In [1]:

```
import numpy as np
import matplotlib.pyplot as plt
#import random  #for selecting randomly from a distibution
import math

import scipy
import scipy.stats as stats #for finding the corresponsing number of successes in a bin
omial distribution given a probability
import scipy.special  #for the function comb= n choose k but faster than I can make it

import pandas as pd

import time #to see runtime

import xarray as xr #multidimensional data

from ipywidgets import interact #for slider on plot

import ipywidgets as widgets  #for play button slider

#to make a gif
import imageio
```

```
C:\Users\72kei\Anaconda3\lib\site-packages\distributed\utils.py:133: Runti
meWarning: Couldn't detect a suitable IP address for reaching '8.8.8.8', d
efaulting to '127.0.0.1': [WinError 10051] A socket operation was attempte
d to an unreachable network
  RuntimeWarning,
```

Population data

In [1]:

```
counties = ["CARLOW","CAVAN","CLARE","CORK","DONEGAL","DUBLIN","GALWAY","KERRY","KILDAR
E","KILKENNY","LAOIS","LEITRIM","LIMERICK","LONGFORD","LOUTH","MAYO","MEATH","MONAGHAN"
,"OFFALY","ROSCOMMON","SLIGO","TIPPERARY","WATERFORD","WESTMEATH","WEXFORD","WICKLOW"]

countiesDictionary = dict(zip(counties, range(0,len(counties))))

countyPops = [56932,
76176,
118817,
542868,
159192,
1347359,
258058,
147707,
222504,
99232,
84697,
32044,
194899,
40873,
128884,
103507,
195044,
61386,
77961,
64544,
65535,
159553,
116176,
88770,
149722,
142425]


#This data has been removed as it is private

data
```

```
---------------------------------------------------------------------------
-
NameError                                 Traceback (most recent call las
t)
<ipython-input-1-eb4797be051e> in <module>
     33 #data = pd.read_csv("./Data/CountyPop.csv")  #initial populations
 are all susceptible
     34
---> 35 data

NameError: name 'data' is not defined
```

above is the census data for commuting, with a few extra rows of information on the bottom

row 27. is the total population of that county

row 28 gives the factor we need to scale up the declared travel numbers by to have the whole population declared as travels and existing in our simulation

In [3]:

```
#get relevant data
S0= data.copy()  #does deep copy by default


S0 = S0.iloc[0:26,0:26] #note slice finishes 1 before final number
#also note that we need to assign the new thing, since this opperation doesn't work in
 place
#but if we had shallow copied the data, then updating S like this would update the data
(kind of?! only the bits that match)

#scale
for i in range(26):
     S0.iloc[i] = S0.iloc[i]*data.iloc[28,i] #apply can work on our specified axis

# #S['CAVAN'] = S['CAVAN'].apply(np.sqrt)  #it needs the title of the column, doesn't t
ake [1] if it has a name
# #S.iloc[1] = S.iloc[1].apply(np.sqrt) #but this works


#cast to ints
S0=S0.astype('int32')

S0

print("total population of this simulation: ",S0.sum().sum())
```

```
total population of this simulation:  4736321
```

The Model

In [4]:

```
def TimestatSIRmodel(initialInfected=(3,3,5),days=300, r0 = 2.2, recoveryPeriod = 5, se
ed=2):

    ####
    #Set up arrays and parameters
    ####


    #Initialise big xarrays (enough to fill in for every day of the simulation)
    S = xr.DataArray( data=np.zeros((len(counties), len(counties), days)),
    dims=["From","To","day"]).astype("int")  #initialising with the right size and with
named dimensions

    #Input fixed population numbers
    S[:,:,0] = S0  #set day 0 to have the right values


    #debugging, checking how long parts take to run
    timeList = [0]*5
    t0 = time.time()

    #tStart=time.time()



    #to get reporducable results
    np.random.seed(seed)



    I = xr.DataArray( data=np.zeros((len(counties), len(counties), days)),
    dims=["From","To","day"])
    Inew = xr.DataArray( data=np.zeros((len(counties), len(counties), days)),
    dims=["From","To","day"])
    R = xr.DataArray( data=np.zeros((len(counties), len(counties), days)),
    dims=["From","To","day"])



    ####
    #Account for the initial infected
    ####

    i=initialInfected[0]
    j=initialInfected[1]
    initI=initialInfected[2]

    #translate from string to integer
    if(type(i)==str):
        i=countiesDictionary[i]
    if(type(j)==str):
        j=countiesDictionary[j]

    I[i][j][0] = initI  #user can input all these parameters

    #these people will recover after the recovery period
    Inew[i][j][0] = initI
```

```
    InfectionProb = [0]*len(counties)
    IndividualThreshold = [0]*len(counties)

    regionInfectionProb = np.zeros((len(counties),len(counties)))  #numpy lists are ver
y fast with for loops so I use that structure here.


    if(type(r0)==list): #then interpolate to make it the right length
        newx = np.linspace(0,len(r0),days) #the x values we want to interpolate at
        r0 = np.interp(newx,range(len(r0)),r0)




    #I can track where get's the most infected and how many people at one time that is.
and what day
    Imax = initI
    imax = i
    jmax = j
    dmax = 0


    peopleInCounty = [0]*len(counties)  #average number of people in county. (same ever
y day)
    for i in range(0,len(counties)):
        peopleInCounty[i] = float(0.5*(S[i,:,0].sum() + S[:,i,0].sum()))

    #risk of infection in a half day
    if(type(r0)!=list): #then we only need to set this once
        for i in range(0,len(counties)):
            InfectionProb[i] = r0/(2*recoveryPeriod*(peopleInCounty[i]))


    IPopVisible = [0]*len(counties)   #the total infected we can see from this county
 (number we are EXPOSED to)

    #this is slight overkill since there is usually only 1 place infected at the start
    for i in range(0,len(counties)):
        IPopVisible[i] = float(0.5*((I[i,:,0].sum() + I[:,i,0].sum())))


    #timeList[0]+=time.time()-tStart


    #each iteration is one day
    d=1
    run=True
    while(run):


        #risk of infection in a half day - that's what the 2*recivPeriod is for
        if(type(r0)==list):
            for i in range(0,len(counties)):
                 InfectionProb[i] = r0[d-1]/(2*recoveryPeriod*(peopleInCounty[i]))

        #timeList[0]+=time.time()-tStart


        if(d%100==0): #for sainty during long trial runs
            print("Day ",d)
```

```
#             #chance of being infected - IN A HALF DAY in county i
#             IndividualThreshold[i] = stats.binom.sf(k=0,n=int(IPop),p=InfectionProb
[i])  #this value is small when p is very small


        IndividualThreshold = list(stats.binom.sf(k=0,n=[ int(IPopVisible[i]) for i in
range(len(counties)) ],p=InfectionProb))
            #we want the integral up to some number of successes to get the probability
            #.sf goes from right to left. It is 1-cummulative function.

            #to be honest, I'm still confused why k=0 works and k=1 doesn't. but after
 lots of testing that's what works
            #there are lots of issues with k>0 since n can easily be 0







        #commuters only go between two places so that's not too complicated - I figured
out how to share risk between more places in a day so I could expand with more data


        #works with chance OF being infected - OLD
        #prob=1-((math.sqrt(1-IndividualThreshold[i]))*(math.sqrt(1-IndividualThreshold
[j])))

        tStart=time.time()

        #works with chance OF being infected
        #we get a symmetric matrix and I'm assigning the values like this because I thi
ng I'm really smart, but readability is probably worth more than the 0 time this way wi
ll likely save
        for i in range(0,len(counties)):
            regionInfectionProb[i][i]=IndividualThreshold[i]+IndividualThreshold[i]-Ind
ividualThreshold[i]*IndividualThreshold[i]
            for j in range(0,i):
                regionInfectionProb[j][i]=regionInfectionProb[i][j]=IndividualThreshold
[j]+IndividualThreshold[i]-IndividualThreshold[i]*IndividualThreshold[j]

        #Let's try some matrix multiplication!
        #regionInfectionProb = np.array([IndividualThreshold]*len(counties)) + np.array
([IndividualThreshold]*len(counties)).T - np.dot(np.diag(IndividualThreshold),np.array
([IndividualThreshold]*len(counties)))
        #This is clearly less readable and harder to understand, and I can't save much
 time since the for loops are fast now
        #It's actually slower! numpy can certainly optimize a for loop that's for sure




        Inew[:,:,d] = stats.binom.rvs(n=S[:,:,d-1],p=regionInfectionProb)

        #update all in a batch, without for loops
        S[:,:,d] = S[:,:,d-1]-Inew[:,:,d]


        if(d>=recoveryPeriod):
```

```
            I[:,:,d] = I[:,:,d-1]+ Inew[:,:,d] - Inew[:,:,d-recoveryPeriod]

            R[:,:,d] = R[:,:,d-1] + Inew[:,:,d-recoveryPeriod]

        #or not if it's not been enough days to have anyone recover (this logic is just
to avoid calling bad array entries)
        else:
            I[:,:,d] = I[:,:,d-1]+ Inew[:,:,d]

            R[:,:,d] = R[:,:,d-1]


        timeList[4]+=time.time()-tStart


        tStart=time.time()


        #just open the dataset once - This change genuinely made this step 100 times fa
ster 1sec to 0.01sec
        Inumpy = np.array(I[:,:,d])
        for i in range(0,len(counties)):
#             IPopVisible[i] = float(0.5*((I[i,:,d].sum() + I[:,i,d].sum())))
            currentSum = 0
            for j in range(0,len(counties)):
                currentSum += Inumpy[i][j] + Inumpy[j][i]

            IPopVisible[i] = 0.5*currentSum


        newMax = int(I[:,:,d].max().max())
        if(newMax > Imax):
            Imax = newMax
            loc = np.array(I[:,:,d]).argmax()
            imax = loc//len(counties)
            jmax = loc%len(counties)
            dmax = d




            #check if the simulation can be finished.
            #no S left or no I left


        if(S[:,:,d-1].max()<=0):
            print("Day ",d)
            print("No more susceptible -- break")

            #I actually only need to find one positive value to know I can continue

            run=False  #I don't need this trigger anymore, I can change back to a for l
oop and use break
            break   #this only breaks out of first for loop,

#             #delete debugging!!
#             #just for saftey I'll put this in once
#             if(testQuantity<0 or int(testQuantity)!=testQuantity):  #test for fractio
ns or negative errors
#                 print("Susceptible number Error -- break")
#                 print("i,j",b//len(counties),b%len(counties),"Susceptible=",testQuant
```

```
ity)
#                 d=days+1
#                 break   #this only breaks out of first for loop, so I set d high enou
gh to break out of outer loop


        #I actually only need to find one positive value to know I can continue
        if(I[:,:,d-1].max()<=0):
            print("Day ",d)
            print("No more infected -- break")

            run=False
            break

        #else all is good - continue simulation

        #delete debugging!!
        #just for saftey I'll put this in once
#         if(testQuantity<0 or int(testQuantity)!=testQuantity):  #test for fractions o
r negative errors
#             print("Infected number Error -- break")
#             print("i,j",b//len(counties),b%len(counties),"Infected=",testQuantity)
#             run=False
#             break   #this only breaks out of first for loop, so I set d high enough t
o break out of outer loop


        d+=1
        if(d>=days):
            run=False


    print("Complete on Day ",d)
    print("Number of infected left = ", int(I[:,:,d-1].sum().sum()))

    print("Max Concurant Infected: ",Imax,"   Location: (",counties[imax],",",counties[
jmax],")    Day: ",dmax)



    t1 = time.time()
    print("Simulation Run Time = ",round(t1-t0,3),"seconds\n")

    return [S,I,R,timeList,d]
```

Running the simulation

In [5]:

