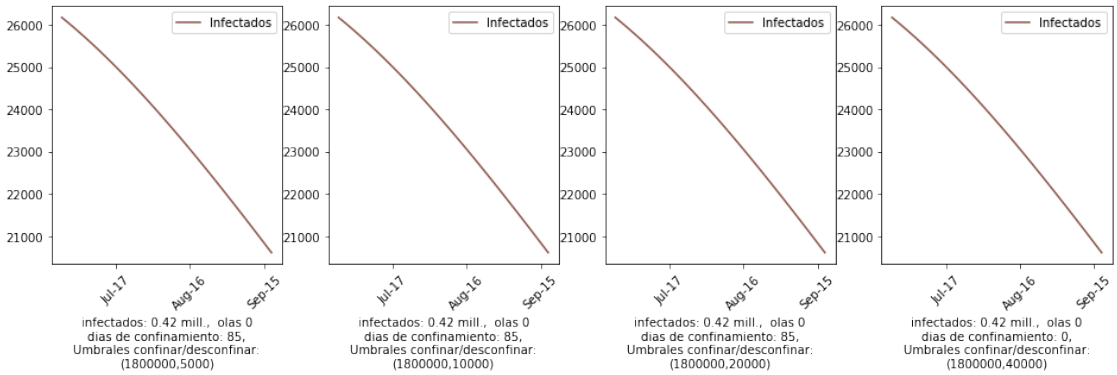

infectados: 4.63 mill., olas 1
dias de confinamiento: 17,
Umbrales confinar/desconfinar:
(1000000,40000)

infectados: 4.63 mill., olas 1
dias de confinamiento: 17,
Umbrales confinar/desconfinar:
(1000000,50000)

infectados: 4.63 mill., olas 1
dias de confinamiento: 17,
Umbrales confinar/desconfinar:
(750000,40000)

infectados: 4.63 mill., olas 1
dias de confinamiento: 17,
Umbrales confinar/desconfinar:
(750000,50000)

vacaciones R0=1.1, 4 mejores resultados en: numero_de_olas



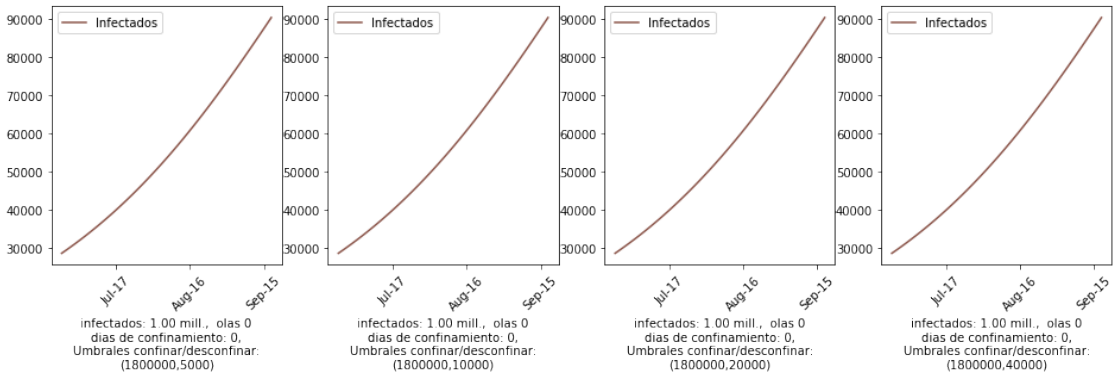

infectados: 0.42 mill., olas 0
dias de confinamiento: 85,
Umbrales confinar/desconfinar:
(1800000,5000)

infectados: 0.42 mill., olas 0
dias de confinamiento: 85,
Umbrales confinar/desconfinar:
(1800000,10000)

infectados: 0.42 mill., olas 0
dias de confinamiento: 85,
Umbrales confinar/desconfinar:
(1800000,20000)

infectados: 0.42 mill., olas 0
dias de confinamiento: 0,
Umbrales confinar/desconfinar:
(1800000,40000)

vacaciones R0=1.2, 4 mejores resultados en: numero_de_olas



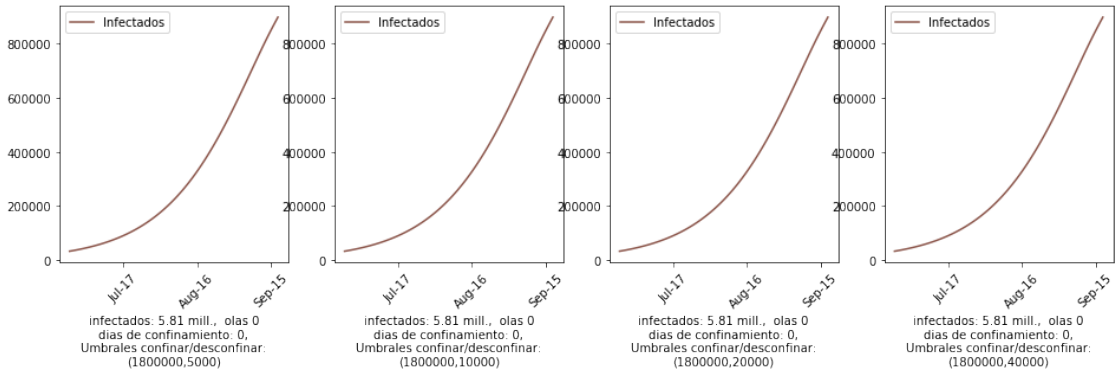

infectados: 1.00 mill., olas 0
dias de confinamiento: 0,
Umbrales confinar/desconfinar:
(1800000,5000)

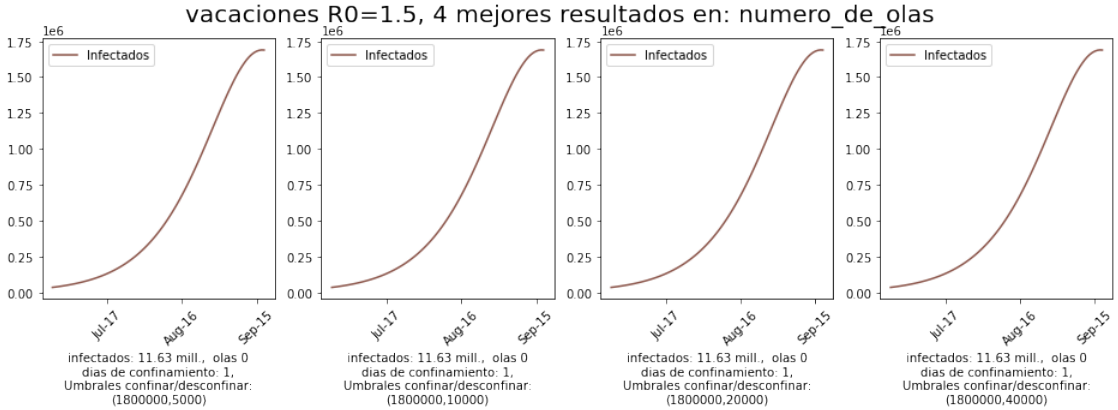
infectados: 1.00 mill., olas 0
dias de confinamiento: 0,
Umbrales confinar/desconfinar:
(1800000,10000)

infectados: 1.00 mill., olas 0
dias de confinamiento: 0,
Umbrales confinar/desconfinar:
(1800000,20000)

