Model algorithm

Constructing the matrix

For each individual, we defined a row vector comprising of the exposure day, an incubation period and the symptoms period. Every element in the vector represents a single day. The date of exposure was marked with a "− 1", and the symptoms period was assigned the values of the relative infectiousness, $f_i^S(t)$. An example of such a vector:

$(-1, 0, 0, 0, 0, 0, 0, 0, 208.14, 559.764, 1169.51, 1964.58, 2800.02, 1804.77, 1736.56, 1201.36, 352.745)$

Visually, this may be represented as shown in Supplement 9.

This vector was then embedded onto the time axis using the dates of the symptomatic periods, according to the data. The relative infectiousness is a function of the viral load, $g(V_i^S(t))$, combined with the individuals’ contacts, $C_i(t)$, and is presented in the Main text, Eq. (2).

If the number of contacts of an individual $i$ was known, it was used to parameterize $C_i$. When the number of contacts was unknown, we sampled from a probability distribution, $f(C)$, constructed from the contact-tracing data: $C_i \sim f(C)$ (Supplement 2).

In each iteration of the stochastic simulation, we sampled values for $RR^{Late \ Phase}$ and $RR^{Early \ Phase}$. Using Eq. (2) in Supplement 3, we calculated $r$, and from the relation $g(V^S(t)) = r \log(V^S(t))$, we assigned each individual with their viral load contribution to the relative infectiousness. In the stochastic simulation, the variables were sampled from distributions (Main text, Table 1). A vector was assigned to each individual, and embedding the vectors in time yielded a matrix (Supplement 5).
Assigning transmission

Using the above matrix, each exposed case was probabilistically assigned with the contact from whom they were infected. In each day, the numbers of exposed and symptomatic individuals were in the same column in the matrix. Symptomatic individuals were considered potential infectors, and were probabilistically assigned as the source of transmission to their infected contact based on their relative infectiousness, $f_i^S(t)$ (see Supplement 10).

We used the relative infectiousness to determine probabilities of being the infector, normalizing over all possible infectors of that day (Main text, Equation (1)).

Testing intervention strategies

In every iteration the overall $R_0$ was calculated, and at the end of each of the simulations, the probability of disease elimination was calculated:

$$Pr(R_0 < 1) = \frac{N(R_0<1)}{N},$$

(1)

where $N(R_0<1)$ denotes the number of iterations in which $R_0 < 1$ was obtained, and $N$ is the overall number of iterations.

Quarantine of infected individuals

Two parameters were varied over this analysis: 1) number of days that have passed since symptoms onset and until recovery/death. 2) the coverage of population on which the policy is successfully implemented.
For every iteration, upon successful quarantine of individuals, all of the individuals' vectors were set to zero, and the $R_0$ was calculated (Main text, equation (3)). This procedure was repeated for each day of the timeframe considered (Supplement 7). To vary the coverage of quarantine, the corresponding proportion of the population’s individuals’ vectors were set to zero. This was repeated with different values of coverage, implemented by sampling a random number, $r \sim \text{Uniform}(0, 1)$, and setting to zero the individuals' vector only if $r < C$, with $C$ representing the tested coverage, varying from half the population ($C = 0.5$) to full coverage ($C = 1$). This stochastic process was conducted separately for the non-survivors (Main text, Figure 2A) and for all infected individuals (Supplement 7).

**Reducing number of contacts**

Reducing the contacts of the infected individuals was implemented by decreasing the probability of being selected as the infector. Eq. (1) in the Main text transforms into:

$$\hat{P}_{ij}(t) = I_j(t) \frac{\gamma f_i^s(t)}{\gamma \sum_{k=1}^n f_k^s(t) + (1 - \gamma)}, \quad (2)$$

where $\gamma$ is the proportion of contacts, $0 < \gamma < 1$, with $\gamma = 1$ implying no intervention, and $\gamma = 0$ is complete isolation of the individual from all contacts. This re-normalization lowers the chance of an individual to be chosen as the infector, which is effectively the same as reducing the number of contacts (Eq. (2), Main text).