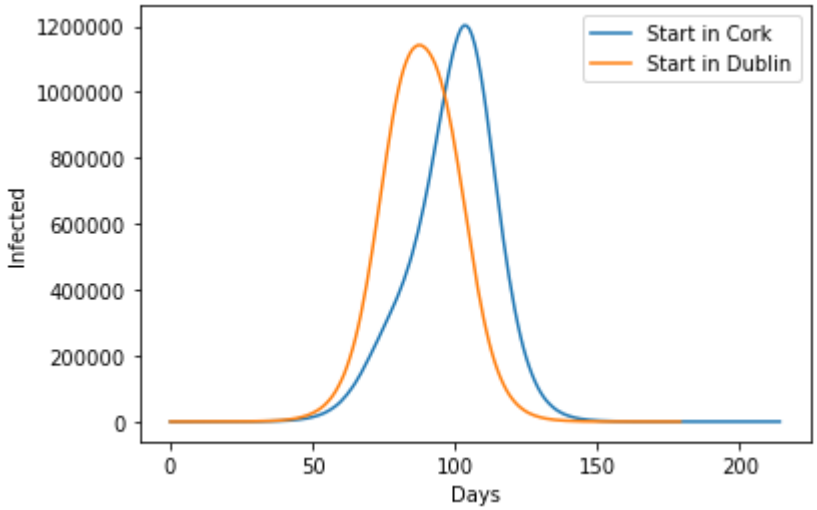
```
setseed=2
r0=2.2#[2.2, 2.2, 1, 2.2, 2.2]
recoveryPeriod=10
initI=5
d=300

#simulation starting 5 infected in cork
[SC,IC,RC,timeC,dC] = TimestatSIRmodel(initialInfected=['CORK','CORK',initI]    , r0=r0
,recoveryPeriod=recoveryPeriod,seed=setseed,days=d)

#note: timeC is timing functions for debbuging purposes


#simulation starting 5 infected in dublin
[SD,ID,RD,timeD,dD] = TimestatSIRmodel(initialInfected=['DUBLIN','DUBLIN',initI], r0=r0
,recoveryPeriod=recoveryPeriod,seed=setseed,days=d)
```

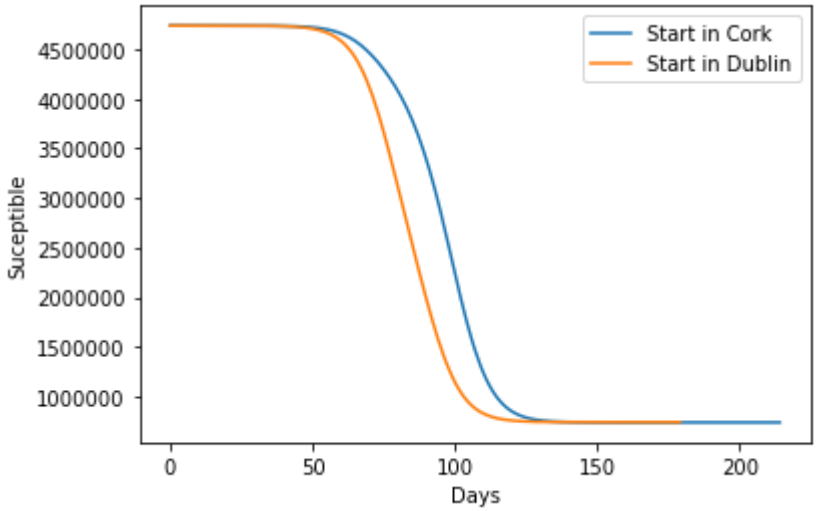
```
Day  100
Day  200
Day  215
No more infected -- break
Complete on Day  215
Number of infected left =  0
Max Concurant Infected:  439064    Location: ( DUBLIN , DUBLIN )    Day:
104
Simulation Run Time =  0.907 seconds

Day  100
Day  180
No more infected -- break
Complete on Day  180
Number of infected left =  0
Max Concurant Infected:  425017    Location: ( DUBLIN , DUBLIN )    Day:
80
Simulation Run Time =  0.725 seconds
```

Plotting the results

In [6]:

```
#%matplotlib inline
#need to use inline after using notebook

xc = [a for a in range(dC)]

totIC = [IC[:,:,dd].sum() for dd in range(dC)]
totSC = [SC[:,:,dd].sum() for dd in range(dC)]

p1=plt.plot(xc,totIC)


xd = [a for a in range(dD)]

totID = [ID[:,:,dd].sum() for dd in range(dD)]
totSD = [SD[:,:,dd].sum() for dd in range(dD)]


p2=plt.plot(xd,totID)

plt.xlabel("Days")
plt.ylabel("Infected")

plt.legend((p1[0], p2[0]), ('Start in Cork', 'Start in Dublin'))


#plt.xlim([0,30])

plt.show()


f
p3=plt.plot(xc,totSC)
p4=plt.plot(xd,totSD)

plt.xlabel("Days")
plt.ylabel("Suceptible")

#plt.ylim([0,100000])
#plt.xlim([30,50])

plt.legend((p3[0], p4[0]), ('Start in Cork', 'Start in Dublin'))

plt.show()

if(type(r0)==list):
    plt.plot(range(len(r0)),r0)
    plt.ylabel("r0")
```





Graphic with anotations

Put scatter dots in the right locations

In [6]:

```
img=plt.imread('./Images/Ireland Provinces.PNG')

#pixels
rightPixels=img.shape[0]
bottomPixels=img.shape[1]


#Geographical coords from google maps
bottomLeft=(51.270287, -10.783876)  #also these go: height, width

topRight = (55.465438, -4.965573)

span = (topRight[0]-bottomLeft[0],topRight[1]-bottomLeft[1])
print("span in degrees",span)

pixelRatio = (img.shape[0]/span[0],img.shape[1]/span[1])


countyCoords=[(52.732467, -6.844899),
              (53.983735, -7.281119),
              (52.898447, -9.028253),
              (51.990030, -8.688295),
              (54.903197, -7.993100),
              (53.379928, -6.267481),
              (53.361234, -8.695694),
              (52.100215, -9.671062),
              (53.221554, -6.807605),
              (52.567254, -7.211916),
              (53.007222, -7.371768),
              (54.097182, -7.945507),
              (52.510807, -8.819750),(53.731579, -7.717827),(53.918029, -6.469238),
              (53.936494, -9.341559),(53.677753, -6.712309),(54.150256, -6.903442),
              (53.287879, -7.646237),(53.752231, -8.300431),(54.154912, -8.621456),
              (52.610155, -7.928906),(52.201831, -7.597639),
              (53.532884, -7.408961),(52.516568, -6.525961),(53.026465, -6.334704)]


#transform from lat long to pixels
trasnformedCC = [(-(bottomLeft[1]-x[1])*pixelRatio[1],(topRight[0]-x[0])*pixelRatio[0])
for x in countyCoords]
```

```
span in degrees (4.195150999999996, 5.818302999999999)
```

In [8]:

```
%matplotlib notebook
#can't move slider with arrow keys - does do annotations

#%matplotlib inline
#can move slider with arrow keys - no annotations

#%matplotlib widget

#from ipywidgets import interact




@interact(day=(0,dC,5))
def plot(day=1):

    def update_annot(ind):

        pos = sc.get_offsets()[ind["ind"][0]]
        annot.xy = pos

        skeleton = "{} \n"*26
        skeleton = "Travel To : Number Infected \n" + skeleton
        text = skeleton.format(*[counties[x] + ":" + str(int(IC[ind["ind"],x,day])) for
x in range(len(counties))])

        #practice formating
        #text = "{}, {} \n {}, {} \n {}, {} ".format(*[x for x in range(6)])  #what a c
ool little trick. star is the "unpacking" operator making a list (*(a,b)) into (a,b)
        #just gives the number and letter
#        text = "{}, {}".format(" ".join(list(map(str,ind["ind"]))),
#                               " ".join([names[n] for n in ind["ind"]]))


        annot.set_text(text)
        annot.get_bbox_patch().set_facecolor('white')   #could set coulour by colour of
the point  #cmap(norm(c[ind["ind"][0]])))
        annot.get_bbox_patch().set_alpha(1) #transpanrency


    def hover(event):
        vis = annot.get_visible()
        if event.inaxes == ax:
            cont, ind = sc.contains(event)
            if cont:
                update_annot(ind)
                annot.set_visible(True)
                fig.canvas.draw_idle()
            else:
                if vis:
                    annot.set_visible(False)
                    fig.canvas.draw_idle()


    fig, ax = plt.subplots(figsize=(10,10))

    img=plt.imread('./Images/Ireland Provinces.PNG')
    plt.imshow(img)
```

```
    #visuals for the data points
    size=[100*np.log10(IC[x,:,day].sum()) for x in range(26)]
    #color gradient
    c=[np.log10(IC[x,:,day].sum()) for x in range(26)]
    #absolute range for colour scale
    norm=plt.Normalize(0, 6)  #10**6 is a million

    sc = plt.scatter([x[0] for x in trasnformedCC],[x[1] for x in trasnformedCC],s=size
,cmap="cool",c=c,norm=norm)#np.arange(26))




    for i, txt in enumerate(counties):
        ax.annotate(txt, (trasnformedCC[i][0], trasnformedCC[i][1]))


    annot = ax.annotate("", xy=(0,0),
                        xytext=(20,20), #where the annotaion should appear
                        textcoords="offset points",
                        bbox=dict(boxstyle="round", fc="w")
                        #you can have an arrow to the point if you want#,arrowprops=dic
t(arrowstyle="->")
                        )
    annot.set_visible(False)


    names = np.array(counties)


    cbar = plt.colorbar()
    cbar.set_label('log10 of number of Infected', rotation=270,size=15, labelpad=25)

    fig.canvas.mpl_connect("motion_notify_event", hover)


    plt.show()
```