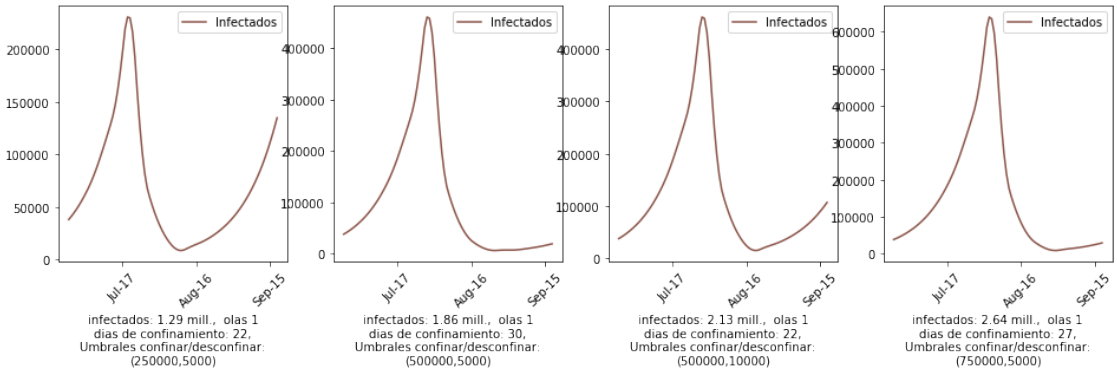
infectados: 1.00 mill., olas 0
dias de confinamiento: 0,
Umbrales confinar/desconfinar:
(1800000,40000)


vacaciones R0=1.3, 4 mejores resultados en: numero_de_olas
Infectados
300000
250000
200000
150000
100000
50000
Jul-17
Aug-16
Sep-15
infectados: 2.48 mill., olas 0
dias de confinamiento: 0,
Umbrales confinar/desconfinar:
(1800000,5000)
infectados: 2.48 mill., olas 0
dias de confinamiento: 0,
Umbrales confinar/desconfinar:
(1800000,10000)
infectados: 2.48 mill., olas 0
dias de confinamiento: 0,
Umbrales confinar/desconfinar:
(1800000,20000)
infectados: 2.48 mill., olas 0
dias de confinamiento: 0,
Umbrales confinar/desconfinar:
(1800000,40000)


vacaciones R0=1.4, 4 mejores resultados en: numero_de_olas
Infectados
800000
600000
400000
200000
0
Jul-17
Aug-16
Sep-15
infectados: 5.81 mill., olas 0
dias de confinamiento: 0,
Umbrales confinar/desconfinar:
(1800000,5000)
infectados: 5.81 mill., olas 0
dias de confinamiento: 0,
Umbrales confinar/desconfinar:
(1800000,10000)
infectados: 5.81 mill., olas 0
dias de confinamiento: 0,
Umbrales confinar/desconfinar:
(1800000,20000)
infectados: 5.81 mill., olas 0
dias de confinamiento: 0,
Umbrales confinar/desconfinar:
(1800000,40000)


vacaciones R0=1.5, 4 mejores resultados en: numero_de_olas
1e6
Infectados
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
Jul-17
Aug-16
Sep-15
infectados: 11.63 mill., olas 0
dias de confinamiento: 1,
Umbrales confinar/desconfinar:
(1800000,5000)
infectados: 11.63 mill., olas 0
dias de confinamiento: 1,
Umbrales confinar/desconfinar:
(1800000,10000)
infectados: 11.63 mill., olas 0
dias de confinamiento: 1,
Umbrales confinar/desconfinar:
(1800000,20000)
infectados: 11.63 mill., olas 0
dias de confinamiento: 1,
Umbrales confinar/desconfinar:
(1800000,40000)

vacaciones R0=1.6, 4 mejores resultados en: numero_de_olas



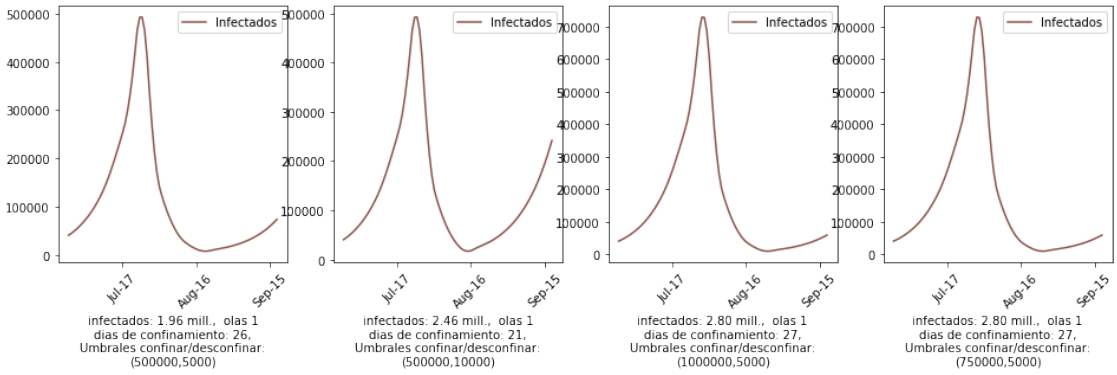

infectados: 1.29 mill., olas 1
dias de confinamiento: 22,
Umbrales confinar/desconfinar:
(250000,5000)

infectados: 1.86 mill., olas 1
dias de confinamiento: 30,
Umbrales confinar/desconfinar:
(500000,5000)

infectados: 2.13 mill., olas 1
dias de confinamiento: 22,
Umbrales confinar/desconfinar:
(500000,10000)

infectados: 2.64 mill., olas 1
dias de confinamiento: 27,
Umbrales confinar/desconfinar:
(750000,5000)

vacaciones R0=1.7, 4 mejores resultados en: numero_de_olas



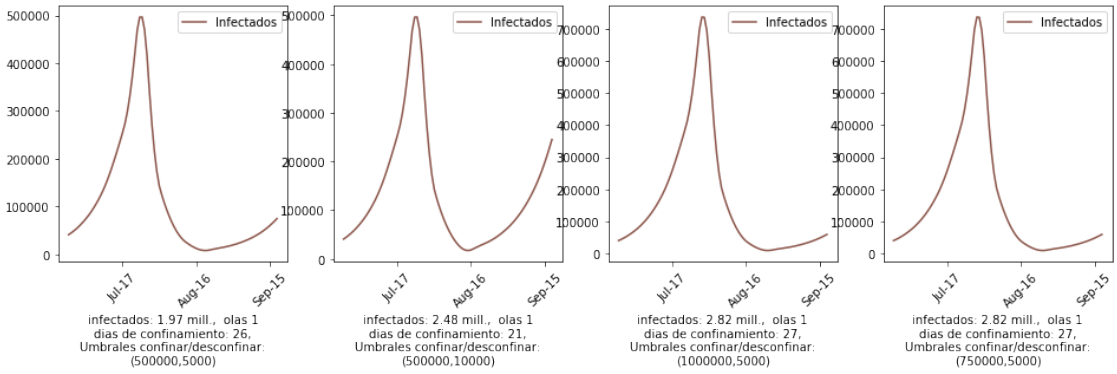

infectados: 1.96 mill., olas 1
dias de confinamiento: 26,
Umbrales confinar/desconfinar:
(500000,5000)

