Differential Equations Final Project

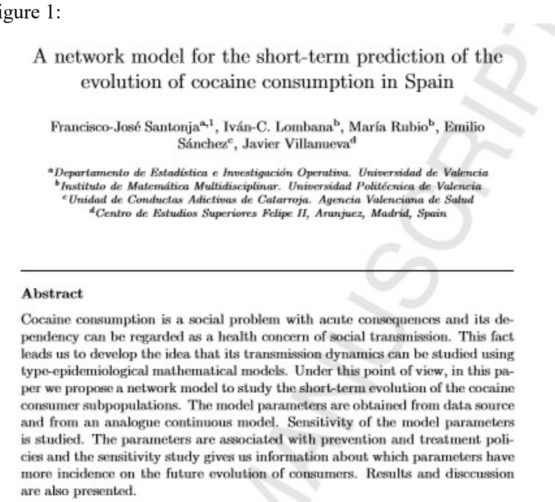
Robert Goodloe
Kalamazoo
3/13/2017

Before any math is involved I will load the MATH280 package for the necessary functions

```
(%i1) load("C:/Users/Rj/Documents/Differential Equations/MATH280.mac")$
```

;; loading \ensuremath{\neq}P"C:/Users/Rj/maxima/binary/5_38_1_5_gdf93b7b_dirty/sbcl/1_3_5/share/draw/grcommon.fasl"
;; loading \ensuremath{\neq}P"C:/Users/Rj/maxima/binary/5_38_1_5_gdf93b7b_dirty/sbcl/1_3_5/share/draw/gnuplot.fasl"
;; loading \ensuremath{\neq}P"C:/Users/Rj/maxima/binary/5_38_1_5_gdf93b7b_dirty/sbcl/1_3_5/share/draw/vtk.fasl"
;; loading \ensuremath{\neq}P"C:/Users/Rj/maxima/binary/5_38_1_5_gdf93b7b_dirty/sbcl/1_3_5/share/draw/picture.fasl"
$MATH280.mac: \ help(MATH280) \ for\, contents$

Figure 1:



A network model for the short-term prediction of the evolution of cocaine consumption in Spain

Francisco-José Santonja[a,1], Iván-C. Lombana[b], María Rubio[b], Emilio Sánchez[c], Javier Villanueva[d]

[a]Departamento de Estadística e Investigación Operativa. Universidad de Valencia
[b]Instituto de Matemática Multidisciplinar. Universidad Politécnica de Valencia
[c]Unidad de Conductas Adictivas de Catarroja. Agencia Valenciana de Salud
[d]Centro de Estudios Superiores Felipe II, Aranjuez, Madrid, Spain

Abstract

Cocaine consumption is a social problem with acute consequences and its dependency can be regarded as a health concern of social transmission. This fact leads us to develop the idea that its transmission dynamics can be studied using type-epidemiological mathematical models. Under this point of view, in this paper we propose a network model to study the short-term evolution of the cocaine consumer subpopulations. The model parameters are obtained from data source and from an analogue continuous model. Sensitivity of the model parameters is studied. The parameters are associated with prevention and treatment policies and the sensitivity study gives us information about which parameters have more incidence on the future evolution of consumers. Results and disccussion are also presented.

This network model was constructed using the following
non-linear system of equations

Figure 2:

$$
\begin{aligned}
N'(t) &= \mu P(t) - dN(t) - \beta\frac{N(t)(C_o(t)+C_r(t)+C_b(t))}{P(t)} + \epsilon C_b(t), && (1)\\
C_o'(t) &= \beta\frac{N(t)(C_o(t)+C_r(t)+C_b(t))}{P(t)} - d_c C_o(t) - \gamma C_o(t), && (2)\\
C_r'(t) &= \gamma C_o(t) - d_c C_r(t) - \sigma C_r(t), && (3)\\
C_b'(t) &= \sigma C_r(t) - d_c C_b(t) - \epsilon C_b(t), && (4)\\
P(t) &= N(t) + C_o(t) + C_r(t) + C_b(t). && (5)
\end{aligned}
$$

Where N(t) represents the function of the non-consumer population in Spain with respect to time.
Co represents the time function of the occasional consumer population: who are defined to consume cocaine sometimes.
Cr represents the time function of the regular consumer population: who are defined to have consumed cocaine in the last year.
Cb represents the time function of the habitual consumer population: who are defined to have consumed cocaine in the last month.

Furthermore, most of the constants used to model the
dynamic of cocaine were derived from research (cited in the paper)
related to SIR, despite the network model approach

Figure 3: Defined Coefficients

- $\mu = 0.01$ years^{-1}, is the average Spanish birth rate between years 1995-2007 [18].
- $d = 0.008388$ years^{-1} is the average Spanish death rate between years 1995-2007 [18].
- $d_c = 0.01636$ years^{-1} is the augmented death rate due to drug consumption. In Spain, approximately 6.8% of mortality is due to drugs consumption [17].