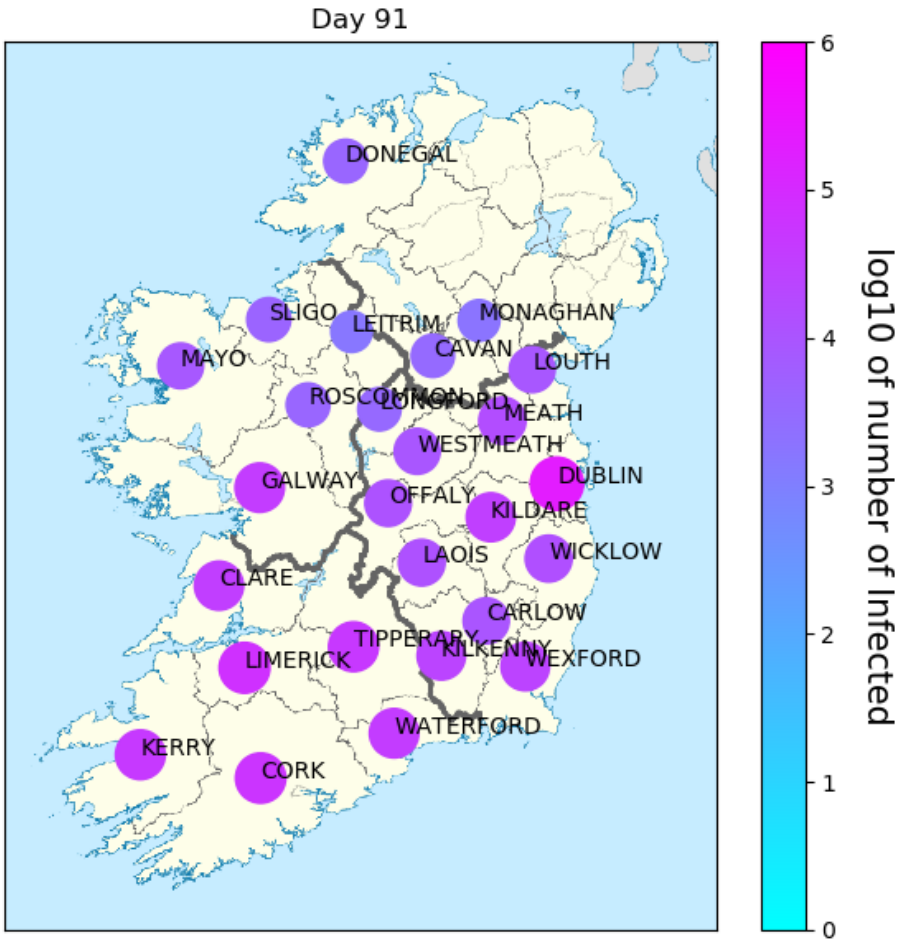
Plot With a play button - Working

In [18]:

```
%matplotlib notebook

fig,ax=plt.subplots(figsize=(7,7))


whiteSpace = [0,1,2,3,-1,1,0,1,1,0,3,2,1,-1,2,3,2,0,3,-2,4,-1,-1,0,0,1]

def plot(day=1):

    #print("hey : " ,day)

    #for some reason it only works when these functions are defined within the plot fun
ction
    def update_annot(ind):

        pos = sc.get_offsets()[ind["ind"][0]]
        annot.xy = pos

#         skeleton = "{} \n"*26
#         skeleton = "Travel To : Number Infected \n" + skeleton
#         text = skeleton.format(*[counties[x] + ":" + str(int(IC[ind["ind"],x,day])) f
or x in range(len(counties))])

        skeleton = "{} \n"*len(counties)
        #start = r'$\textcolor{blue}{From}$' + " {}\n".format(counties[ind["ind"][0]])
        start = 'From ' + " {}\n".format(counties[ind["ind"][0]])
        skeleton = "Travel To : Number Infected \n" + skeleton

        #I can't allign the whitespace porperly - I feel it might be impossible
        #text = skeleton.format(*[counties[x] + ":" + ' '*(20 + whiteSpace[x] - len(cou
nties[x])) + str(int(IC[ind["ind"][0],x,day])) for x in range(len(counties))])
        text = skeleton.format(*[counties[x] + ":" + ' '*4 + str(int(IC[ind["ind"][0],x
,day])) for x in range(len(counties))])


        text = start + text + "\nTotal: {}".format(str(int(IC[ind["ind"],:,day].sum
()))))

        #practice formating
        #text = "{}, {} \n {}, {} \n {}, {} ".format(*[x for x in range(6)])  #what a c
ool little trick. star is the "unpacking" operator making a list (*(a,b)) into (a,b)
        #just gives the number and letter
#         text = "{}, {}".format(" ".join(list(map(str,ind["ind"]))),
#                                " ".join([names[n] for n in ind["ind"]]))


        annot.set_text(text)
        annot.get_bbox_patch().set_facecolor('white')   #could set coulour by colour of
the point  #cmap(norm(c[ind["ind"][0]])))
        annot.get_bbox_patch().set_alpha(1) #transpanrency


    def hover(event):
        vis = annot.get_visible()
        if event.inaxes == ax:
            cont, ind = sc.contains(event)
            if cont:
                update_annot(ind)
                annot.set_visible(True)
```

```
                fig.canvas.draw_idle()
            else:
                if vis:
                    annot.set_visible(False)
                    fig.canvas.draw_idle()


    #fig, ax = plt.subplots(figsize=(10,10))

    img=plt.imread('./Images/Ireland Provinces.PNG')
    plt.imshow(img)

    #visuals for the data points
    size=[100*np.log10(IC[x,:,day].sum()) for x in range(26)]
    #color gradient
    c=[np.log10(IC[x,:,day].sum()) for x in range(26)]
    #absolute range for colour scale
    norm=plt.Normalize(0, 6)  #10**6 is a million

    sc = plt.scatter([x[0] for x in trasnformedCC],[x[1] for x in trasnformedCC],s=size
,cmap="cool",c=c,norm=norm)#np.arange(26))




    for i, txt in enumerate(counties):
        ax.annotate(txt, (trasnformedCC[i][0], trasnformedCC[i][1]))


    annot = ax.annotate("", xy=(0,0),  #the point to be annotated  #this is updated lat
er
                        xytext=(-120,700), #where the annotaion should appear
                        #textcoords="offset points", #forces the annotation to appear r
elative to xy
                        bbox=dict(boxstyle="round", fc="w")
                        #you can have an arrow to the point if you want#,arrowprops=dic
t(arrowstyle="->")
                        )
    annot.set_visible(False)


    names = np.array(counties)


#     cbar = plt.colorbar()
#     cbar.set_label('log10 of number of Infected', rotation=270,size=15, labelpad=25)

    fig.canvas.mpl_connect("motion_notify_event", hover)

    #title of plot
    plt.title("Day "+str(day))

    #don't show tickmarks or numbers
    ax.axes.xaxis.set_visible(False)
    ax.axes.yaxis.set_visible(False)

    plt.show()

    #return sc
```

```
#plot zeroth day
plot(0)


#play button and slider settings
play = widgets.Play(
    value=0,
    min=1,
    max=dC,
    step=1,
    interval=100,
    description="Press play",
    disabled=False
)


slider = widgets.IntSlider(min=0,max=dC)
widgets.jslink((play, 'value'), (slider, 'value'))  #need this - breaks slider if we do
n't have it
slider_withPlay=widgets.HBox([play, slider])


#int_range = widgets.IntSlider()

output2 = widgets.Output()  #this is like the object that our on value change function
 can see

#display(slider, output2)  #this is the slider without play buttons
display(slider_withPlay)#, output2)


def on_value_change(change):
    with output2:
        plt.cla()  #get's rid of old points but not the whole figure

        plot(change['new'])

#only need to draw colour bar once
cbar = plt.colorbar()
cbar.set_label('log10 of number of Infected', rotation=270,size=15, labelpad=25)

slider.observe(on_value_change, names='value')
```


Day 91
DONEGAL
SLIGO
LEITRIM
MONAGHAN
CAVAN
MAYO
LOUTH
ROSCOMMON
LONGFORD
MEATH
WESTMEATH
GALWAY
DUBLIN
OFFALY
KILDARE
LAOIS
WICKLOW
CLARE
CARLOW
TIPPERARY
KILKENNY
LIMERICK
WEXFORD
WATERFORD
KERRY
CORK
log10 of number of Infected
0
1
2
3
4
5
6

ideas to add

- fix the layout of the annotations
- add in an option to look at the evolution of R or S populations
- Only remove and redraw scatter points to save processing power and memory

-add a colour to the county i i to distinguish it quickly - doesn't work really

In [26]:

```
print("\033[91m  hey hey\033[0m hey")
```

```
  hey hey hey
```

More functionality and generality

In [35]:

```
from functools import partial


#I'm super annoyed that I have to make it this complicated to have a button increment a
number
class Counter:
    def __init__(self, initial=1): #start on 1 - 0=S 1=I 2=R 3=S0=population
        self.value = initial

    def increment(self):
        self.value = (self.value + 1) %4
        #return self.value

    def __iter__(self, sentinal=False):
        return iter(self.increment, sentinal)

    def value(self):
        return self.value



def interactiveDisplay(data):

    [S,I,R,dfin] = data

    cd = Counter() #cd is an object of my Counter class
    #cd = currentDisplay #meaning, we display infected info - 0=S 1=I 2=R 3=S0=populati
on

    %matplotlib notebook

    fig,ax=plt.subplots(figsize=(7,7))


    def plot(day,cd):

        #print("hey : " ,day)

        #for some reason it only works when these functions are defined within the plot
function
        def update_annot(ind):

            pos = sc.get_offsets()[ind["ind"][0]]
            annot.xy = pos


            #stich together the string for the annotation
            skeleton = "{} \n"*len(counties)
            start = 'From ' + " {}\n".format(counties[ind["ind"][0]])

            if(cd==0):
                header = "Travel To : Number Susceptible \n"
            if(cd==1):
                header = "Travel To : Number Infected \n"
            if(cd==2):
                header = "Travel To : Number Recovered \n"
            if(cd==3):
                header = "Travel To : Total Population \n"
```

```
            skeleton = header + skeleton

            #S0 is S at day=0
            if(cd==3):
                text = skeleton.format(*[counties[x] + ":" + '_'*4 + str(int(data[0][in
d["ind"][0],x,0])) for x in range(len(counties))])
                text = start + text + "\nTotal: {}".format(str(int(data[0][ind["ind"
],:,0].sum())))

            else:
                text = skeleton.format(*[counties[x] + ":" + '_'*4 + str(int(data[cd][i
nd["ind"][0],x,day])) for x in range(len(counties))])
                text = start + text + "\nTotal: {}".format(str(int(data[cd][ind["ind"
],:,day].sum())))


            annot.set_text(text)
            annot.get_bbox_patch().set_facecolor('white')   #could set coulour by colou
r of the point  #cmap(norm(c[ind["ind"][0]]))
            annot.get_bbox_patch().set_alpha(1) #transpanrency


        def hover(event):
            vis = annot.get_visible()
            if event.inaxes == ax:
                cont, ind = sc.contains(event)
                if cont:
                    update_annot(ind)
                    annot.set_visible(True)
                    fig.canvas.draw_idle()
                else:
                    if vis:
                        annot.set_visible(False)
                        fig.canvas.draw_idle()


        #fig, ax = plt.subplots(figsize=(10,10)) #only works outside the definition of
 this function

        img=plt.imread('./Images/Ireland Provinces.PNG')
        plt.imshow(img)

        #title of plot
        plt.title("Day "+str(day))

        #don't show tickmarks or numbers
        ax.axes.xaxis.set_visible(False)
        ax.axes.yaxis.set_visible(False)


        #visuals for the data points
        size=[100*np.log10(data[1][x,:,day].sum()) for x in range(26)]
        #color gradient
        c=[np.log10(data[1][x,:,day].sum()) for x in range(26)]
        #absolute range for colour scale
        norm=plt.Normalize(0, 6)  #10**6 is a million

        sc = plt.scatter([x[0] for x in trasnformedCC],[x[1] for x in trasnformedCC],s=
size,cmap="cool",c=c,norm=norm)#np.arange(26))
```

```
        for i, txt in enumerate(counties):
            ax.annotate(txt, (trasnformedCC[i][0], trasnformedCC[i][1]))


        annot = ax.annotate("", xy=(0,0),  #the point to be annotated  #this is updated
later
                            xytext=(-120,700), #where the annotaion should appear
                            #textcoords="offset points", #forces the annotation to appe
ar relative to xy
                            bbox=dict(boxstyle="round", fc="w")
                            #you can have an arrow to the point if you want#,arrowprops
=dict(arrowstyle="->")
                            )
        annot.set_visible(False)


        names = np.array(counties)


    #     cbar = plt.colorbar()
    #     cbar.set_label('log10 of number of Infected', rotation=270,size=15, labelpad=
25)

        fig.canvas.mpl_connect("motion_notify_event", hover)



        plt.show()

        #return sc


    #plot zeroth day
    plot(0,cd)


    #play button and slider settings
    play = widgets.Play(
        value=0,
        min=1,
        max=dfin,
        step=1,
        interval=100,
        description="Press play",
        disabled=False
    )


    slider = widgets.IntSlider(min=0,max=dfin)
    widgets.jslink((play, 'value'), (slider, 'value'))  #need this - breaks slider if w
e don't have it
    slider_withPlay=widgets.HBox([play, slider])


    #int_range = widgets.IntSlider()

    output2 = widgets.Output()  #this is like the object that our on value change funct
```

```
ion can see

    #display(slider, output2)  #this is the slider without play buttons
    display(slider_withPlay)#, output2)


    def on_value_change(change):
        with output2:
            plt.cla()  #get's rid of old points but not the whole figure

            plot(change['new'],cd.value)

    #only need to draw colour bar once
    cbar = plt.colorbar()
    cbar.set_label('log10 of number of Infected', rotation=270,size=15, labelpad=25)

    slider.observe(on_value_change, names='value')



    #change displayed data
    button = widgets.Button(description="Cycle Data - (I,R,S,N)")
    output = widgets.Output()

    display(button, output)

    def on_button_clicked(cd,b):
        cd.increment()  #cycle through all 3 options

        #print(cd.value)
        #redraw plot to get new annotations
        plt.cla()
        plot(slider.value,cd.value)

    button.on_click(partial(on_button_clicked,cd)) #partial is some weird hack I don't
 understand, but it let's me pass an extra argument
```