**Legends**

Supplement 9. Disease Progression for an Infected Individual
Supplement 10. Assigning Transmission
Mathematica 9.0 code

Impact of Ebola disease progression on transmission and control in Liberia

Initialization
init[] :=
{
  SetDirectory[NotebookDirectory[]];
days = DateRange[{2014, 5, 1}, {2014, 12, 1}];
} (*init*)

Data
(*In "id" vector, "ext" represents cases from WHO. Numbers are coded cases collected by LMoHS*)

1.2, 
134., 217., 225., 254., 257., 
34., 
147., 178., 
27., 
27., 
Viral loads - RNA copies
fatalMean={4.5, 7.2, 7.9, 8.3, 8.5, 8.2, 8.7, 7.4, 5.9, 6.8, 5.7, 4.9, 4.4};
nonFatalMean={5.0, 7.8, 0.6, 0.2, 0.3, 0.2, 0.4, 0.3, 0.6, 0.9, 0.9, 0.4, 0.4, 0.4};
fatalSD={0.7, 0.8, 0.6, 0.2, 0.3, 0.2, 0.4, 0.3, 0.6, 0.9, 0.9, 0.4, 0.4, 0.4};
nonFatalSD={0.4, 1.0, 0.4, 0.6, 0.4, 0.5, 0.8, 0.7, 0.7, 0.2, 0.1, 0.1, 0.1, 0.1};

Constructing "disease vector"
constructSickVector[] :=
{ scaledViralLoadDead = viralLoadDead[[1 ;; sickLengthDead]]; scaledViralLoadAlive = viralLoadAlive[[1 ;; sickLengthAlive]]; contactCalibrationDead = Table[1, {i, 1, sickLengthDead}]; contactCalibrationAlive = Table[1, {i, 1, sickLengthAlive}];
  contactCalibrationDead[[1 ;; sickLengthDead - stageCutoff]] = contactFirstStageDead; contactCalibrationAlive[[1 ;; sickLengthDead - stageCutoff]] = contactFirstStageAlive;
  contactCalibrationDead[[sickLengthDead - stageCutoff + 1 ;; sickLengthDead]] = contactSecondStageDead; contactCalibrationAlive[[sickLengthDead - stageCutoff ;; sickLengthDead]] = contactSecondStageAlive;
  sickVectorDead = Join[{-1}, Table[0, {i, 1, incubationDead}], scaledViralLoadDead * contactCalibrationDead]; sickVectorAlive = Join[{-1}, Table[0, {i, 1, incubationAlive}], scaledViralLoadAlive * contactCalibrationAlive];
} (*constructSickVector*)

Constructing matrix
constructMatrix[] :=
{ ebolaMatrix = Table[0, {i, 1, Length[infectionDate]}, {j, 1, Length[days] + maxIncubationPeriod + maxSickPeriod + 1}];
  symptomDateIndex = Table[0, {i, 1, Length[infectionDate]}]; (*For later use*)
  For[1, i <= Length[infectionDate], i++,
    index = Position[days, infectionDate[[i]]][[1, 1]]; If[status[[i]] == "Dead",
      ebolaMatrix[[i]][index + 2 ;; index + 1]] = sickVectorsForMatrix[[i]]; symptomDateIndex[[i]] = index + 2 - sickLengthForMatrix[[i]]; ebolaMatrix[[i]][index + 2 ;; index + 2 + incubationDeadForMatrix[[i]]] = sickVectorsForMatrix[[i]]; symptomDateIndex[[i]] = index + 1; ];

  infectedByDay = Table[0, {i, 1, Length[days]}]; (*Overall infected every day*)
  For[1, j <= Length[days], j++,
    infectedByDay[[j]] = -1*Total[Cases[Transpose[ebolaMatrix][[j]], x_ /; x < 0]]; ];

  rMatrixTEMP = ebolaMatrix; (*Zeroing negative elements*)
  For[1, i <= Length[ebolaMatrix], i++,
    For[1, j <= Length[ebolaMatrix][[1]], j++,
      If[ebolaMatrix[[i, j]] < 0,
        rMatrixTEMP[[i, j]] = 0]; ];

  rMatrix = Table[0, {i, 1, Length[infectionDate]}, {j, 1, Length[days]}]; (*Initialize zero matrix*)
  For[1, j <= Length[days], j++,
    If[Count[Transpose[rMatrixTEMP][[j]], x_ /; x > 0] == 0,
      Return,
      If[coverage! = 0 & RandomReal[] < coverage, probSelected = RandomChoice[Join[{-1, propContacts}, propContacts]/Total[Transpose[rMatrixTEMP][[j]]]]*Transpose[rMatrixTEMP][[j]]] - >Join[{-1}, Range[Length[Transpose[rMatrixTEMP][[j]]]]], infectedByDay[[j]]],
      probSelected = RandomChoice[Transpose[rMatrixTEMP][[j]]] - >Range[Length[Transpose[rMatrixTEMP][[j]]], infectedByDay[[j]]]]; ];
  For[k = 1, k <= Length[probSelected], k++,
    If[probSelected[[k]] == -1,
      Return,
rMatrix[[probSelected[[k]],j]]+=1
;
;
;
;
cropTop=1;
cropBottom=Length[ebolaMatrix];(*Counted from top*)
cropLeft=Min[Table[Position[ebolaMatrix[[row]],-1],{row,1,Length[id]}]]
cropRight=Max[Table[Position[ebolaMatrix[[row]],-1],{row,1,Length[id]}]];(*Counted from left*)
rMatrixCrop=rMatrix[[cropTop;;cropBottom,cropLeft;;cropRight]];

rPerDayAll={};
rPerDayAlive={};
rPerDayDead={};
rAlive={};
rDead={};