Figure 4:

- $\epsilon = 0.0000456$ years^{-1}. From official data [17] 4.25% of habitual consumers begin a therapy program every year. Furthermore, using data from Table 1 corresponding to National Drug Observatory Reports [17], the average proportion of population with habitual consumption is 0.93%. Moreover, the conclusion specified in [17, 19, 20] that a habitual consumer takes about nine years before going to therapy. Therefore, the percentage of habitual consumers in therapy per year is 0.00439%. To be precise, $0.0093 \times 0.0425 \times 1/9 = 0.0000439$. Additionally, [20, 21, 22, 23, 24, 25, 26] they conclude that around 52% of the individuals on therapy recover with an average of six months. Then, we obtained $\epsilon = 0.0000439 \times 0.52 \times 1/0.5 = 0.0000456$, i.e.,

$$\begin{aligned} \epsilon &= \epsilon_1 \times \epsilon_2 \times \epsilon_3 \times \epsilon_4 \times \epsilon_5 = \\ &= 0.0093 \times 0.0425 \times 1/9 \times 0.52 \times 1/0.5 = 0.0000456. \end{aligned} \tag{6}$$

- $\beta = 0.09614$, transmission rate due to social pressure to consume cocaine.
- $\gamma = 0.0596$, rate at which an occasional consumer transits to the regular consumption subpopulation.
- $\sigma = 0.0579$, rate at which a regular consumer transits to the habitual consumption subpopulation.
- The initial conditions of the model are taken for year 1995 ($t = 0$), $N(t = 0) = 0.944$, $C_o(t = 0) = 0.034$, $C_r(t = 0) = 0.018$ and $C_b(t = 0) = 0.004$.

```
(%i1304) /* determined coefficients (all are adjusted for
           proper units since the mathematical model will be
           evaluated with respect to months) */
         mu:0.01/12$
         beta:0.09614/12$
         gamma:0.0596/12$
         sigma:0.0579/12$
         epsilon:0.0000456/12$
         dc:0.01636/12$
         d:0.008388/12$
```

Figure 5:

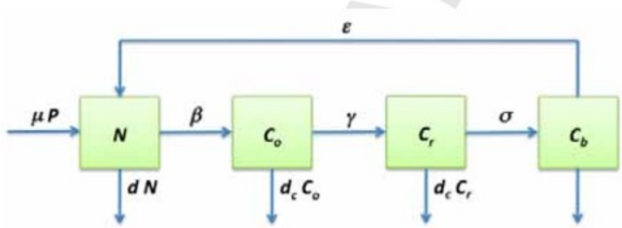


Figure 6: In order to simulate new individuals entering the population, the authors include:

The evolution rules are as follows:

- The new 15-years-old individuals enter in the system as non-consumers following a Poisson distribution of mean $\lambda = \mu N/12 = \frac{0.01}{12} N$ people/month. Poisson distribution has been chosen because is the natural discretization of the underlying exponential distribution in the continuous model which mean is μ. Thus, for every time step, we compute a pseudorandom number $0 \le s \le 1$ and we find the minimum natural number i such that

$$\sum_{k=0}^{i} \frac{e^{-\lambda}\lambda^k}{k!} > s.$$

Based on this "evolution rule",

```
(%i885) N:9440$ /*initial N given in data table (0.944*10000) */
        ss:sum(((%e^(-mu*N))*(mu*N)^k)/k!,k,0,i);
```

$$3.833099348473731 10^{-4} \sum_{k=0}^{i} \frac{7.866666666666667^k}{k!} \quad \text{(ss)}$$

Unfortunately, in maxima it is difficult to solve for "i" in this situation.
However, there are other ways to model this even if they are crude.
Here, I ran the sum for various values of i until the sum ss=1.
At this point, ss > s must be true and we can add the new individuals to population N

```
(%i901) sstest0:sum(((%e^(-mu*N))*(mu*N)^k)/k!,k,0,0);
        sstest1:sum(((%e^(-mu*N))*(mu*N)^k)/k!,k,0,1);
        sstest2:sum(((%e^(-mu*N))*(mu*N)^k)/k!,k,0,2);
        sstest3:sum(((%e^(-mu*N))*(mu*N)^k)/k!,k,0,3);
        sstest4:sum(((%e^(-mu*N))*(mu*N)^k)/k!,k,0,4);
        sstest5:sum(((%e^(-mu*N))*(mu*N)^k)/k!,k,0,5);
        sstest6:sum(((%e^(-mu*N))*(mu*N)^k)/k!,k,0,6);
        sstest7:sum(((%e^(-mu*N))*(mu*N)^k)/k!,k,0,7);
        sstest8:sum(((%e^(-mu*N))*(mu*N)^k)/k!,k,0,8);
        sstest9:sum(((%e^(-mu*N))*(mu*N)^k)/k!,k,0,9);
        sstest10:sum(((%e^(-mu*N))*(mu*N)^k)/k!,k,0,10);
        sstest11:sum(((%e^(-mu*N))*(mu*N)^k)/k!,k,0,11);
        sstest12:sum(((%e^(-mu*N))*(mu*N)^k)/k!,k,0,12);
        sstest13:sum(((%e^(-mu*N))*(mu*N)^k)/k!,k,0,13);
        sstest14:sum(((%e^(-mu*N))*(mu*N)^k)/k!,k,0,14);
        sstest15:sum(((%e^(-mu*N))*(mu*N)^k)/k!,k,0,15);
```