infectados: 2.46 mill., olas 1
dias de confinamiento: 21,
Umbrales confinar/desconfinar:
(500000,10000)

infectados: 2.80 mill., olas 1
dias de confinamiento: 27,
Umbrales confinar/desconfinar:
(1000000,5000)

infectados: 2.80 mill., olas 1
dias de confinamiento: 27,
Umbrales confinar/desconfinar:
(750000,5000)

vacaciones R0=1.8, 4 mejores resultados en: numero_de_olas

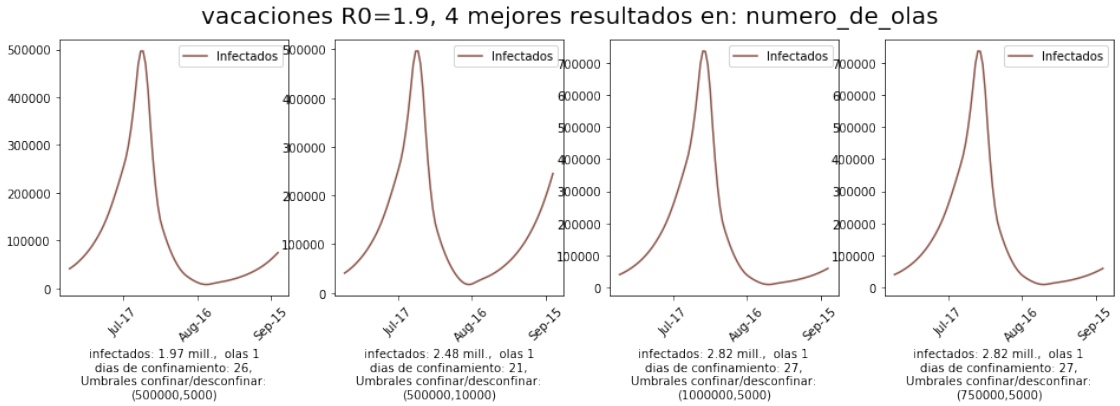



infectados: 1.97 mill., olas 1
dias de confinamiento: 26,
Umbrales confinar/desconfinar:
(500000,5000)

infectados: 2.48 mill., olas 1
dias de confinamiento: 21,
Umbrales confinar/desconfinar:
(500000,10000)

infectados: 2.82 mill., olas 1
dias de confinamiento: 27,
Umbrales confinar/desconfinar:
(1000000,5000)

infectados: 2.82 mill., olas 1
dias de confinamiento: 27,
Umbrales confinar/desconfinar:
(750000,5000)




### 1.5.1 Conclusiones del Escenario: vacaciones

Este modelo es muy sensible a las condiciones iniciales:

- Cuando empiezan a subir las temperaturas.
- El uso de mascarillas.
- La capacidad de detectar y aislar nuevos brotes.
- Cuantos casos importados tengamos.

Si se cumplen las condiciones, es posible que aguantasemos el verano sin confinamientos, si logramos mantener el R0<1.3.

Aún en el caso de que no lográramos mantener bajo el R0, si hicieramos confinamiento, y fueramos lo bastante previsores con los umbrales de confinamiento y desconfinamiento, podriamos mantener bajo el número de infectados - medio millón de infectados, unas 1200 muertes.

## 1.6 Hipotesis: Datos de Madrid

Esta hipotesis traslada los datos de R0 la comunidad de Madrid a toda España.

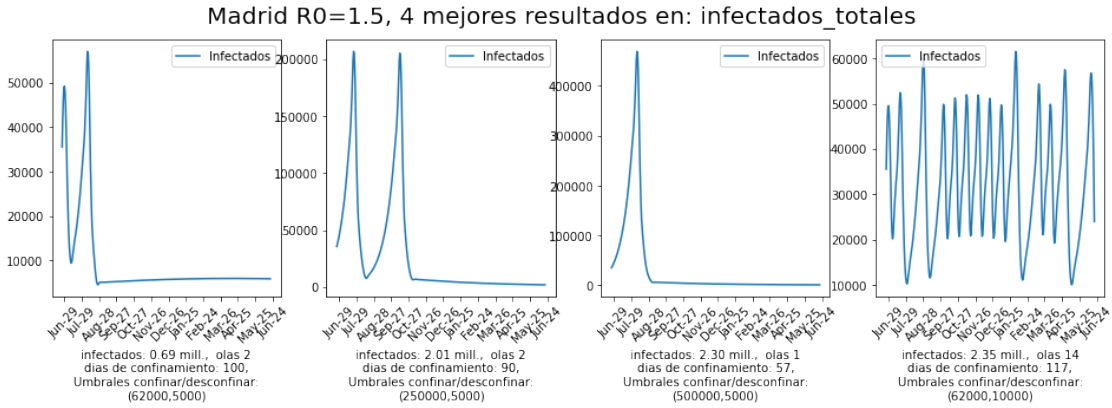
[14]:
```
# Caso Datos de Madrid
ESTIMACION_CASOS_ESPAÑA_EN_UNA_GENERACION_INFECCION=26400

array_parametros_estudio_vacaciones = [
 { "escenario"        : "Madrid" ,
  "descripcion"       : "Madrid R0=1.5" ,
  "R0_calor"          : 1.23  ,
  "SITUACION_INICIAL" : ESTIMACION_CASOS_ESPAÑA_EN_UNA_GENERACION_INFECCION,
  "R0"                : 1.4952 }
]
```