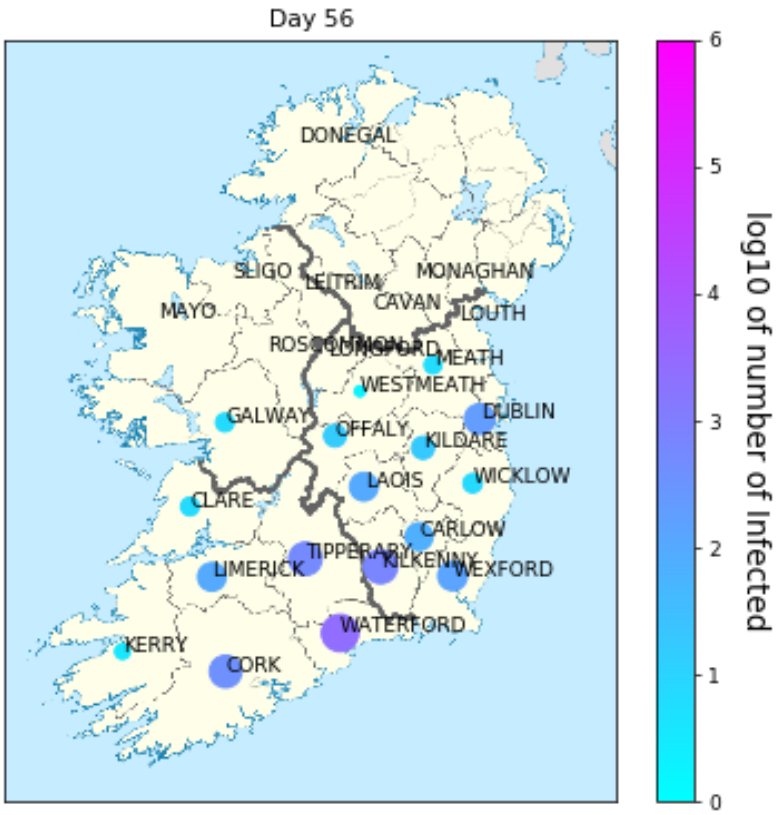
In [140]:

```
setseed=2
r0=2.2#[2.2, 2.2, 1, 2.2, 2.2]
recoveryPeriod=10
initI=5
d=300


[SD,ID,RD,timeD,dD] = TimestatSIRmodel(initialInfected=['WATERFORD','WATERFORD',initI],
r0=r0,recoveryPeriod=recoveryPeriod,seed=setseed,days=d)
```

```
Day  100
Day  199
No more infected -- break
Complete on Day  199
Number of infected left =  0
Max Concurant Infected:  440579    Location: ( DUBLIN , DUBLIN )    Day:
109
Simulation Run Time =  0.762 seconds
```

In [141]:

```
#chose which data you want to visualise
#simulationResults = [SC,IC,RC,dC]   #number susceptible, infected, recovered. dC=numbe
r of days until the infection ended
simulationResults = [SD,ID,RD,dD]

#SC,IC,RC may be much longer than the number dC, with useless filler at the end


#display the stuff
interactiveDisplay(simulationResults)
```



Make a gif

In [77]:

```
def makeGif(I,dfin):

    images = [] #list of file names

    #save an image for each plot
    for day in range(dfin):

        fig,ax=plt.subplots(figsize=(7,7))

        img=plt.imread('./Images/Ireland Provinces.PNG')
        plt.imshow(img)

        #title of plot
        plt.title("Day "+str(day))

        #don't show tickmarks or numbers
        ax.axes.xaxis.set_visible(False)
        ax.axes.yaxis.set_visible(False)

        for i, txt in enumerate(counties):
            ax.annotate(txt, (trasnformedCC[i][0], trasnformedCC[i][1]))


        #visuals for the data points
        size=[100*np.log10(I[x,:,day].sum()) for x in range(26)]
        #color gradient
        c=[np.log10(I[x,:,day].sum()) for x in range(26)]
        #absolute range for colour scale
        norm=plt.Normalize(0, 6)  #10**6 is a million

        sc = plt.scatter([x[0] for x in trasnformedCC],[x[1] for x in trasnformedCC],s=
size,cmap="cool",c=c,norm=norm)#np.arange(26))

        cbar = plt.colorbar()
        cbar.set_label('log10 of number of Infected', rotation=270,size=15, labelpad=25
)

        plt.savefig("./Map Plot Practice/to_gif/"+ str(day), bbox_inches='tight')
        #plt.show() #don't show to save time
        plt.close()


        images.append(imageio.imread("./Images and GIFs/to_gif/"+str(day)+".png"))



    #create gif
    imageio.mimsave('./Map Plot Practice/output.gif', images)
```

In [ ]:

```
#just choose the data you want to use
makeGif(ID,dD)
```

Testing out stuff

compare

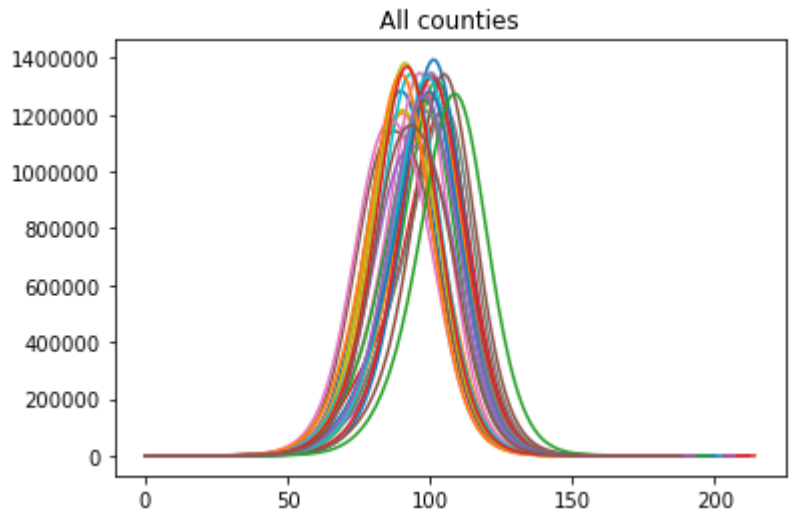
compare starting in each county

In [ ]:

```
setseed=2
r0=2.2#[2.2, 2.2, 1, 2.2, 2.2]
recoveryPeriod=10
initI=5
d=300

countySim = [0]*len(counties)

#simulation starting 5 infected in cork
for i in range(len(counties)):
    countySim[i] = TimestatSIRmodel(initialInfected=[i,i,initI]    , r0=r0,recoveryPeri
od=recoveryPeriod,seed=setseed,days=d)

#[SC,IC,RC,timeC,dC]
```

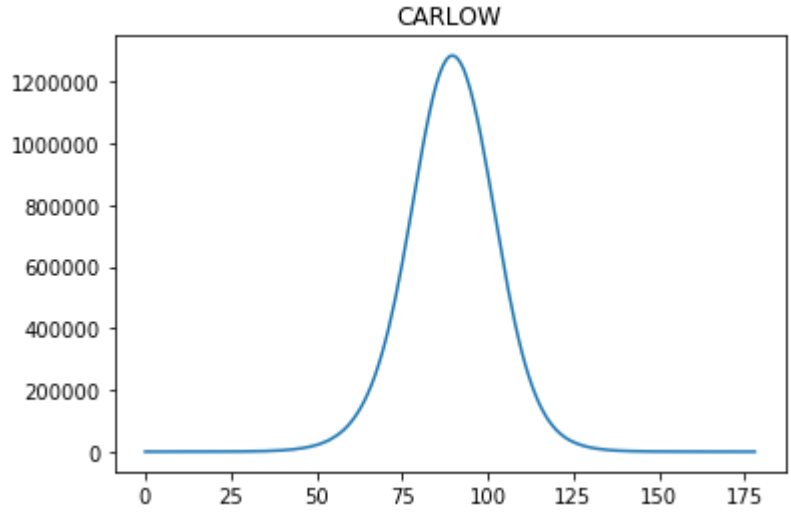
In [112]:

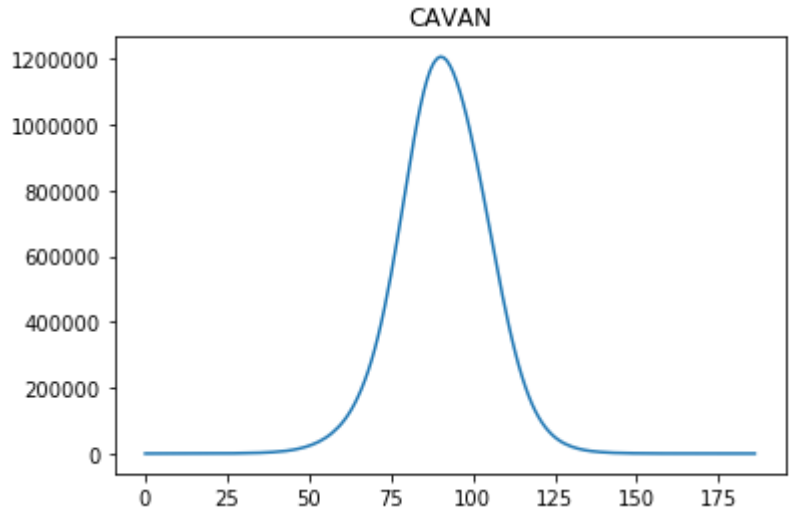
```
for i in range(len(counties)):
    dayFinal = countySim[i][4]
    totI = [countySim[i][1][:,:,dd].sum() for dd in range(dayFinal)]
    plt.plot(range(dayFinal),totI)
    plt.title("All counties")
```

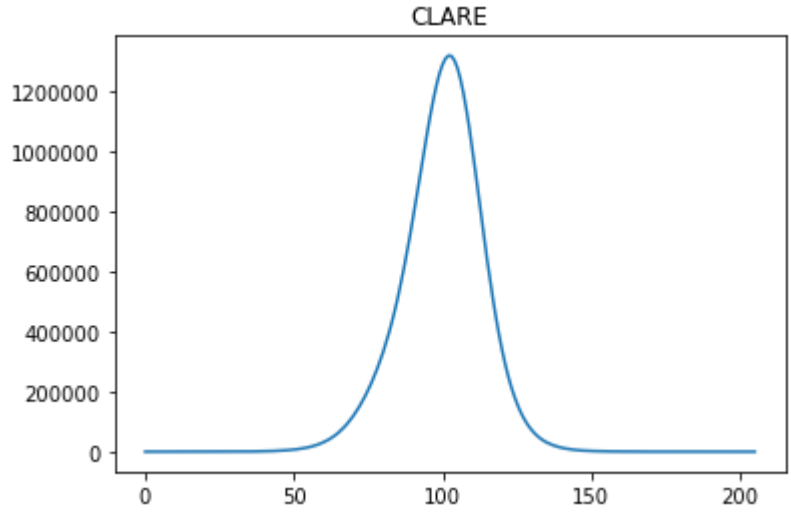


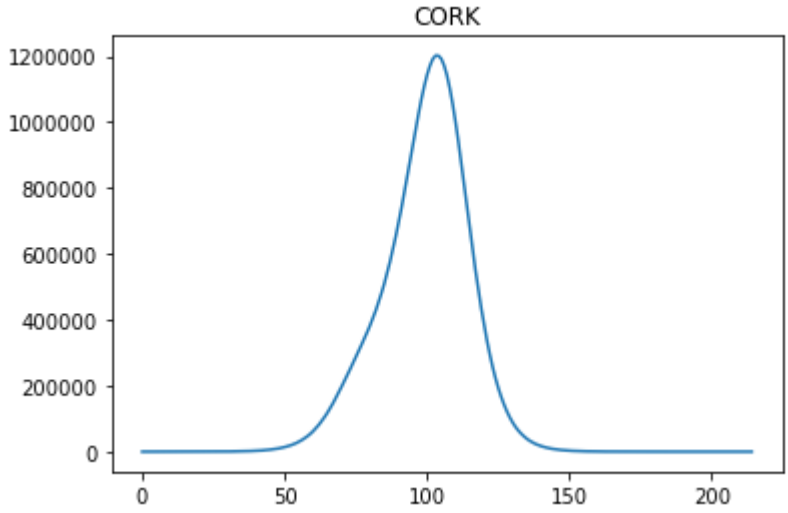
In [111]:

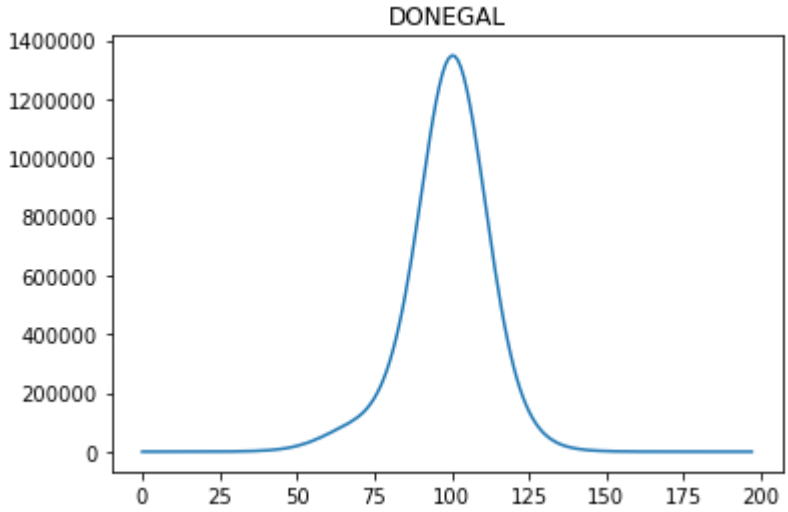
```
%matplotlib inline

for i in range(len(counties)):
    dayFinal = countySim[i][4]
    totI = [countySim[i][1][:,:,dd].sum() for dd in range(dayFinal)]
    plt.plot(range(dayFinal),totI)
    plt.title(str(counties[i]))
    plt.show()
```