$3.833099348473731 10^{-4}$ (sstest0)

0.003398681422313375 (sstest1)

0.01525914260634632 (sstest2)

0.04635990748892159 (sstest3)

0.1075247450913196 (sstest4)

0.2037574229190925 (sstest5)

0.3299291560710615 (sstest6)

0.4717221514227981 (sstest7)

0.611151930185339 (sstest8)

0.7330238849555599 (sstest9)

0.8288964893748004 (sstest10)

0.8974599276867421 (sstest11)

0.9424070705801261 (sstest12)

0.9696058544848405 (sstest13)

0.9929041313051455 (sstest15)
0.9848889806789182 (sstest14)

Using these, I estimated that the original authors had new individuals
enter the system at an average rate of 0.000075*t
so in rhsN (the derivative of N) this looks like +0.000075

(%i922)
```
/*Given Equations*/
rhsN:mu*(N+Co+Cr+Cb)-d*N-beta*((N*(Co+Cr+Cb))/(N+Co+Cr+Cb))+epsilon*Cb+.000075$
rhsCo:beta*((N*(Co+Cr+Cb))/(N+Co+Cr+Cb))-dc*Co-gamma*Co$
rhsCr:gamma*Co-dc*Cr-sigma*Cr$
rhsCb:sigma*Cr-dc*Cb-epsilon*Cb$
P:N+Co+Cr+Cb$
```

Lastly, the initial conditions are found in data table 1.
t=0 corresponds to December 1995 with the given values.

Figure 7: Data Table 1

Percentages/years	1995	1997	1999	2001	2003	2005
% Non-consumer	0.944	0.948	0.948	0.911	0.903	0.884
% Occasional consumers	0.034	0.032	0.031	0.049	0.059	0.070
% Regular consumers	0.018	0.015	0.015	0.026	0.027	0.030
% Habitual consumers	0.004	0.005	0.006	0.014	0.011	0.016

Let's solve using the numerical solver rkf45. Then graph the populations vs time

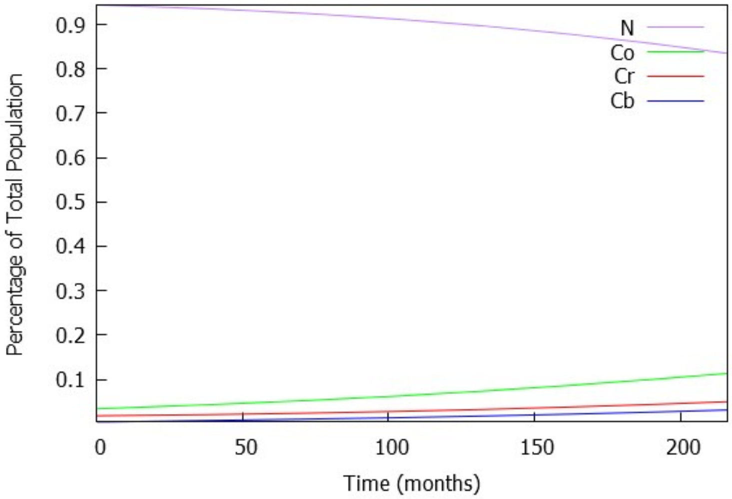
(%i908)
```
kill(N)$
```

(%i923)
```
sol:rkf45(["rhsN,"rhsCo,"rhsCr,"rhsCb],[N,Co,Cr,Cb],
     [.9440,0.0340,0.0180,0.004],[t,0,216],
      report=true,
      absolute_tolerance=1e-9,
      max_iterations=20000
     )$
```

$Info$: $rkf45$:
$Integration\,points\,selected$: 10
$Total\,number\,of\,iterations$: 9
$Bad\,steps\,corrected$: 0
$Minimum\,estimated\,error$: $1.310312912016444 10^{-14}$
$Maximum\,estimated\,error$: $5.319346769188562 10^{-10}$
$Minimum\,integration\,step\,taken$: 2.16
$Maximum\,integration\,step\,taken$: 30.62385568522025

(%i924)
```
wxdraw2d(point_type=-1,points_joined=true,
xlabel="Time (months)", ylabel="Percentage of Total Population",color=purple,
key="N",points(makelist([p[1],p[2]],p,sol)),color=green,
key="Co",points(makelist([p[1],p[3]],p,sol)),color=red,
key="Cr",points(makelist([p[1],p[4]],p,sol)),color=blue,
key="Cb",points(makelist([p[1],p[5]],p,sol)));
```

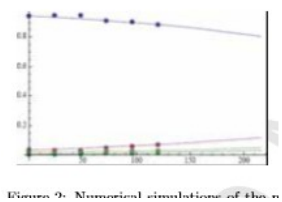
(\%{}t924)



(\%{}o924)

Our results match those published in the paper

Figure 8:



In addition to solving the system, the authors introduce a sensitivity analysis to simulate the randomness present in human populations. They do this by varying the given parameters using a Latin Hypercube Sampling algorithm.