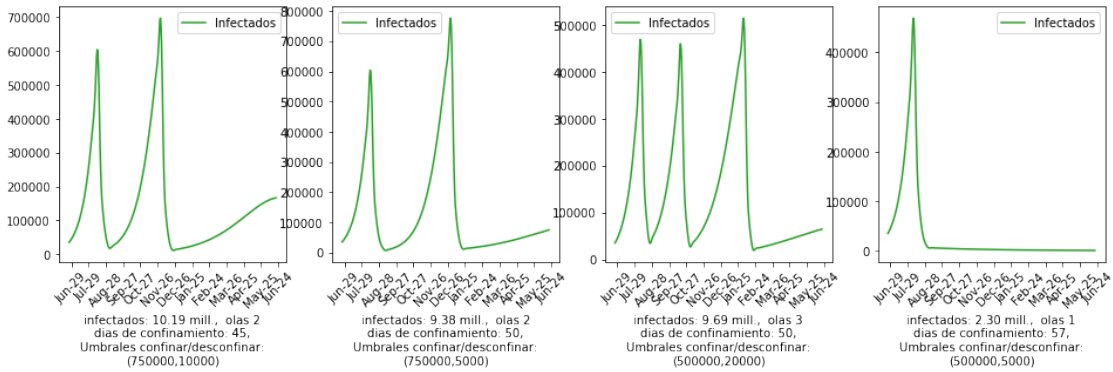
```
dict_default_values = {
    "R0_min" : 0.5,
    "FECHA_INICIAL_STR" : '2020-06-20'  ,
    "FECHA_FINAL_STR"   : '2021-06-20'  ,
    'GENERACIONES_SUBIDA' : 1 , 'GENERACIONES_BAJADA' : 1,
    'POBLACION_INICIAL_INFECTADA' : 4500000  ,
    "UMBRAL_DETECCION_BROTES"           :  7200                          ,
    "PORCENTAJE_DETECCION_BROTES"       :  .25                          ,
    "INCREMENTOS_SEMANAL_DESCONFINADOS" :  50                            ,
    "INCREMENTOS_SEMANAL_CONFINADOS"    :  20
    }


default_grid_thresolds ={"Umbral_max": [1800000,
                                        1000000,
                                         750000,
                                         500000,
                                         250000,
                                         125000,
                                         62000,
                                        ],
                         "Umbral_min": [ 10000,20000,40000,50000] }

madrid =␣
 ↪Generar_Datos_Conjunto_Escenarios(array_parametros_estudio_vacaciones,dict_default_values,"
Mostrar_Resultados_Conjunto_Escenarios(madrid)
```

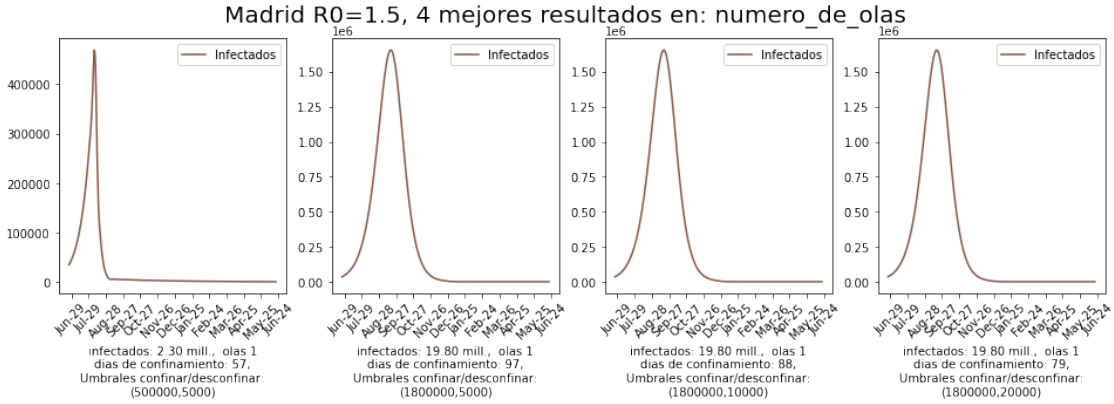
```
<IPython.core.display.HTML object>
```

```
HBox(children=(FloatProgress(value=1.0, bar_style='info', description='Procesando escenarios',
```

```
HBox(children=(FloatProgress(value=0.0, description='Madrid R0=1.5', max=35.0, style=ProgressSt
```




### 1.6.1 Conclusiones del Escenario: Madrid

Este escenario es interesante, porque muestra que si se consigue bajar los casos por debajo de cierto umbral, ocasiona que la detección de brotes sea capaz de prevenir nuevas olas.

Si se supera el umbral, como cuando se supera la masa crítica de un sistema, se produce una ola epídemica.

Lo interesante de este escenario es que sugiere que en algunas comunidades se podrían prevenir nuevas olas, pero en otras comunidades sean imposibles de evitar.

### 1.7 Hipótesis : Casos importados.

Esta hipotesis coge los datos de la comunidad de Madrid, y estudia la variación, al modificar el nº de casos importados.

Se pretende comprender el impacto de abrir las fronteras, y permitir la llegada de turistas.

[15]:
```
# Caso Datos Importados
ESTIMACION_CASOS_ESPAÑA_EN_UNA_GENERACION_INFECCION=26400

array_parametros_estudio_importacion = [
 { "escenario" : "Importacion" ,   "descripcion" : "Importacion R0=1.5,␣
 ↪importacion casos 50/semana" ,
    "INCREMENTOS_SEMANAL_DESCONFINADOS" : 50 , ␣
 ↪"INCREMENTOS_SEMANAL_CONFINADOS"    :  20    } ,
 { "escenario" : "Importacion" ,   "descripcion" : "Importacion R0=1.5,␣
 ↪importacion casos 100/semana" ,
  "INCREMENTOS_SEMANAL_DESCONFINADOS" : 100 ,  ␣
 ↪"INCREMENTOS_SEMANAL_CONFINADOS" : 40 ,   },
 { "escenario" : "Importacion" ,   "descripcion" : "Importacion R0=1.5,␣
 ↪importacion casos 500/semana" ,
    "INCREMENTOS_SEMANAL_DESCONFINADOS" : 500 , ␣
 ↪"INCREMENTOS_SEMANAL_CONFINADOS"    :  200  },
 { "escenario" : "Importacion" ,   "descripcion" : "Importacion R0=1.5,␣
 ↪importacion casos 1000/semana" ,
   "INCREMENTOS_SEMANAL_DESCONFINADOS" : 1000 ,  ␣
 ↪"INCREMENTOS_SEMANAL_CONFINADOS" : 400}
]



dict_default_values = {
    "R0_min" : 0.5,
    "FECHA_INICIAL_STR" : '2020-06-20'  ,
    "FECHA_FINAL_STR"   : '2021-06-20'  ,
    'GENERACIONES_SUBIDA' : 7 ,
    'GENERACIONES_BAJADA' : 7,
    'POBLACION_INICIAL_INFECTADA' : 4500000  ,
    "UMBRAL_DETECCION_BROTES"           :  7200                        ,
    "PORCENTAJE_DETECCION_BROTES"      :  .25                         ,
    "R0_calor"          : 1.23  ,
    "SITUACION_INICIAL" : ESTIMACION_CASOS_ESPAÑA_EN_UNA_GENERACION_INFECCION,
```

```
    "R0"   : 1.4952 ,
    }


importacion =␣
 ↪Generar_Datos_Conjunto_Escenarios(array_parametros_estudio_importacion,dict_default_values,
Mostrar_Resultados_Conjunto_Escenarios(importacion)
```

