Vaxx Project: Source Code
(using Beta distribution based on CA county-level flu claims data)

Code created by: Myong-Hun Chang

This code is intended for private research and not for public use. As such, the code is specifically designed for the narrow set of research questions and not for mass consumption. The author will provide no support, no help debugging, and no evaluation of model output.

Directory

```
NotebookDirectory[]
```

```
C:\MyResearchTemp\Vaccination\Codex\
```

```
SetDirectory[NotebookDirectory[]];
```

Functions

Initialize the Population Grid

```
population[X_,Y_]:=
(
Z=X*Y;
toIndex[n_,m_]:=n-1+(m-1)*X; (* function for transforming (n, m) into a single inde
toCoord[x_]:={Mod[x,X]+1,Quotient[x,X]+1}; (* function for transforming an index nu
);
```

Create and Populate the Regions (Hubs)

```
hubmember[xH_, yH_] :=
  {
   Do[
    hub[a, b] = Flatten[Table[toIndex[i, j],
        {j, (b - 1) * yH + 1, b * yH}, {i, (a - 1) * xH + 1, a * xH}]],
    {a, 1, Floor[X, xH]},
    {b, 1, Floor[Y, yH]}];
   Do[
    Do[
     agentHub[k] = {i, j},
     {k, hub[i, j]}],
    {i, 1, Floor[X / xH]},
    {j, 1, Floor[Y / yH]}];
  };
```

```
contact[netSize_, pC_] :=
  (
   (* construct hub source *)
   Do[cntP[i] = DeleteCases[hub[agentHub[i][[1]], agentHub[i][[2]]], i],
    {i, 0, X * Y - 1}]; (* exclude oneself from one's own hub *)
   (* construct actual network *)
   Do[cntA[i] = {}, {i, 0, X * Y - 1}];
   avail = Table[i, {i, 0, X * Y - 1}];
   totLen = 0; avgLen = 0;
   invZ = Table[i, {i, 0, X * Y - 1}];
   While[avgLen < netSize,
    trial1 = RandomChoice[avail];
    indp = RandomReal[];
    trial2 = Which[0 ≤ indp ≤ pC, RandomChoice[cntP[trial1]],
      pC < indp ≤ 1, RandomChoice[Complement[invZ,
        hub[agentHub[trial1][[1]], agentHub[trial1][[2]]]]]];
If[FreeQ[cntA[trial1], trial2],
cntA[trial1] = Join[cntA[trial1], {trial2}];
cntA[trial2] = Join[cntA[trial2], {trial1}];
totLen = totLen + 2;
avgLen = totLen / (X * Y);
    ];
   ];
  );
```

Parameters

```
X = 100; Y = 100; (* z (= X*Y): population of 10,000 agents *)
xH = 10; yH = 10; (* H = xH*yH: 100 regions (hubs) *)
netSize = 20; (* c̄: average size of the contact networks *)
nseed = 10; (* number of seed agents who are initially infected *)
pC = 0.95; (* p^C: local specificity of contact networks *)
S = 100; (* s̃: number of seasons *)
nB = 1; (*κ: multiple for beta distribution parameters*)
```

Population and the Hub (Region) Membership

```
population[X, Y];
```

```
hubmember[xH, yH];
```

Epidemics

```
dx = 0; (* data file index *)
```

Vaccination rates for CA counties

```
V = {46.63, 20.86, 49.86, 40.34, 40.50, 40.99, 48.12, 38.14, 49.45, 46.50,
     35.93, 40.52, 33.53, 43.51, 37.75, 36.97, 28.54, 30.47, 41.86, 39.79,
     51.93, 31.02, 27.97, 42.72, 16.86, 31.82, 45.13, 45.52, 49.46, 49.71,
     52.95, 21.60, 40.94, 47.16, 50.17, 32.36, 43.49, 44.32, 45.23, 49.79,
     52.17, 49.25, 53.08, 51.29, 43.04, 20.11, 30.96, 37.76, 48.11,
     44.36, 51.33, 42.22, 30.24, 43.65, 46.75, 48.11, 52.58, 41.66} / 100;
```

Beta distribution shape parameters based on CA data

```
α = (((Mean[V])^2 * (1 - Mean[V])) / Variance[V]) - Mean[V];
β = ((Mean[V] * (1 - Mean[V])^2) / Variance[V]) - (1 - Mean[V]);
```

Infection characteristics

```
beta = 0.02; (* β: infection probability *)
tauE = 2; (* length of the latent period *)
tauI = 7; (* length of the infectious period *)
```

Contagion dynamics

state[s, t, i]: infection state of agent i in period t of season s

= 0 if vaccinated and immune (V, R);

= 1 if unvaccinated and susceptible (NV, S);

= 2 if unvaccinated and exposed (NV, E);

= 3 if unvaccinated and infectious (NV, I);