Figure 9:

In order to perform the sensitivity analysis, let us use the technique called Latin Hypercube Sampling (LHS) to vary parameter values in the proposed model. Latin Hypercube Sampling, a type of stratified Monte Carlo sampling, is a sophisticated and efficient method for achieving equitable sampling of all input parameters simultaneously [28, 29]. Each parameter of the model can be defined as having an appropriate probability density function associated with it. It is usual to use the uniform distribution centred at deterministic parameters estimators in absence of data to inform on the distribution for a given parameter [30, 31]. Then, the model can be simulated by sampling a single value from each parameter distribution. Many samples should be taken and many simulations should be run, producing variable output values.

To vary ϵ_2, ϵ_3, ϵ_4, ϵ_5 and β, we assume that all of them follow a uniform probability distribution with support on the intervals $[0, 0.085]$, $[0.06, 0.2]$, $[0.32, 0.72]$, $[0, 4]$ and $[0, 0.19]$ respectively. The intervals for ϵ_2 and β are chosen

Here are the inital parameter values along with the previously defined beta=0.09614

Figure 10:

$$\begin{aligned}\epsilon &= \epsilon_1 \times \epsilon_2 \times \epsilon_3 \times \epsilon_4 \times \epsilon_5 = \\ &= 0.0093 \times 0.0425 \times 1/9 \times 0.52 \times 1/0.5 = 0.0000456.\end{aligned}$$

Here, I have attempted to create this algorithm to select unique variants of the parameters. The code is identical to the famous computer science "N queens" problem; The objective of "N queens" problem is to create a recursive algorithm which will determine where to place N queens on an NxN chessboard so that no two queens will threaten eachother- that is no queens share a row, column, or diagonal. This translates to the LHS (Latin Hypercube Sampling) in the sense that the random numbers generated will not share any coordinate values thus generating a unique set. This will yield a better sample than random numbers generated in the same bounds.

Here is the algorithm for 5 variants of each parameter. Then
I will average the outputs to give me a new E2,... E5. Then I will evaluate the
equation in figure 10. I will create 5 sets of these new variables
so they can be introduced at different time intervals on the graph to simulate the authors' "realizations".

In my explanations, I will continually refer to the N queens problem since it is simpler to conceptualize

```
(%i1433) /* the 5x5 array represents a 5x5 chessboard*/
array(A,5,5)$
block(
for i:1 thru 5 do(
  for j:1 thru 5 do(
     /* 0.085 here will represent the variance of the E1 parameter*/
     A[1,j]:random(0.085),
     A[2,j]:random(0.085),
     A[3,j]:random(0.085),
     A[4,j]:random(0.085),
     A[5,j]:random(0.085)
    )));
/* next the code will set each position on the chessboard to false, since it is not yet determined to be unique for
which it will mark the position as true (essentially placing a queen there)*/
solved:false$
c:makelist(false,i,1,5,1)$
r:makelist(false,i,1,5,1)$
d1:makelist(false,i,1,9,1)$
d2:makelist(false,i,1,9,1)$
t:makelist(0,i,1,5,1)$
f:makelist(0,i,1,5,1)$
place(x,A):=block(
  for i:1 thru 5 step 1 do(
    if not c[i] and not d1[(x-i)+5] and not d2[x+i-1] and not r[x] then(
       /* here the code evaluates whether the ith column, ith row, and the ith diagonal are free or non threatening
       if so, the algorithm will "place a queen" at that value by marking it true, essentially verifying that the value is unique*/
       c[i]:true,
       d1[(x-i)+5]:true,
       d2[(x+i)-1]:true,
       r[x]:true,
       t[x]:A[x,i],
       if x=5 and not solved then (
          /* since solved was previously set to false, the code will read not solved as true,
          then in the next few lines will output the unique set of values (positions of queens) */
          for j:1 thru 5 step 1 do(
              final[j]:t[j]),
          solved:true)
       /* the recursive property comes into play next, since if the algorithm gets to the 5th column and
          it is unable to find a coordinate value in the row for which the value is unique (the queen is not threatened),
          it will return to the previous row and reevalute for a new free coordinate */
       else if not x=5 then
          place(x+1,A),
       c[i]:false,
       d1[(x-i)+5]:false,
       d2[(x+i)-1]:false,
       r[x]:false,
       t[x]:0)))$

   place(1,A),
  for i:1 thru 5 step 1 do(
     f[i]:final[i],
     print(f[i]))$
E11:f[1]$
E12:f[2]$
E13:f[3]$
E14:f[4]$
E15:f[5]$
```