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

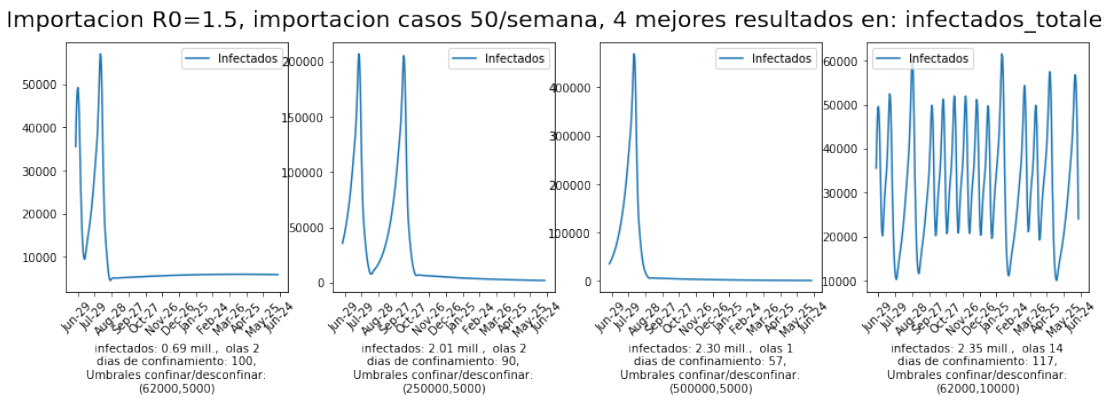
```
HBox(children=(FloatProgress(value=1.0, bar_style='info', description='Procesando escenarios',
```

```
HBox(children=(FloatProgress(value=0.0, description='Importacion R0=1.5, importacion casos 50/s
```

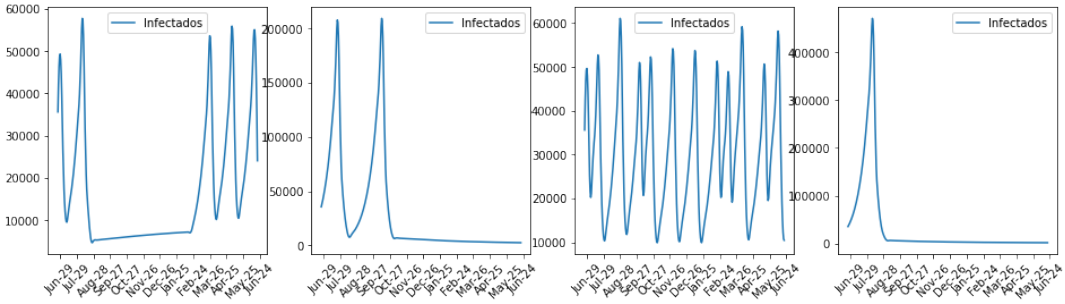
```
HBox(children=(FloatProgress(value=0.0, description='Importacion R0=1.5, importacion casos 100/
```

```
HBox(children=(FloatProgress(value=0.0, description='Importacion R0=1.5, importacion casos 500/
```

```
HBox(children=(FloatProgress(value=0.0, description='Importacion R0=1.5, importacion casos 1000
```



Importacion R0=1.5, importacion casos 100/semana, 4 mejores resultados en: infectados_totales



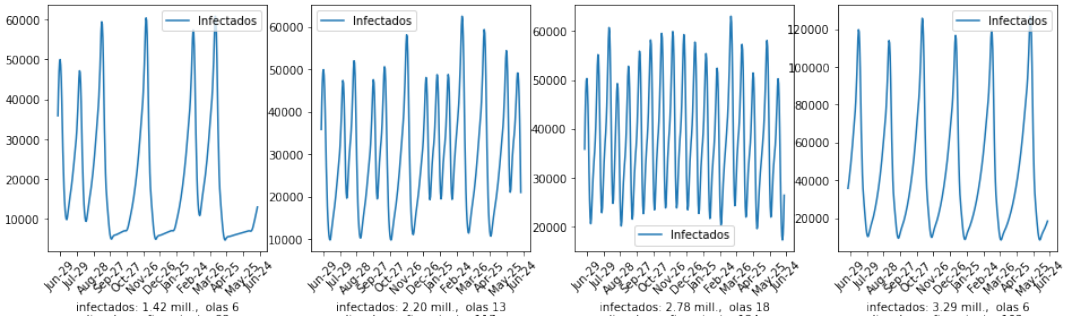

infectados: 1.22 mill., olas 5
dias de confinamiento: 64,
Umbrales confinar/desconfinar:
(62000,5000)

infectados: 2.06 mill., olas 2
dias de confinamiento: 104,
Umbrales confinar/desconfinar:
(250000,5000)

infectados: 2.18 mill., olas 12
dias de confinamiento: 116,
Umbrales confinar/desconfinar:
(62000,10000)

infectados: 2.33 mill., olas 1
dias de confinamiento: 65,
Umbrales confinar/desconfinar:
(500000,5000)

Importacion R0=1.5, importacion casos 500/semana, 4 mejores resultados en: infectados_totales



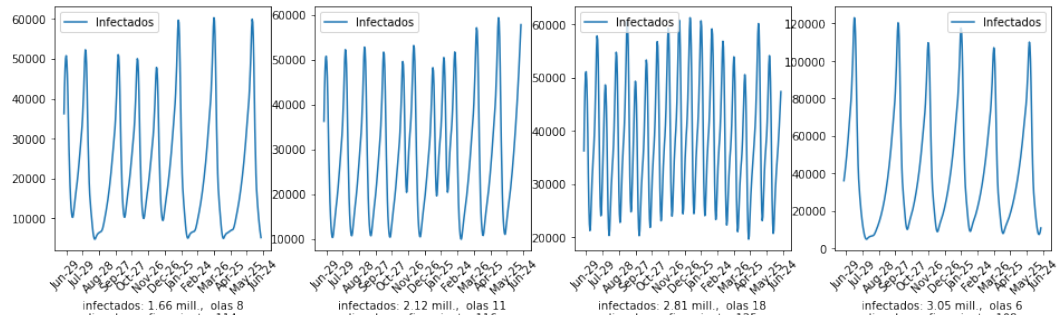

infectados: 1.42 mill., olas 6
dias de confinamiento: 92,
Umbrales confinar/desconfinar:
(62000,5000)

infectados: 2.20 mill., olas 13
dias de confinamiento: 117,
Umbrales confinar/desconfinar:
(62000,10000)

infectados: 2.78 mill., olas 18
dias de confinamiento: 124,
Umbrales confinar/desconfinar:
(62000,20000)

infectados: 3.29 mill., olas 6
dias de confinamiento: 102,
Umbrales confinar/desconfinar:
(125000,5000)

Importacion R0=1.5, importacion casos 1000/semana, 4 mejores resultados en: infectados_totales