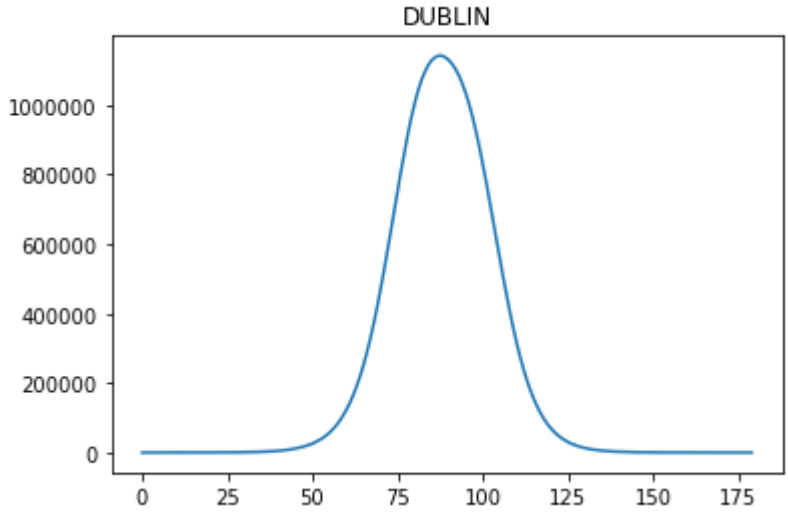

CARLOW
1200000
1000000
800000
600000
400000
200000
0
0
25
50
75
100
125
150
175

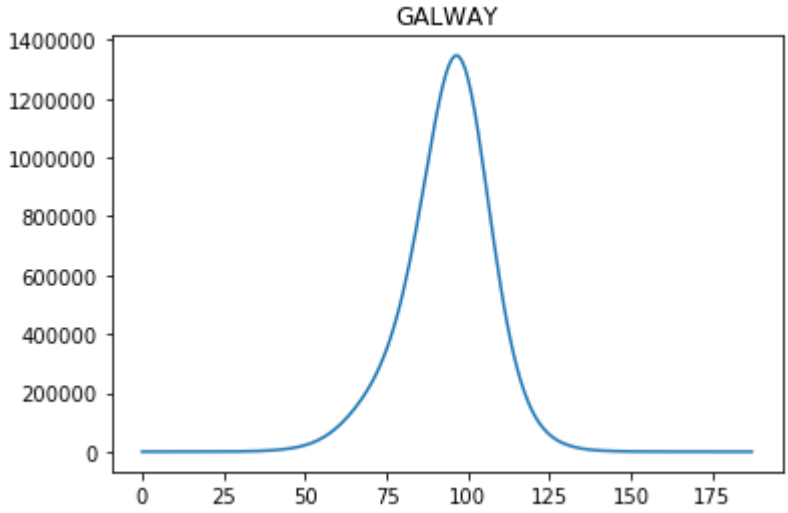

CAVAN
1200000
1000000
800000
600000
400000
200000
0
0
25
50
75
100
125
150
175

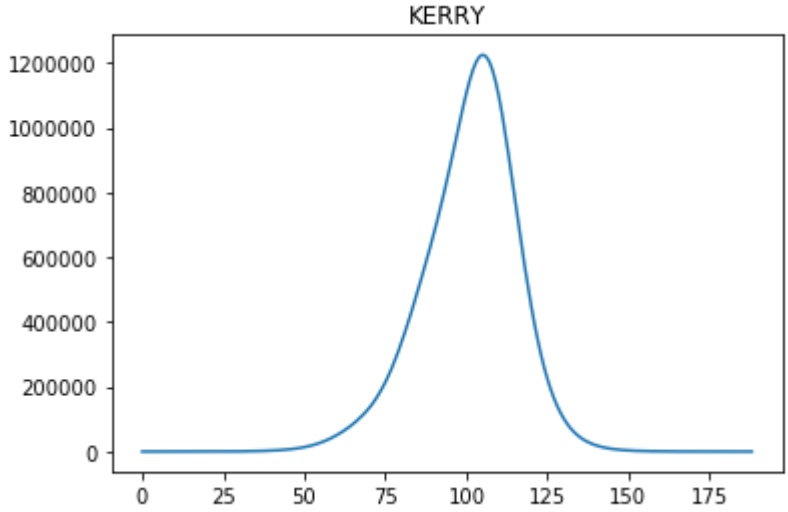

CLARE
1200000
1000000
800000
600000
400000
200000
0
0
50
100
150
200

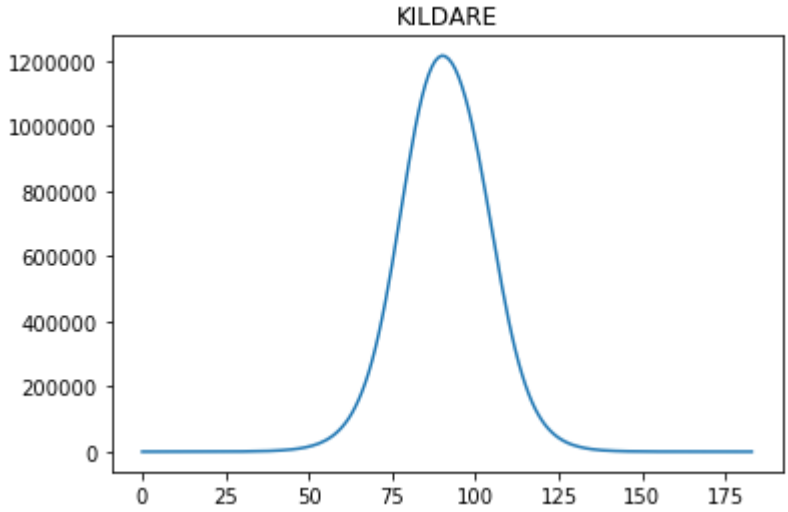

CORK
1200000
1000000
800000
600000
400000
200000
0
0
50
100
150
200

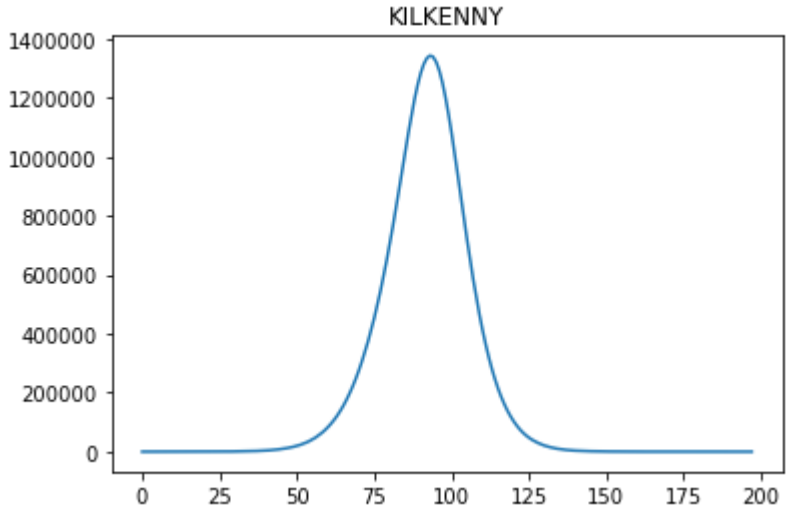

DONEGAL
1400000
1200000
1000000
800000
600000
400000
200000
0
0
25
50
75
100
125
150
175
200

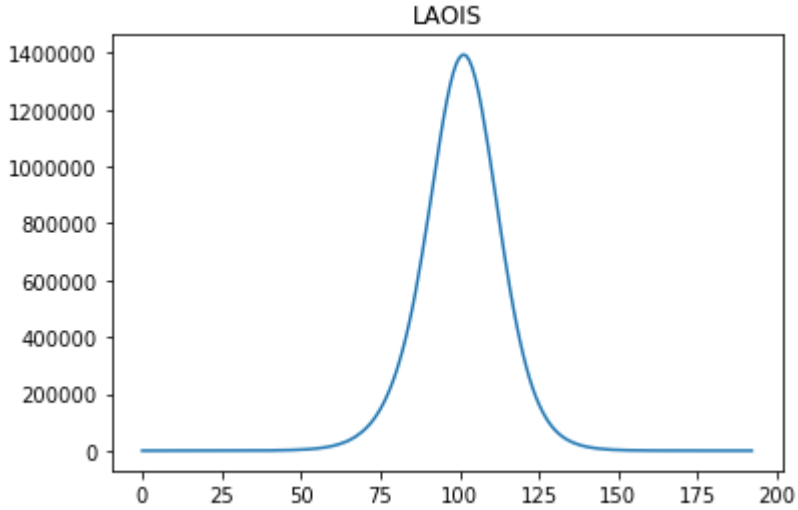

DUBLIN
1000000
800000
600000
400000
200000
0
0
25
50
75
100
125
150
175

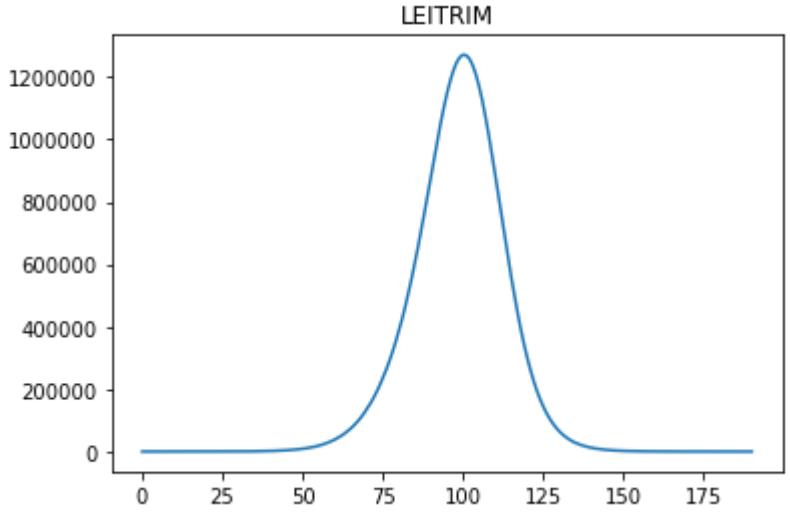

GALWAY
1400000
1200000
1000000
800000
600000
400000
200000
0
0
25
50
75
100
125
150
175

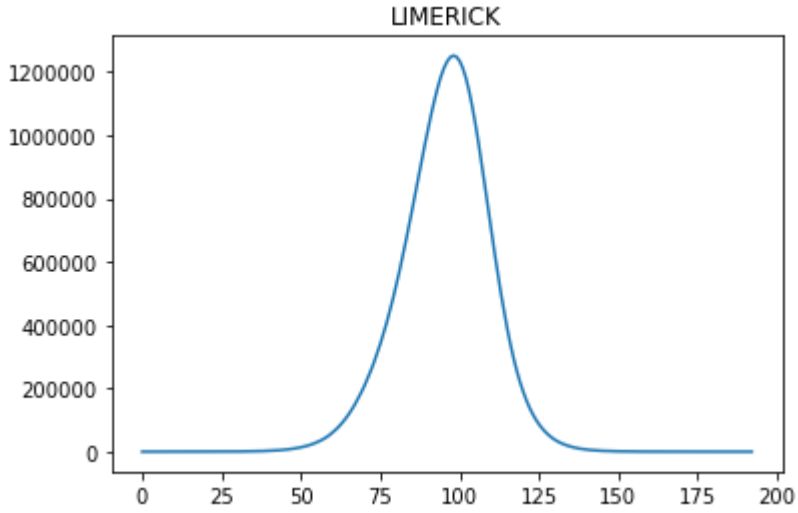

KERRY
1200000
1000000
800000
600000
400000
200000
0
0
25
50
75
100
125
150
175

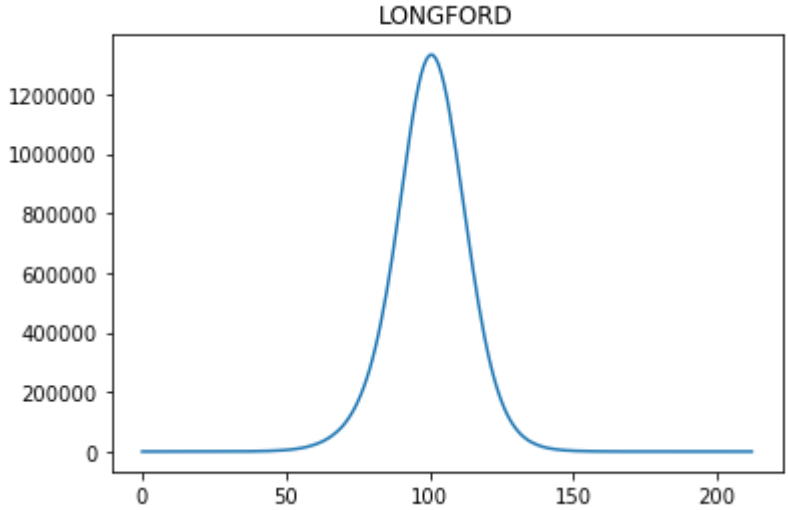

KILDARE
1200000
1000000
800000
600000
400000
200000
0
0
25
50
75
100
125
150
175

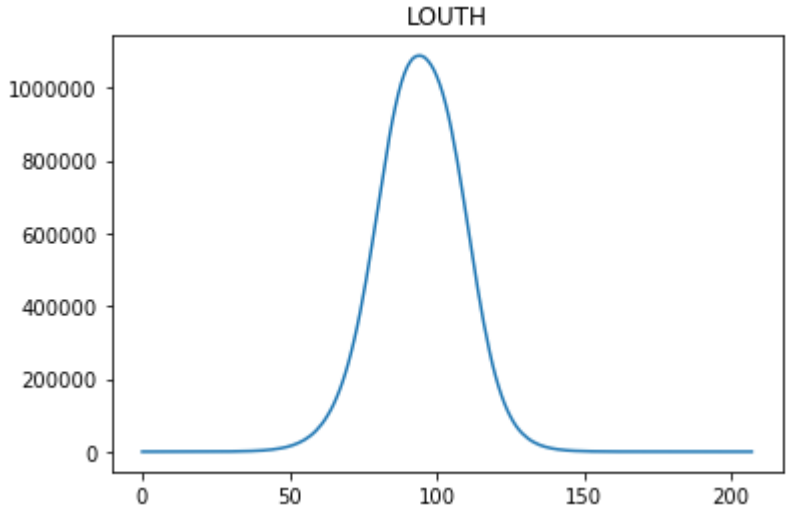

KILKENNY
1400000
1200000
1000000
800000
600000
400000
200000
0
0
25
50
75
100
125
150
175
200

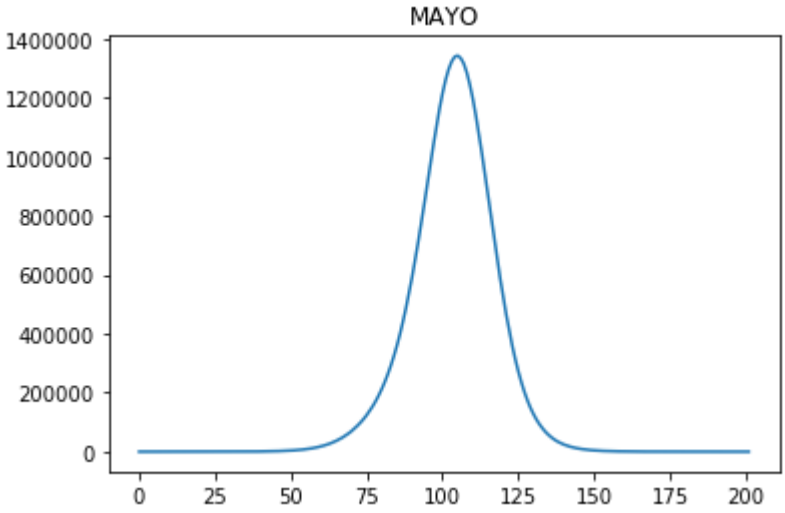

LAOIS
1400000
1200000
1000000
800000
600000
400000
200000
0
0
25
50
75
100
125
150
175
200

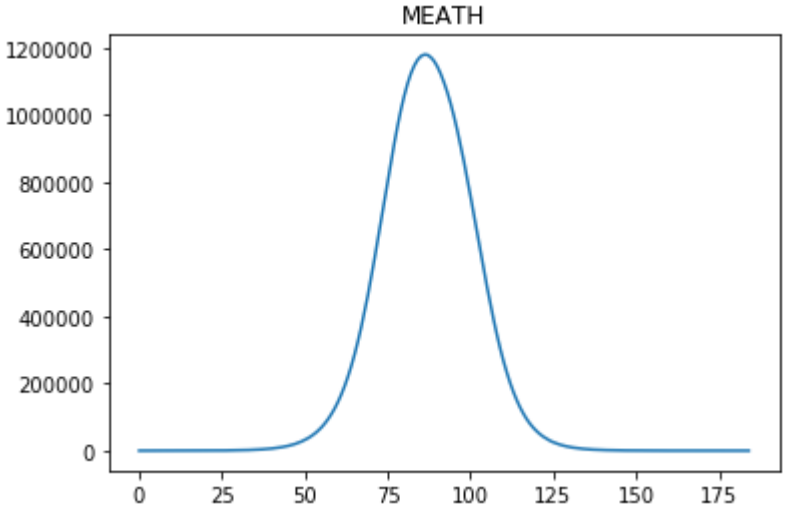

LEITRIM
1200000
1000000
800000
600000
400000
200000
0
0
25
50
75
100
125
150
175

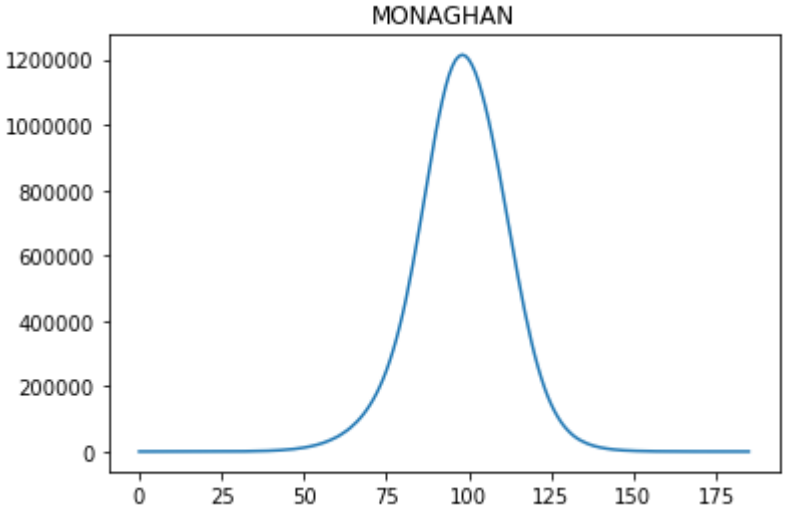

LIMERICK
1200000
1000000
800000
600000
400000
200000
0
0
25
50
75
100
125
150
175
200

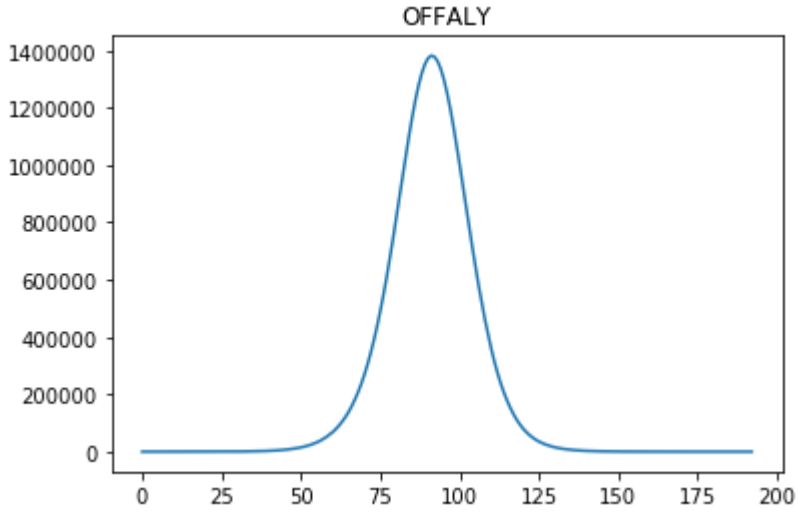

LONGFORD
1200000
1000000
800000
600000
400000
200000
0
0
50
100
150
200

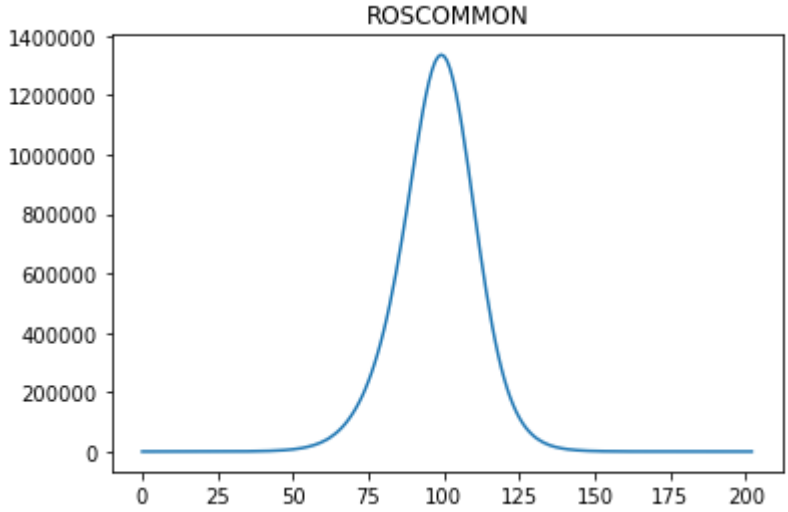

LOUTH
1000000
800000
600000
400000
200000
0
0
50
100
150
200

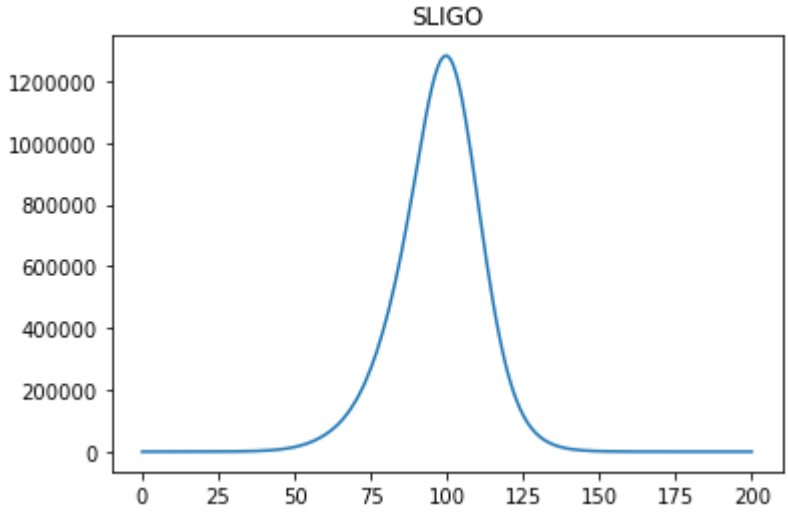

MAYO
1400000
1200000
1000000
800000
600000
400000
200000
0
0
25
50
75
100
125
150
175
200

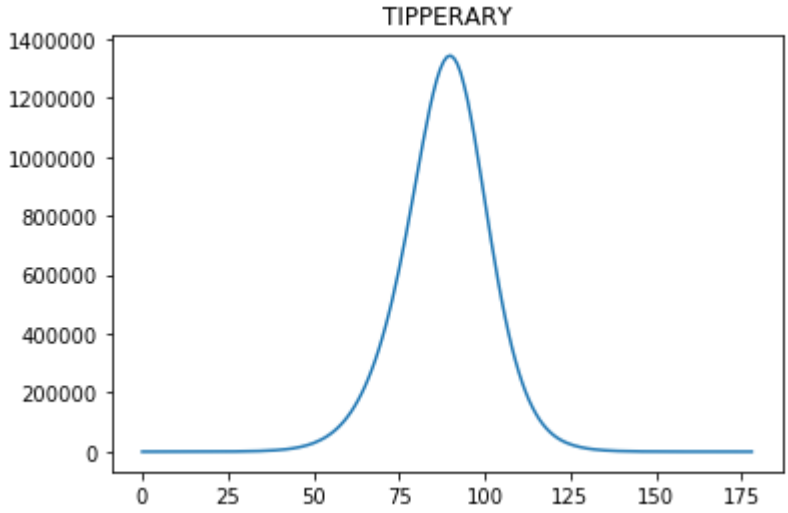

MEATH
1200000
1000000
800000
600000
400000
200000
0
0
25
50
75
100
125
150
175

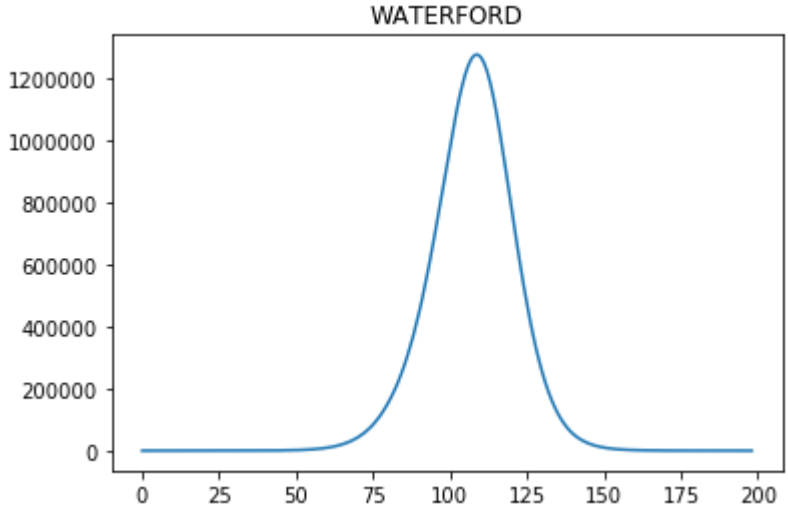

MONAGHAN
1200000
1000000
800000
600000
400000
200000
0
0
25
50
75
100
125
150
175