*done*0.031782298066689240.072479136199108400.028213020939530210.003317820008848298 0.08475821742753331 (\%{}o1419)

```
(%i1366) E11avg:mean([E11,E12,E13,E14,E15])$

(%i1383) E12avg:mean([E11,E12,E13,E14,E15])$

(%i1400) E13avg:mean([E11,E12,E13,E14,E15])$

(%i1417) E14avg:mean([E11,E12,E13,E14,E15])$

(%i1434) E15avg:mean([E11,E12,E13,E14,E15])$
```

Again for E3

(%i1518)
```
array(A,5,5)$
block(
for i:1 thru 5 do(
  for j:1 thru 5 do(
      A[1,j]:random(0.2),
      A[2,j]:random(0.2),
      A[3,j]:random(0.2),
      A[4,j]:random(0.2),
      A[5,j]:random(0.2)
     )));
solved:false$
c:makelist(false,i,1,5,1)$
r:makelist(false,i,1,5,1)$
d1:makelist(false,i,1,9,1)$
d2:makelist(false,i,1,9,1)$
t:makelist(0,i,1,5,1)$
f:makelist(0,i,1,5,1)$
place(x,A):=block(
  for i:1 thru 5 step 1 do(
     if not c[i] and not d1[(x-i)+5] and not d2[x+i-1] and not r[x] then(
         c[i]:true,
         d1[(x-i)+5]:true,
         d2[(x+i)-1]:true,
         r[x]:true,
         t[x]:A[x,i],
        if x=5 and not solved then (
           for j:1 thru 5 step 1 do(
               final[j]:t[j]),
            solved:true)
        else if not x=5 then
            place(x+1,A),
         c[i]:false,
         d1[(x-i)+5]:false,
         d2[(x+i)-1]:false,
         r[x]:false,
         t[x]:0)))$

   place(1,A),
  for i:1 thru 5 step 1 do(
      f[i]:final[i],
      print(f[i]));
E21:f[1]$
E22:f[2]$
E23:f[3]$
E24:f[4]$
E25:f[5]$
```

*done*0.16061597462184570.10880191663566970.03966018513347973 0.0536177361139237 10.168598844165845 (\%{}o1504)

done (\%{}o1513)

(%i1451) `E21avg:mean([E21,E22,E23,E24,E25])$`

(%i1468) `E22avg:mean([E21,E22,E23,E24,E25])$`

(%i1485) `E23avg:mean([E21,E22,E23,E24,E25])$`

(%i1502) `E24avg:mean([E21,E22,E23,E24,E25])$`

(%i1519) `E25avg:mean([E21,E22,E23,E24,E25])$`

E4

(%i1603)
```
array(A,5,5)$
block(
for i:1 thru 5 do(
  for j:1 thru 5 do(
      A[1,j]:random(0.65),
      A[2,j]:random(0.65),
      A[3,j]:random(0.65),
      A[4,j]:random(0.65),
      A[5,j]:random(0.65)
```

```
    )));
solved:false$
c:makelist(false,i,1,5,1)$
r:makelist(false,i,1,5,1)$
d1:makelist(false,i,1,9,1)$
d2:makelist(false,i,1,9,1)$
t:makelist(0,i,1,5,1)$
f:makelist(0,i,1,5,1)$
place(x,A):=block(
  for i:1 thru 5 step 1 do(
    if not c[i] and not d1[(x-i)+5] and not d2[x+i-1] and not r[x] then(
      c[i]:true,
      d1[(x-i)+5]:true,
      d2[(x+i)-1]:true,
      r[x]:true,
      t[x]:A[x,i],
      if x=5 and not solved then (
        for j:1 thru 5 step 1 do(
          final[j]:t[j]),
        solved:true)
      else if not x=5 then
        place(x+1,A),
      c[i]:false,
      d1[(x-i)+5]:false,
      d2[(x+i)-1]:false,
      r[x]:false,
      t[x]:0)))$
  place(1,A),
  for i:1 thru 5 step 1 do(
     f[i]:final[i],
     print(f[i]));
E31:f[1]$
E32:f[2]$
E33:f[3]$
E34:f[4]$
E35:f[5]$
```

*done*0.4122349267601710.04377085466581220.23764406280720090.11704582399049280.2242180038066705 (\%{}o1589)

done (\%{}o1598)

```
(%i1536) E31avg:mean([E31,E32,E33,E34,E35])$
(%i1553) E32avg:mean([E31,E32,E33,E34,E35])$
(%i1570) E33avg:mean([E31,E32,E33,E34,E35])$
(%i1587) E34avg:mean([E31,E32,E33,E34,E35])$
(%i1604) E35avg:mean([E31,E32,E33,E34,E35])$
```