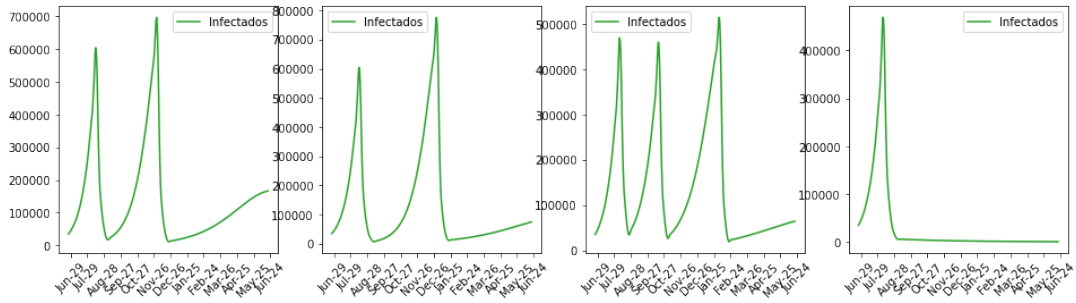

infectados: 1.66 mill., olas 8
dias de confinamiento: 114,
Umbrales confinar/desconfinar:
(62000,5000)

infectados: 2.12 mill., olas 11
dias de confinamiento: 116,
Umbrales confinar/desconfinar:
(62000,10000)

infectados: 2.81 mill., olas 18
dias de confinamiento: 125,
Umbrales confinar/desconfinar:
(62000,20000)

infectados: 3.05 mill., olas 6
dias de confinamiento: 108,
Umbrales confinar/desconfinar:
(125000,5000)

Importacion R0=1.5, importacion casos 50/semana, 4 mejores resultados en: dias_confinamiento



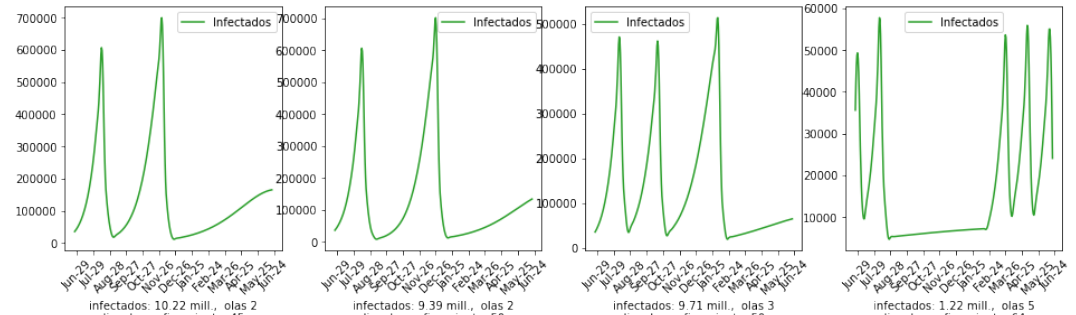

infectados: 10.19 mill., olas 2
dias de confinamiento: 45,
Umbrales confinar/desconfinar:
(750000,10000)

infectados: 9.38 mill., olas 2
dias de confinamiento: 50,
Umbrales confinar/desconfinar:
(750000,5000)

infectados: 9.69 mill., olas 3
dias de confinamiento: 50,
Umbrales confinar/desconfinar:
(500000,20000)

infectados: 2.30 mill., olas 1
dias de confinamiento: 57,
Umbrales confinar/desconfinar:
(500000,5000)

Importacion R0=1.5, importacion casos 100/semana, 4 mejores resultados en: dias_confinamiento



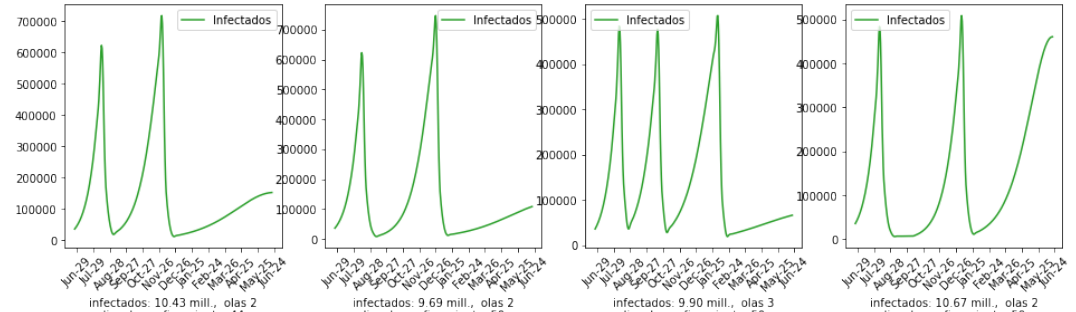

infectados: 10.22 mill., olas 2
dias de confinamiento: 45,
Umbrales confinar/desconfinar:
(750000,10000)

infectados: 9.39 mill., olas 2
dias de confinamiento: 50,
Umbrales confinar/desconfinar:
(750000,5000)

infectados: 9.71 mill., olas 3
dias de confinamiento: 50,
Umbrales confinar/desconfinar:
(500000,20000)

infectados: 1.22 mill., olas 5
dias de confinamiento: 64,
Umbrales confinar/desconfinar:
(62000,5000)

Importacion R0=1.5, importacion casos 500/semana, 4 mejores resultados en: dias_confinamiento



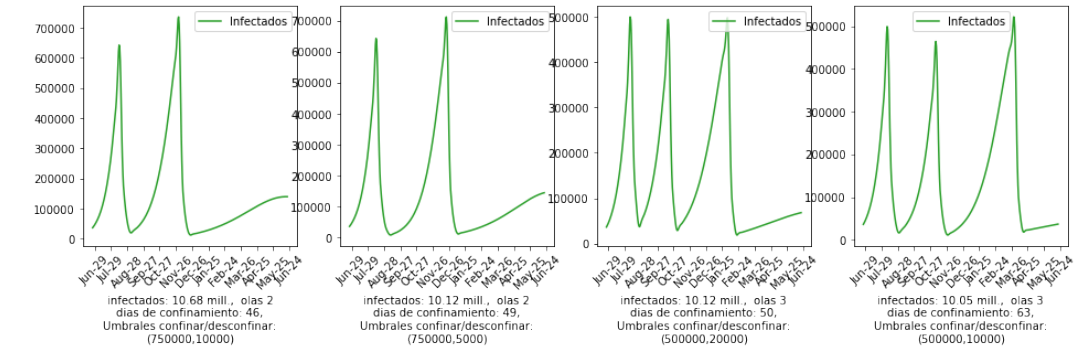

infectados: 10.43 mill., olas 2
dias de confinamiento: 44,
Umbrales confinar/desconfinar:
(750000,10000)

infectados: 9.69 mill., olas 2
dias de confinamiento: 50,
Umbrales confinar/desconfinar:
(750000,5000)