= 4 if unvaccinated and recovered (NV, R);

```
stInt = {}; stFin = {}; contactN = {};

Do[
  SeedRandom[s];
  (*contact network*)
  contact[netSize, pC];
  comNet = Table[Map[UndirectedEdge[i, #] &, cntA[i]], {i, 0, X * Y - 1}];
  dirNet = Table[Map[i → # &, cntA[i]], {i, 0, X * Y - 1}];
  pl[s] = MeanGraphDistance[Flatten[dirNet]] // N;
  (*vaccination rates*)
  rV = RandomVariate[BetaDistribution[nB * α, nB * β], 100];
  (*accination rates for the hubs *)
  Do[nV[k] = rV[[10 * (agentHub[k][[2]] - 1) + agentHub[k][[1]]]],
   {k, 0, X * Y - 1}]; (*individual's vaccination rate*)
  (*epidemic*)
  SeedRandom[s];
  t = 1;
  Do[y[k] = RandomChoice[{nV[k], 1 - nV[k]} → {0, 1}], {k, 0, X * Y - 1}];
  (*individual's vaccination status*)
  init = Flatten[Table[{y[k]}, {k, 0, X * Y - 1}]];
  seed = RandomSample[Position[init, 1], nseed];
  state[s, t] = ReplacePart[init, seed → 2];
  tinf = ReplacePart[Table[0, {i, 1, X * Y}], seed → 1];
  stemp = state[s, t];
  numV[s, t] = Count[stemp, 0]; (* number of vaccinated agents in t *)
  numS[s, t] = Count[stemp, 1]; (* number of susceptible agents in t *)
  numE[s, t] = Count[stemp, 2]; (* number of exposed agents in t *)
  numI[s, t] = Count[stemp, 3]; (* number of infectious agents in t *)
  numR[s, t] = Count[stemp, 4]; (* number of recovered agents in t *)
  t = t + 1;
  While[numE[s, t - 1] > 0 || numI[s, t - 1] > 0,
   q[s, t] = Table[
     1 - (1 - beta) ^Count[state[s, t - 1][[cntA[i] + 1]], 3], {i, 0, X * Y - 1}];
   randQ = Table[RandomReal[], {i, 1, X * Y}];
   infected = Intersection[
     Position[state[s, t - 1], 1], Position[randQ - q[s, t], n_ /; n ≤ 0]];
   stemp = ReplacePart[stemp, infected → 2];
   tinf = ReplacePart[tinf, infected → t];
   infectious = Intersection[Position[state[s, t - 1], 2],
```

```
    Position[tinf, n_ /; n > 0], Position[t - tinf, n_ /; n ≥ tauE]];
  stemp = ReplacePart[stemp, infectious → 3];
  recovered = Intersection[Position[state[s, t - 1], 3],
    Position[tinf, n_ /; n > 0], Position[t - tinf, n_ /; n ≥ tauE + tauI]];
  stemp = ReplacePart[stemp, recovered → 4];
  numV[s, t] = Count[stemp, 0];
  numS[s, t] = Count[stemp, 1];
  numE[s, t] = Count[stemp, 2];
  numI[s, t] = Count[stemp, 3];
  numR[s, t] = Count[stemp, 4];
  state[s, t] = stemp;
  t++];
 final[s] = t - 1;
 (* save state and network data *)
 stInt = Append[stInt, state[s, 1]];
 (* initial state of the population in t=1 *)
 stFin = Append[stFin, state[s, final[s]]];
 (* final state of the population at the end of epidemic *)
 contactN = Append[contactN, Table[cntA[i], {i, 0, X * Y - 1}]];
 (* contact networks for population *)
 , {s, 1, S}];
```

Export the simulation output

```
vx = Table[Table[numV[k, i], {i, 1, final[k]}], {k, 1, S}];
sx = Table[Table[numS[k, i], {i, 1, final[k]}], {k, 1, S}];
ex = Table[Table[numE[k, i], {i, 1, final[k]}], {k, 1, S}];
ix = Table[Table[numI[k, i], {i, 1, final[k]}], {k, 1, S}];
rx = Table[Table[numR[k, i], {i, 1, final[k]}], {k, 1, S}];

Export["dataCAB" <> ToString[dx] <> "/final.dat",
  Table[final[i], {i, 1, S}], "WDX"];
Export["dataCAB" <> ToString[dx] <> "/pl.dat", Table[pl[i], {i, 1, S}], "WDX"];
Export["dataCAB" <> ToString[dx] <> "/vx.dat", vx, "WDX"];
Export["dataCAB" <> ToString[dx] <> "/sx.dat", sx, "WDX"];
Export["dataCAB" <> ToString[dx] <> "/ex.dat", ex, "WDX"];
Export["dataCAB" <> ToString[dx] <> "/ix.dat", ix, "WDX"];
Export["dataCAB" <> ToString[dx] <> "/rx.dat", rx, "WDX"];
Export["dataCAB" <> ToString[dx] <> "/stInt.dat", stInt, "WDX"];
Export["dataCAB" <> ToString[dx] <> "/stFin.dat", stFin, "WDX"];
Export["dataCAB" <> ToString[dx] <> "/contactN.dat", contactN, "WDX"];
```