E5

```
(%i1738) array(A,5,5)$
block(
for i:1 thru 5 do(
  for j:1 thru 5 do(
     A[1,j]:random(4.0),
     A[2,j]:random(4.0),
     A[3,j]:random(4.0),
     A[4,j]:random(4.0),
     A[5,j]:random(4.0)
    )));
solved:false$
c:makelist(false,i,1,5,1)$
r:makelist(false,i,1,5,1)$
d1:makelist(false,i,1,9,1)$
d2:makelist(false,i,1,9,1)$
t:makelist(0,i,1,5,1)$
f:makelist(0,i,1,5,1)$
place(x,A):=block(
  for i:1 thru 5 step 1 do(
    if not c[i] and not d1[(x-i)+5] and not d2[x+i-1] and not r[x] then(
      c[i]:true,
      d1[(x-i)+5]:true,
      d2[(x+i)-1]:true,
```

```
            r[x]:true,
            t[x]:A[x,i],
           if x=5 and not solved then (
               for j:1 thru 5 step 1 do(
                    final[j]:t[j]),
                solved:true)
           else if not x=5 then
                place(x+1,A),
            c[i]:false,
            d1[(x-i)+5]:false,
            d2[(x+i)-1]:false,
            r[x]:false,
            t[x]:0)))$

  place(1,A),
  for i:1 thru 5 step 1 do(
       f[i]:final[i],
       print(f[i]));
E41:f[1]$
E42:f[2]$
E43:f[3]$
E44:f[4]$
E45:f[5]$
```

*done*2.5738473646889140.76114222465296422.1242510990962782.0305337724773170.189174805702379 (\%{}o1724)

done (\%{}o1733)

```
(%i1621) E41avg:mean([E41,E42,E43,E44,E45])$
(%i1638) E42avg:mean([E41,E42,E43,E44,E45])$
(%i1655) E43avg:mean([E41,E42,E43,E44,E45])$
(%i1672) E44avg:mean([E41,E42,E43,E44,E45])$
(%i1739) E45avg:mean([E41,E42,E43,E44,E45])$
```

Next, I will evaluate the equation in figure 10 for each of these determined values.
These will become the new epsilon parameters for each realization

```
(%i1744) newepsilon1:(0.0093**E11avg*E21avg*E41avg*E31avg)/12$
         newepsilon2:(0.0093**E12avg*E22avg*E42avg*E32avg)/12$
         newepsilon3:(0.0093**E13avg*E23avg*E43avg*E33avg)/12$
         newepsilon4:(0.0093**E14avg*E24avg*E44avg*E34avg)/12$
         newepsilon5:(0.0093**E15avg*E25avg*E45avg*E35avg)/12$
```

Again for beta

```
(%i2025) B:array(B,5,5)$
         block(
         for i:1 thru 5 do(
           for j:1 thru 5 do(
               B[i,j]:random(0.19))));
         solved:false$
         c:makelist(false,i,1,5,1)$
         r:makelist(false,i,1,5,1)$
         d1:makelist(false,i,1,9,1)$
         d2:makelist(false,i,1,9,1)$
         t:makelist(0,i,1,5,1)$
         f:makelist(0,i,1,5,1)$
         place(x,B):=block(
           for i:1 thru 5 step 1 do(
              if not c[i] and not d1[(x-i)+5] and not d2[x+i-1] and not r[x] then(
                  c[i]:true,
                  d1[(x-i)+5]:true,
                  d2[(x+i)-1]:true,
                  r[x]:true,
                  t[x]:B[x,i],
                 if x=5 and not solved then (
                     for j:1 thru 5 step 1 do(
                          final[j]:t[j]),
                      solved:true)
                 else if not x=5 then
```

```
        place(x+1,B),
      c[i]:false,
      d1[(x-i)+5]:false,
      d2[(x+i)-1]:false,
      r[x]:false,
      t[x]:0)))$
  place(1,B),
  for i:1 thru 5 step 1 do(
    f[i]:final[i],
    print(f[i]));
beta1:f[1]$
beta2:f[2]$
beta3:f[3]$
beta4:f[4]$
beta5:f[5]$
```

*done*0.10282499860587750.15484461602107010.178461342075260.16142565671535120.04093371082151553 (\%{}o2011)

done (\%{}o2020)

```
(%i1958) newbeta1:mean([beta1,beta2,beta3,beta4,beta5])/12$

(%i1975) newbeta2:mean([beta1,beta2,beta3,beta4,beta5])/12$

(%i1992) newbeta3:mean([beta1,beta2,beta3,beta4,beta5])/12$

(%i2009) newbeta4:mean([beta1,beta2,beta3,beta4,beta5])/12$

(%i2026) newbeta5:mean([beta1,beta2,beta3,beta4,beta5])/12$
```

Now solving the system of equations with the new variant
parameters for the time interval 216 months and graphing.
WARNING: To obtain these results took numerous simulations.
Since my LHS used far less simulations and randomizations than the author's the results are not perfect.
Fortunately, the reproduced graphs do resemble the graphs in the paper