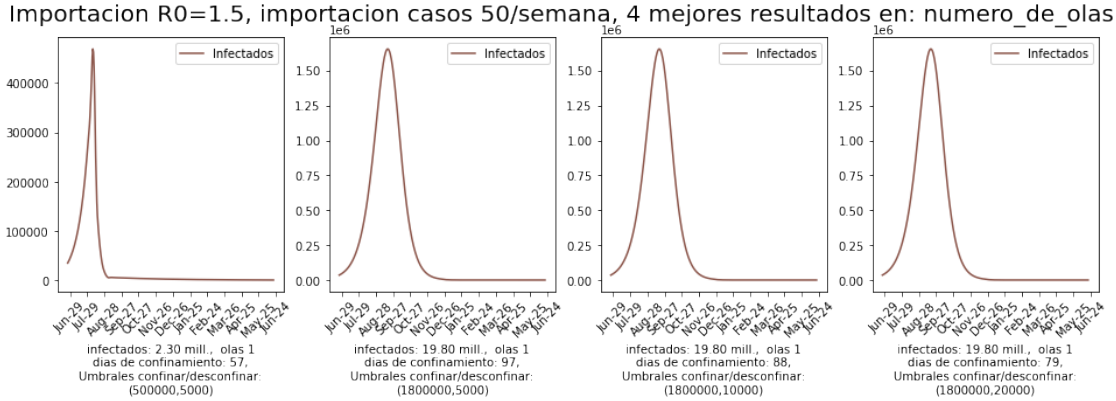
infectados: 9.90 mill., olas 3
dias de confinamiento: 50,
Umbrales confinar/desconfinar:
(500000,20000)

infectados: 10.67 mill., olas 2
dias de confinamiento: 59,
Umbrales confinar/desconfinar:
(500000,5000)

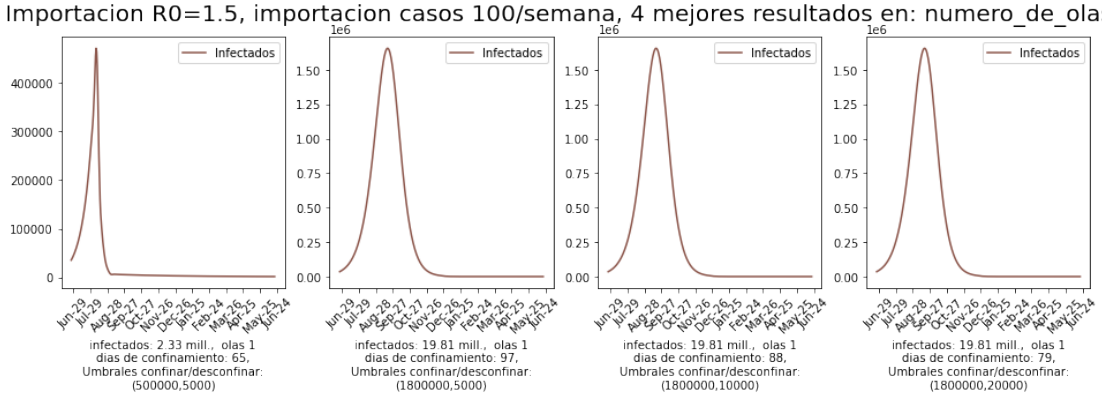
Importacion R0=1.5, importacion casos 1000/semana, 4 mejores resultados en: dias_confinamiento

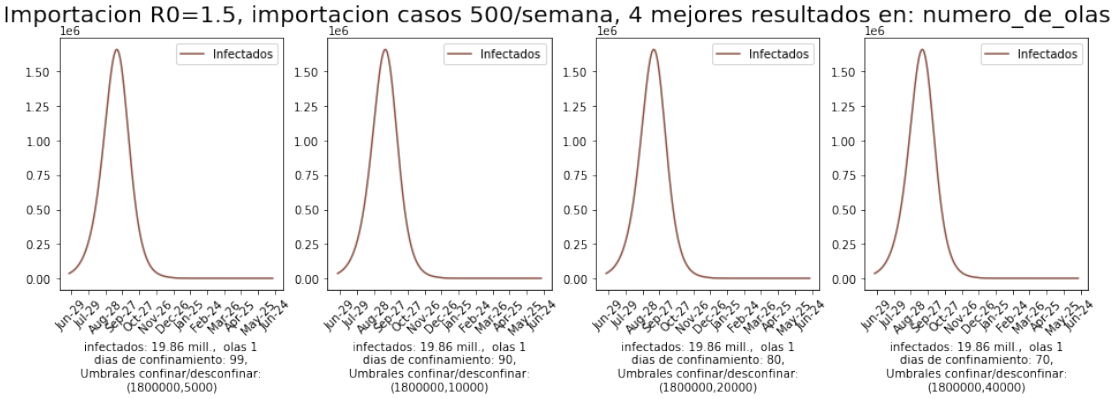


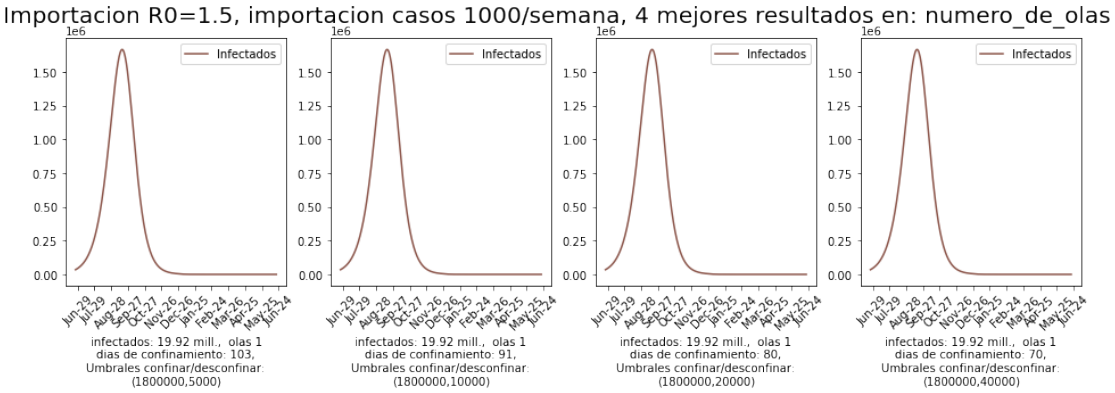

Importacion R0=1.5, importacion casos 50/semana, 4 mejores resultados en: numero_de_olas




Importacion R0=1.5, importacion casos 100/semana, 4 mejores resultados en: numero_de_olas






## 1.8 Conclusiones hipótesis casos importados.

No parece haber un gran impacto al subir los casos importados, a rasgos generales. Pero si puede influir en el nº de dias confinados - lo que a su vez si tiene impacto significativo en la economía.

Podía darse el caso de tener que elegir entre cerrar el pais para no importar casos, o cerrar el país por confinamiento, y en ambos casos sufre el sector turístico.

La conclusión mas obvia es que es necesario reforzar los mecanismos de detección en las fronteras, especialmente aeropuertos.