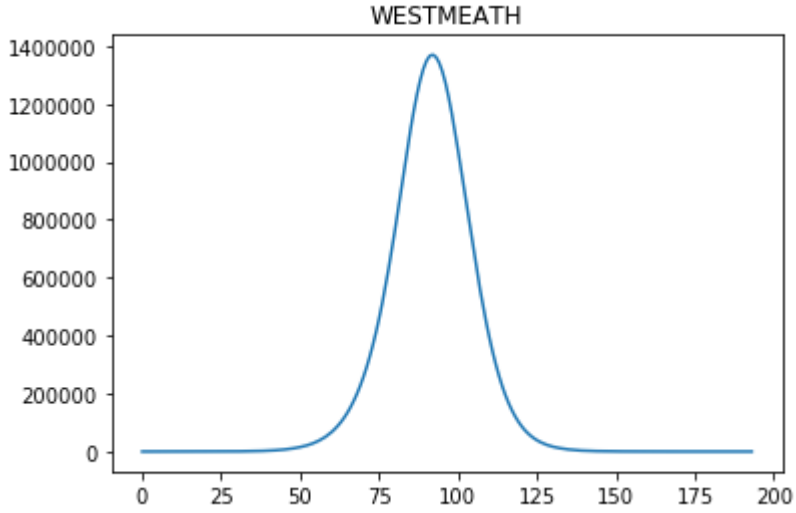

OFFALY
1400000
1200000
1000000
800000
600000
400000
200000
0
0
25
50
75
100
125
150
175
200

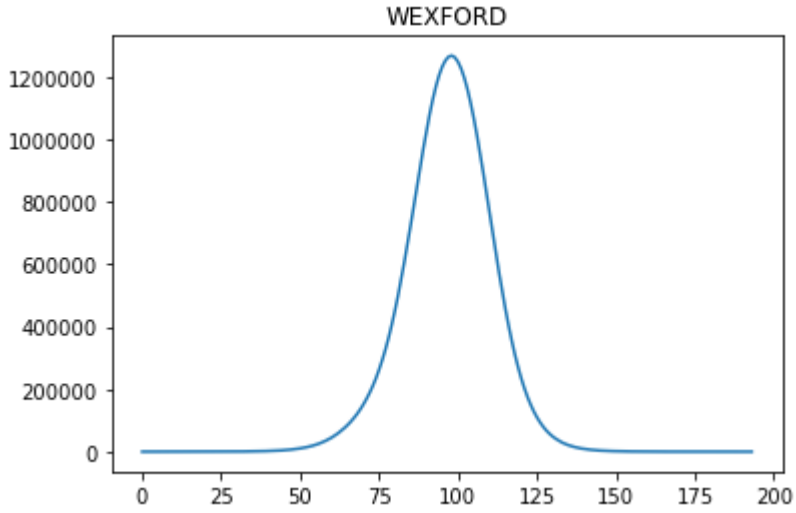

ROSCOMMON
1400000
1200000
1000000
800000
600000
400000
200000
0
0
25
50
75
100
125
150
175
200

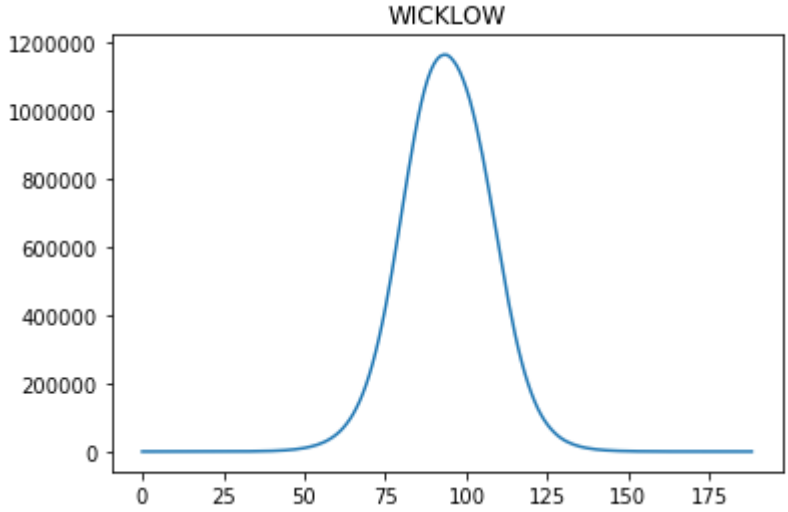

SLIGO
1200000
1000000
800000
600000
400000
200000
0
0
25
50
75
100
125
150
175
200


TIPPERARY
1400000
1200000
1000000
800000
600000
400000
200000
0
0
25
50
75
100
125
150
175


WATERFORD
1200000
1000000
800000
600000
400000
200000
0
0
25
50
75
100
125
150
175
200


WESTMEATH
1400000
1200000
1000000
800000
600000
400000
200000
0
0
25
50
75
100
125
150
175
200


WEXFORD
1200000
1000000
800000
600000
400000
200000
0
0
25
50
75
100
125
150
175
200


WICKLOW
1200000
1000000
800000
600000
400000
200000
0
0
25
50
75
100
125
150
175

In [126]:

```
#I want to know the order each county peaks, and I want to check this against another r
andom seed

maxDays = [0]*len(counties)

for i in range(len(counties)):
    dayFinal = countySim[i][4]
    totI = [countySim[i][1][:,:,dd].sum() for dd in range(dayFinal)]

    maxDays[i] = (totI.index(max(totI)),counties[i])

maxDays
```

Out[126]:

```
[(90, 'CARLOW'),
 (90, 'CAVAN'),
 (102, 'CLARE'),
 (103, 'CORK'),
 (100, 'DONEGAL'),
 (87, 'DUBLIN'),
 (96, 'GALWAY'),
 (105, 'KERRY'),
 (90, 'KILDARE'),
 (93, 'KILKENNY'),
 (101, 'LAOIS'),
 (100, 'LEITRIM'),
 (98, 'LIMERICK'),
 (101, 'LONGFORD'),
 (94, 'LOUTH'),
 (105, 'MAYO'),
 (86, 'MEATH'),
 (98, 'MONAGHAN'),
 (91, 'OFFALY'),
 (99, 'ROSCOMMON'),
 (100, 'SLIGO'),
 (90, 'TIPPERARY'),
 (109, 'WATERFORD'),
 (92, 'WESTMEATH'),
 (98, 'WEXFORD'),
 (93, 'WICKLOW')]
```

In [127]:

```
maxDays.sort(key=lambda x : x[0])

maxDays
```

Out[127]:

```
[(86, 'MEATH'),
 (87, 'DUBLIN'),
 (90, 'CARLOW'),
 (90, 'CAVAN'),
 (90, 'KILDARE'),
 (90, 'TIPPERARY'),
 (91, 'OFFALY'),
 (92, 'WESTMEATH'),
 (93, 'KILKENNY'),
 (93, 'WICKLOW'),
 (94, 'LOUTH'),
 (96, 'GALWAY'),
 (98, 'LIMERICK'),
 (98, 'MONAGHAN'),
 (98, 'WEXFORD'),
 (99, 'ROSCOMMON'),
 (100, 'DONEGAL'),
 (100, 'LEITRIM'),
 (100, 'SLIGO'),
 (101, 'LAOIS'),
 (101, 'LONGFORD'),
 (102, 'CLARE'),
 (103, 'CORK'),
 (105, 'KERRY'),
 (105, 'MAYO'),
 (109, 'WATERFORD')]
```