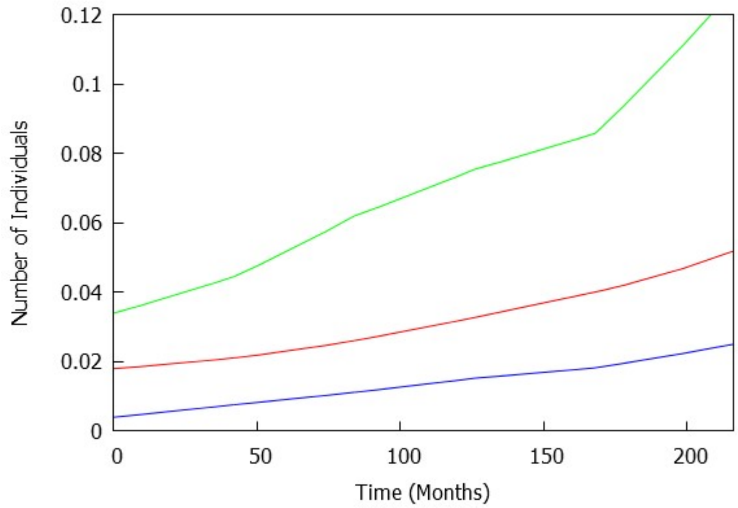
```
(%i2052) kill(t);

/*Given Equations with newvar1*/
rhsN1:mu*(N+Co+Cr+Cb)-d*N-newbeta1*((N*(Co+Cr+Cb))/(N+Co+Cr+Cb))+newepsilon1*Cb+0.000075$
rhsCo1:newbeta1*((N*(Co+Cr+Cb))/(N+Co+Cr+Cb))-dc*Co-gamma*Co$
rhsCr1:gamma*Co-dc*Cr-sigma*Cr$
rhsCb1:sigma*Cr-dc*Cb-newepsilon1*Cb$

sol1:rkf45(["rhsN1,"rhsCo1,"rhsCr1,"rhsCb1],[N,Co,Cr,Cb],
    [.9440,.0340,0.0180,0.004],[t,0,42],
     report=false,
     absolute_tolerance=1e-9,
     max_iterations=20000)$

/*Given Equations with newvar2*/
rhsN2:mu*(N+Co+Cr+Cb)-d*N-newbeta2*((N*(Co+Cr+Cb))/(N+Co+Cr+Cb))+newepsilon2*Cb+0.000075$
rhsCo2:newbeta2*((N*(Co+Cr+Cb))/(N+Co+Cr+Cb))-dc*Co-gamma*Co$
rhsCr2:gamma*Co-dc*Cr-sigma*Cr$
rhsCb2:sigma*Cr-dc*Cb-newepsilon2*Cb$

sol2:rkf45(["rhsN2,"rhsCo2,"rhsCr2,"rhsCb2],[N,Co,Cr,Cb],
     rest(last(sol1)),[t,42,84],
     report=false,
     absolute_tolerance=1e-9,
     max_iterations=20000
)$

/*Given Equations with newvar3*/
rhsN3:mu*(N+Co+Cr+Cb)-d*N-newbeta3*((N*(Co+Cr+Cb))/(N+Co+Cr+Cb))+newepsilon3*Cb+0.000075$
rhsCo3:newbeta3*((N*(Co+Cr+Cb))/(N+Co+Cr+Cb))-dc*Co-gamma*Co$
rhsCr3:gamma*Co-dc*Cr-sigma*Cr$
rhsCb3:sigma*Cr-dc*Cb-newepsilon3*Cb$

sol3:rkf45(["rhsN3,"rhsCo3,"rhsCr3,"rhsCb3],[N,Co,Cr,Cb],
     rest(last(sol2)),[t,84,126],
     report=false,
     absolute_tolerance=1e-9,
     max_iterations=20000
)$
```

```
/*Given Equations with newvar4*/
rhsN4:mu*(N+Co+Cr+Cb)-d*N-newbeta4*((N*(Co+Cr+Cb))/(N+Co+Cr+Cb))+newepsilon4*Cb+0.000075$
rhsCo4:newbeta4*((N*(Co+Cr+Cb))/(N+Co+Cr+Cb))-dc*Co-gamma*Co$
rhsCr4:gamma*Co-dc*Cr-sigma*Cr$
rhsCb4:sigma*Cr-dc*Cb-newepsilon4*Cb$

sol4:rkf45(['rhsN4,'rhsCo4,'rhsCr4,'rhsCb4],[N,Co,Cr,Cb],
      rest(last(sol3)),[t,126,168],
      report=false,
      absolute_tolerance=1e-9,
      max_iterations=20000
)$


rhsN5:mu*(N+Co+Cr+Cb)-d*N-newbeta5*((N*(Co+Cr+Cb))/(N+Co+Cr+Cb))+newepsilon5*Cb+0.000075$
rhsCo5:newbeta5*((N*(Co+Cr+Cb))/(N+Co+Cr+Cb))-dc*Co-gamma*Co$
rhsCr5:gamma*Co-dc*Cr-sigma*Cr$
rhsCb5:sigma*Cr-dc*Cb-newepsilon5*Cb$

sol5:rkf45(['rhsN5,'rhsCo5,'rhsCr5,'rhsCb5],[N,Co,Cr,Cb],
      rest(last(sol4)),[t,168,216],
      report=false,
      absolute_tolerance=1e-9,
      max_iterations=20000
)$
```

done (\%{}o2027)