## 1.9 Datos agregados

### 1.9.1 Cogemos los datos anteriores, y los comparamos, para sacar conclusiones

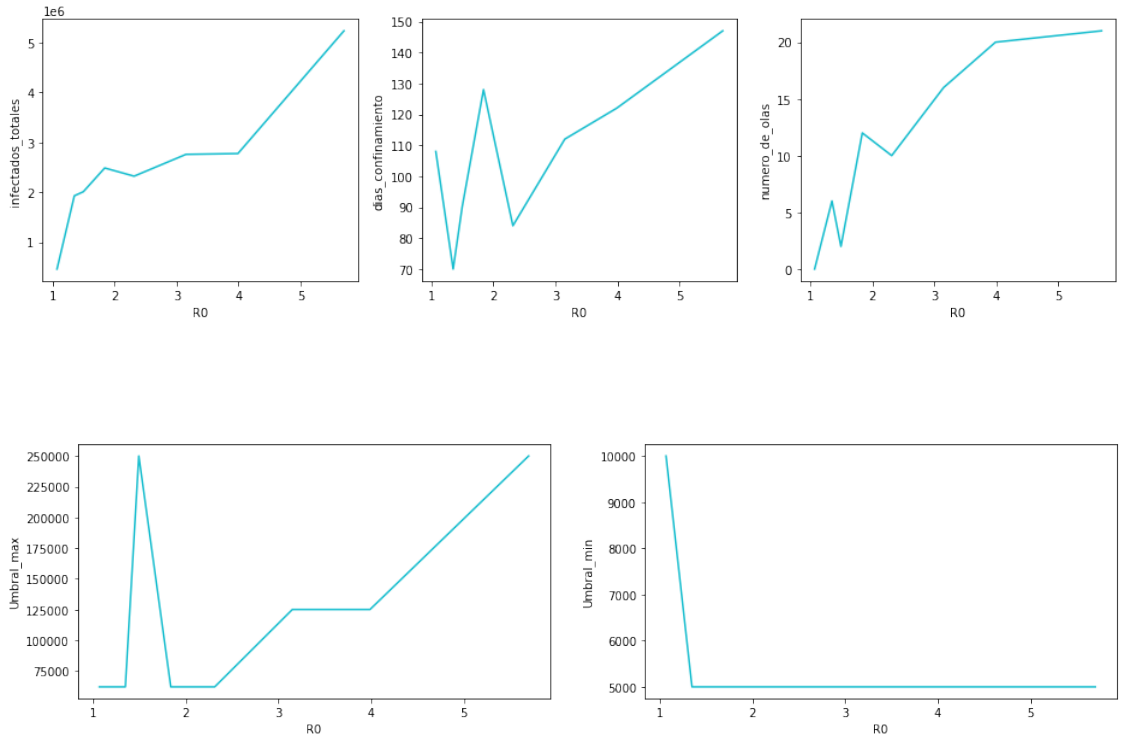
[16]:
```
# Nos quedamos solo con los mejores resultados
import janitor
df = pd.concat([pd.DataFrame(mascarillas),pd.DataFrame(madrid),pd.
 ↪DataFrame(airborne)])
df = df.drop('df', axis=1)
```

```
array_results = []
median_array = []
df_escenario=pd.DataFrame()
for escenario in df.descripcion.unique():
    df_escenario = df.filter_on(f"""descripcion == '{escenario}'"""  )
    criterio = ['infectados_totales', 'Umbral_max']
    criterio_order = [True, True]
    df_escenario.sort_values(by = criterio,ascending=criterio_order␣
 ↪,inplace=True)
    df_escenario=df_escenario.head(3)
   ␣
 ↪iter_dict=df_escenario[['dias_confinamiento','numero_de_olas','infectados_totales','R0','Um
 ↪median()
    iter_dict['escenario'] = escenario
    median_array.append(iter_dict)

    array_results.append(df_escenario)

df =pd.concat(array_results)
median_df = pd.DataFrame(median_array)

median_df = median_df.set_index('escenario')
median_df.sort_values(['R0'], inplace=True,ascending=[False])

#median_df
```

[17]:
```
import seaborn as sns
display(HTML("<h2>Miramos como varia los parámetros con el R0</h2>"))

fig, ax = plt.subplots(1, 3, figsize=(16, 4))
color='tab:cyan'
sns.lineplot(ax=ax.flatten()[0] , x="R0" ,  y="infectados_totales",␣
 ↪data=median_df,color=color )
sns.lineplot(ax=ax.flatten()[1] , x="R0" ,  y="dias_confinamiento",␣
 ↪data=median_df,color=color )
sns.lineplot(ax=ax.flatten()[2] , x="R0" ,  y="numero_de_olas", data=median_df␣
 ↪,color=color)

fig, ax = plt.subplots(1, 2, figsize=(16, 4))
sns.lineplot(ax=ax.flatten()[0] , x="R0" ,  y="Umbral_max", data=median_df␣
 ↪,color=color)
sns.lineplot(ax=ax.flatten()[1] , x="R0" ,  y="Umbral_min", data=median_df␣
 ↪,color=color)
```

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

[17]: <matplotlib.axes._subplots.AxesSubplot at 0x7f1b26707dd8>




[18]:
```
display(HTML("<h2>Cuadro resumen</h2>"))

median_df[[   'R0',
    'dias_confinamiento',
    'numero_de_olas',
    'infectados_totales','Umbral_max','Umbral_min']].style.format ({  c : "{:
 ↪20,.0f}" for c in [
    'dias_confinamiento',
    'numero_de_olas',
    'infectados_totales','Umbral_max','Umbral_min'] }).format ({ 'R0':  "{:20,.
 ↪3f}"}).background_gradient(cmap='Blues')
```

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

[18]: <pandas.io.formats.style.Styler at 0x7f1b2dd2c6a0>

## 1.10 Conclusiones

- Aquí si se ve como una buena política de confinamiento sería atajar los brotes cuanto antes - bajos umbrales de confinamiento y desconfinamiento-, aunque el umbral máximo va subiendo, conforme se incrementa R0.

- Son preferible muchas olas pequeñas, seguidas de confinamientos cortos, y espaciados igualmente en el tiempo.
- Con una buena política de umbrales confinamiento, parte sustancial del ”daño” se puede absorber en la dimensión de días de confinamiento, mas que en la de número de infectados.
- En casi todos los escenarios, menos en los mas catastróficos, los empeoramientos en el R0 redundan en más días confinados, pero no en más fallecimientos.
- Es de esperar que en algunas comunidades se podrían evitar nuevas olas, pero en otras comunidades será mucho más difícil.
- Una opción sería establecer un ”confinamiento mensual periódico”, p.e.: la última semana completa del mes y el fin de semana anterior, y que cada autonomía escogiera si ese mes se confina durante este periodo o no. Al hacer previsible los confinamientos, las empresas podrían planificar y anticiparse - los restaurantes no comprar género, las fábricas y hoteles no contratar turnos - durante esos días.
- Este confinamiento mensual periódico mitigaría el impacto económico .