In [26]:

```
orderList = []
```

In [36]:

```
#let's get a second random seed

setseed=5
r0=2.2#[2.2, 2.2, 1, 2.2, 2.2]
recoveryPeriod=10
initI=5
d=300

countySim = [0]*len(counties)

#simulation starting 5 infected in cork
for i in range(len(counties)):
    countySim[i] = TimestatSIRmodel(initialInfected=[i,i,initI]    , r0=r0,recoveryPeri
od=recoveryPeriod,seed=setseed,days=d)

#[SC,IC,RC,timeC,dC]

#I want to know the order each county peaks, and I want to check this against another r
andom seed

maxDays = [0]*len(counties)

for i in range(len(counties)):
    dayFinal = countySim[i][4]
    totI = [countySim[i][1][:,:,dd].sum() for dd in range(dayFinal)]

    maxDays[i] = (totI.index(max(totI)),counties[i])


maxDays.sort(key=lambda x : x[0])
```

```
Day  100
Day  200
Day  202
No more infected -- break
Complete on Day  202
Number of infected left =  0
Max Concurant Infected:  441592    Location: ( DUBLIN , DUBLIN )    Day:
92
Simulation Run Time =  0.798 seconds

Day  100
Day  178
No more infected -- break
Complete on Day  178
Number of infected left =  0
Max Concurant Infected:  437295    Location: ( DUBLIN , DUBLIN )    Day:
85
Simulation Run Time =  0.687 seconds

Day  100
Day  197
No more infected -- break
Complete on Day  197
Number of infected left =  0
Max Concurant Infected:  437482    Location: ( DUBLIN , DUBLIN )    Day:
109
Simulation Run Time =  0.752 seconds

Day  100
Day  199
No more infected -- break
Complete on Day  199
Number of infected left =  0
Max Concurant Infected:  436348    Location: ( DUBLIN , DUBLIN )    Day:
105
Simulation Run Time =  0.774 seconds

Day  100
Day  195
No more infected -- break
Complete on Day  195
Number of infected left =  0
Max Concurant Infected:  434308    Location: ( DUBLIN , DUBLIN )    Day:
107
Simulation Run Time =  0.757 seconds

Day  100
Day  187
No more infected -- break
Complete on Day  187
Number of infected left =  0
Max Concurant Infected:  424556    Location: ( DUBLIN , DUBLIN )    Day:
81
Simulation Run Time =  0.707 seconds

Day  100
Day  183
No more infected -- break
Complete on Day  183
Number of infected left =  0
Max Concurant Infected:  441658    Location: ( DUBLIN , DUBLIN )    Day:
```

```
99
Simulation Run Time =  0.696 seconds

Day  100
Day  192
No more infected -- break
Complete on Day  192
Number of infected left =  0
Max Concurant Infected:  433600    Location: ( DUBLIN , DUBLIN )    Day:
104
Simulation Run Time =  0.735 seconds

Day  100
Day  181
No more infected -- break
Complete on Day  181
Number of infected left =  0
Max Concurant Infected:  435923    Location: ( DUBLIN , DUBLIN )    Day:
81
Simulation Run Time =  0.692 seconds

Day  100
Day  196
No more infected -- break
Complete on Day  196
Number of infected left =  0
Max Concurant Infected:  444421    Location: ( DUBLIN , DUBLIN )    Day:
96
Simulation Run Time =  0.751 seconds

Day  100
Day  185
No more infected -- break
Complete on Day  185
Number of infected left =  0
Max Concurant Infected:  438092    Location: ( DUBLIN , DUBLIN )    Day:
88
Simulation Run Time =  0.708 seconds

Day  100
Day  200
Day  218
No more infected -- break
Complete on Day  218
Number of infected left =  0
Max Concurant Infected:  439956    Location: ( DUBLIN , DUBLIN )    Day:
117
Simulation Run Time =  0.831 seconds

Day  100
Day  185
No more infected -- break
Complete on Day  185
Number of infected left =  0
Max Concurant Infected:  439753    Location: ( DUBLIN , DUBLIN )    Day:
100
Simulation Run Time =  0.708 seconds

Day  100
Day  200
Day  202
```

```
No more infected -- break
Complete on Day  202
Number of infected left =  0
Max Concurant Infected:  438131    Location: ( DUBLIN , DUBLIN )    Day:
97
Simulation Run Time =  0.772 seconds

Day  100
Day  200
Day  200
No more infected -- break
Complete on Day  200
Number of infected left =  0
Max Concurant Infected:  435730    Location: ( DUBLIN , DUBLIN )    Day:
85
Simulation Run Time =  0.759 seconds

Day  100
Day  195
No more infected -- break
Complete on Day  195
Number of infected left =  0
Max Concurant Infected:  434271    Location: ( DUBLIN , DUBLIN )    Day:
99
Simulation Run Time =  0.76 seconds

Day  100
Day  192
No more infected -- break
Complete on Day  192
Number of infected left =  0
Max Concurant Infected:  435164    Location: ( DUBLIN , DUBLIN )    Day:
84
Simulation Run Time =  0.731 seconds

Day  100
Day  198
No more infected -- break
Complete on Day  198
Number of infected left =  0
Max Concurant Infected:  437360    Location: ( DUBLIN , DUBLIN )    Day:
91
Simulation Run Time =  0.753 seconds

Day  100
Day  200
Day  202
No more infected -- break
Complete on Day  202
Number of infected left =  0
Max Concurant Infected:  439650    Location: ( DUBLIN , DUBLIN )    Day:
87
Simulation Run Time =  0.767 seconds

Day  100
Day  199
No more infected -- break
Complete on Day  199
Number of infected left =  0
Max Concurant Infected:  439370    Location: ( DUBLIN , DUBLIN )    Day:
101
```

```
Simulation Run Time =  0.757 seconds

Day  100
Day  190
No more infected -- break
Complete on Day  190
Number of infected left =  0
Max Concurant Infected:  437853    Location: ( DUBLIN , DUBLIN )    Day:
100
Simulation Run Time =  0.732 seconds

Day  100
Day  196
No more infected -- break
Complete on Day  196
Number of infected left =  0
Max Concurant Infected:  442951    Location: ( DUBLIN , DUBLIN )    Day:
95
Simulation Run Time =  0.746 seconds

Day  100
Day  200
Day  207
No more infected -- break
Complete on Day  207
Number of infected left =  0
Max Concurant Infected:  442902    Location: ( DUBLIN , DUBLIN )    Day:
105
Simulation Run Time =  0.782 seconds

Day  100
Day  200
Day  200
No more infected -- break
Complete on Day  200
Number of infected left =  0
Max Concurant Infected:  441047    Location: ( DUBLIN , DUBLIN )    Day:
95
Simulation Run Time =  0.757 seconds

Day  100
Day  200
Day  209
No more infected -- break
Complete on Day  209
Number of infected left =  0
Max Concurant Infected:  434011    Location: ( DUBLIN , DUBLIN )    Day:
89
Simulation Run Time =  0.792 seconds

Day  100
Day  192
No more infected -- break
Complete on Day  192
Number of infected left =  0
Max Concurant Infected:  429545    Location: ( DUBLIN , DUBLIN )    Day:
88
Simulation Run Time =  0.742 seconds
```

In [37]:

```
orderList.append(list(map(lambda x : x[1],maxDays)))
```

In [33]:

```
orderList
```

Out[33]:

```
[['MEATH',
  'LOUTH',
  'DUBLIN',
  'LONGFORD',
  'KILDARE',
  'TIPPERARY',
  'WESTMEATH',
  'ROSCOMMON',
  'CAVAN',
  'LAOIS',
  'WATERFORD',
  'WICKLOW',
  'WEXFORD',
  'KILKENNY',
  'MAYO',
  'CARLOW',
  'CLARE',
  'CORK',
  'GALWAY',
  'LEITRIM',
  'DONEGAL',
  'KERRY',
  'MONAGHAN',
  'LIMERICK',
  'SLIGO',
  'OFFALY'],
 ['MEATH',
  'DUBLIN',
  'CARLOW',
  'CAVAN',
  'KILDARE',
  'TIPPERARY',
  'OFFALY',
  'WESTMEATH',
  'KILKENNY',
  'WICKLOW',
  'LOUTH',
  'GALWAY',
  'LIMERICK',
  'MONAGHAN',
  'WEXFORD',
  'ROSCOMMON',
  'DONEGAL',
  'LEITRIM',
  'SLIGO',
  'LAOIS',
  'LONGFORD',
  'CLARE',
  'CORK',
  'KERRY',
  'MAYO',
  'WATERFORD'],
 ['KILKENNY',
  'KILDARE',
  'DUBLIN',
  'LAOIS',
  'LIMERICK',
  'CAVAN',
  'MEATH',
```

```
    'WEXFORD',
    'GALWAY',
    'LOUTH',
    'ROSCOMMON',
    'WICKLOW',
    'CARLOW',
    'OFFALY',
    'WATERFORD',
    'LEITRIM',
    'LONGFORD',
    'CLARE',
    'MONAGHAN',
    'SLIGO',
    'WESTMEATH',
    'KERRY',
    'TIPPERARY',
    'DONEGAL',
    'MAYO',
    'CORK']]
```

In [40]:

```
countyWeight=[]
for i in counties:
    count=0
    for j in range(len(orderList)):
        count += orderList[j].index(i)

    countyWeight.append(count/5)
```

In [41]:

```
countyWeight
```

Out[41]:

```
[8.6,
 4.2,
 20.2,
 21.2,
 20.8,
 2.6,
 13.4,
 19.8,
 4.0,
 8.0,
 9.6,
 18.6,
 15.8,
 12.6,
 8.8,
 20.4,
 2.0,
 12.8,
 9.6,
 12.8,
 20.2,
 9.6,
 17.8,
 10.2,
 11.2,
 10.2]
```

In [42]:

```
table = [(counties[i],countyWeight[i]) for i in range(26)]
table
```

Out[42]:

```
[('CARLOW', 8.6),
 ('CAVAN', 4.2),
 ('CLARE', 20.2),
 ('CORK', 21.2),
 ('DONEGAL', 20.8),
 ('DUBLIN', 2.6),
 ('GALWAY', 13.4),
 ('KERRY', 19.8),
 ('KILDARE', 4.0),
 ('KILKENNY', 8.0),
 ('LAOIS', 9.6),
 ('LEITRIM', 18.6),
 ('LIMERICK', 15.8),
 ('LONGFORD', 12.6),
 ('LOUTH', 8.8),
 ('MAYO', 20.4),
 ('MEATH', 2.0),
 ('MONAGHAN', 12.8),
 ('OFFALY', 9.6),
 ('ROSCOMMON', 12.8),
 ('SLIGO', 20.2),
 ('TIPPERARY', 9.6),
 ('WATERFORD', 17.8),
 ('WESTMEATH', 10.2),
 ('WEXFORD', 11.2),
 ('WICKLOW', 10.2)]
```

In [43]:

```
table.sort(key=lambda x: x[1])
table
```

Out[43]:

```
[('MEATH', 2.0),
 ('DUBLIN', 2.6),
 ('KILDARE', 4.0),
 ('CAVAN', 4.2),
 ('KILKENNY', 8.0),
 ('CARLOW', 8.6),
 ('LOUTH', 8.8),
 ('LAOIS', 9.6),
 ('OFFALY', 9.6),
 ('TIPPERARY', 9.6),
 ('WESTMEATH', 10.2),
 ('WICKLOW', 10.2),
 ('WEXFORD', 11.2),
 ('LONGFORD', 12.6),
 ('MONAGHAN', 12.8),
 ('ROSCOMMON', 12.8),
 ('GALWAY', 13.4),
 ('LIMERICK', 15.8),
 ('WATERFORD', 17.8),
 ('LEITRIM', 18.6),
 ('KERRY', 19.8),
 ('CLARE', 20.2),
 ('SLIGO', 20.2),
 ('MAYO', 20.4),
 ('DONEGAL', 20.8),
 ('CORK', 21.2)]
```

In [ ]:

```

```