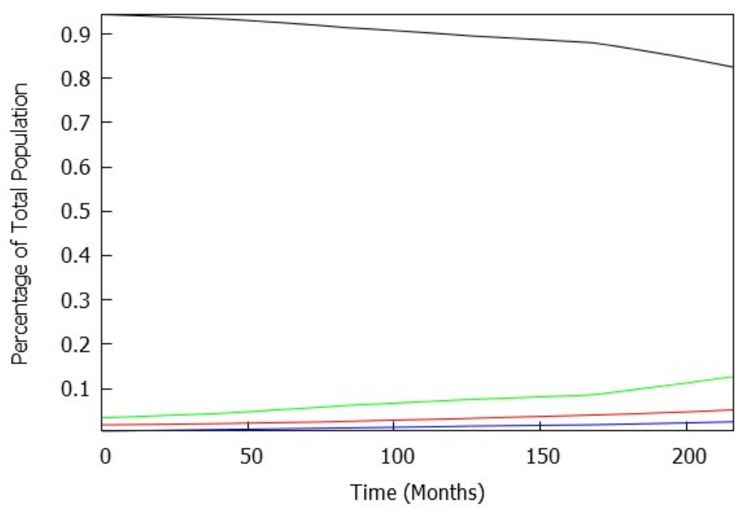
```
(%i2054) wxdraw2d(point_type=-1,points_joined=true,yrange=([0,0.12]),
   ylabel="Number of Individuals",
   xlabel="Time (Months)",color=green,
points(makelist([p[1],p[3]],p,sol1)),color=red,
points(makelist([p[1],p[4]],p,sol1)), color=blue,
points(makelist([p[1],p[5]],p,sol1)), color=green,
points(makelist([p[1],p[3]],p,sol2)),color=red,
points(makelist([p[1],p[4]],p,sol2)),color=blue,
points(makelist([p[1],p[5]],p,sol2)),color=green,
points(makelist([p[1],p[3]],p,sol3)),color=red,
points(makelist([p[1],p[4]],p,sol3)),color=blue,
points(makelist([p[1],p[5]],p,sol3)),color=green,
points(makelist([p[1],p[3]],p,sol4)),color=red,
points(makelist([p[1],p[4]],p,sol4)),color=blue,
points(makelist([p[1],p[5]],p,sol4)),color=green,
points(makelist([p[1],p[3]],p,sol5)),color=red,
points(makelist([p[1],p[4]],p,sol5)),color=blue,
points(makelist([p[1],p[5]],p,sol5)));


wxdraw2d(point_type=-1,points_joined=true,xrange=[0,216],
   ylabel="Percentage of Total Population",
   xlabel="Time (Months)",color=black,
points(makelist([p[1],p[2]],p,sol1)),color=green,
points(makelist([p[1],p[3]],p,sol1)),color=red,
points(makelist([p[1],p[4]],p,sol1)),color=blue,
points(makelist([p[1],p[5]],p,sol1)),color=black,
points(makelist([p[1],p[2]],p,sol2)),color=green,
points(makelist([p[1],p[3]],p,sol2)),color=red,
points(makelist([p[1],p[4]],p,sol2)),color=blue,
points(makelist([p[1],p[5]],p,sol2)),color=black,
points(makelist([p[1],p[2]],p,sol3)),color=green,
points(makelist([p[1],p[3]],p,sol3)),color=red,
points(makelist([p[1],p[4]],p,sol3)),color=blue,
points(makelist([p[1],p[5]],p,sol3)),color=black,
points(makelist([p[1],p[2]],p,sol4)),color=green,
points(makelist([p[1],p[3]],p,sol4)),color=red,
points(makelist([p[1],p[4]],p,sol4)),color=blue,
points(makelist([p[1],p[5]],p,sol4)),color=black,
points(makelist([p[1],p[2]],p,sol5)),color=green,
points(makelist([p[1],p[3]],p,sol5)),color=red,
points(makelist([p[1],p[4]],p,sol5)),color=blue,
points(makelist([p[1],p[5]],p,sol5)));
```

(\%{}t2053)



(\%{}o2053)

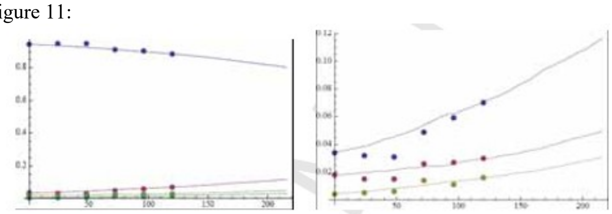
(\%{}t2054)



(\%{}o2054)

Clearly, this graph resembles the provided graphs. Unfortunately, without the proper number of realizations and computations of the LHS the data does suffer.
However, the code written above does prove that it is a reproduction of the original that could be improved with more of the same computations.

Figure 11:



Created with wxMaxima.