For[i=cropTop,i<=cropBottom,i++,
If[cropLeft<= symptomDateIndex[[i]]&&symptomDateIndex[[i]]+sickLengthForMatrix[[i]]<=cropRight,

rPerDayAll=Insert[rPerDayAll,rMatrix[[i,symptomDateIndex[[i]];;symptomDateIndex[[i]]+sickLengthForMatrix[[i]]-1]],-1];
If[status[[i]]=="Dead",

rPerDayDead=Insert[rPerDayDead,rMatrix[[i,symptomDateIndex[[i]];;symptomDateIndex[[i]]+sickLengthForMatrix[[i]]-1]],-1];

rDead=Join[rDead,{Total[rMatrix[[i,symptomDateIndex[[i]];;symptomDateIndex[[i]]+sickLengthForMatrix[[i]]-1]]}]
];
];
]

rAll=Join[rAlive,rDead];
rPerDayAll=Flatten[rPerDayAll,1];
rPerDayDead=Flatten[rPerDayDead,1];
rPerDayAlive=Flatten[rPerDayAlive,1];
(*constructMatrix*)
Constructing "visual" matrix
constructVisualMatrix[]:=
{ visualEbolaMatrixTEMP=ebolaMatrix[[cropTop;;cropBottom,cropLeft;;cropRight+maxIncubationPeriod +maxSickPeriod]];

(*indexOfPeopleInfected=Table[0,{day,1,Length[visualEbolaMatrixTEMP[[1]]]}];*)
visualEbolaMatrix=Table[0,{day,1,Length[visualEbolaMatrixTEMP[[1]]]}];

For[day=1,day<=Length[visualEbolaMatrixTEMP[[1]]],day++,

indexOfPeopleInfected=Flatten[Position[Transpose[visualEbolaMatrixTEMP][[day]],-1]];

visualEbolaMatrix[[day]]=Total[visualEbolaMatrixTEMP[[indexOfPeopleInfected]]];
If[visualEbolaMatrix[[day]]==0,
visualEbolaMatrix[[day]]=Table[0,{i,1,Length[visualEbolaMatrixTEMP[[1]]]}];
];
];
cutBottom=Length[visualEbolaMatrix];

While[visualEbolaMatrix[[cutBottom]]== Table[0,{r,1,Length[visualEbolaMatrix[[cutBottom]]]]],
cutBottom-=1
];
cutRight=Length[Transpose[visualEbolaMatrix]];

While[Transpose[visualEbolaMatrix][[cutRight]]==
Table[0,{r,1,Length[Transpose[visualEbolaMatrix][[cutRight]]]}],
cutRight-=1
];

visualEbolaMatrixCropped=visualEbolaMatrix[[1;;cutBottom,1;;cutRight]];
plotVisualMatrix=MatrixPlot[visualEbolaMatrixCropped,ImageSize->Large];
totalVisualInfections=Table[0,{j,1,Length[Transpose[visualEbolaMatrixCropped]]}];
totalVisualViralLoad=Table[0,{j,1,Length[Transpose[visualEbolaMatrixCropped]]}];

For[j=1,j<=Length[Transpose[visualEbolaMatrixCropped]],j++,

totalVisualInfections[[j]] = Total[Cases[Transpose[visualEbolaMatrixCropped][[j]], x_ /; x < 0]]

visualLines = {totalVisualViralLoad, Table[0, {j, 1, Length[Transpose[visualEbolaMatrixCropped]]}], totalVisualInfections}

plotVisualLines = MatrixPlot[visualLines, ImageSize -> Large];

(*constructVisualMatrix*)

Relative risk calibration

generateR[viralLoadIN_] :=
   {RcoefOUT = 0;
    While[RcoefOUT <= 1,
     viralLoadIN = Max[1, Exp[RandomVariate[NormalDistribution[Log[1.2], 0.415]]]/meal;
     RcoefOUT = RcoefOUT^((1/((52*Sum[fatalMean[[i]], {i, 5, 10}])/6 + 3*Sum[nonFatalMean[[1]], {i, 1, 4}])/4 + 3*Sum[nonFatalMean[[1]], {i, 1, 4}]/4)/55 - ((52*Sum[fatalMean[[i]], {i, 1, 4}])/4 + 3*Sum[nonFatalMean[[1]], {i, 1, 4}]/4)/55));
     meal = Max[1, Exp[RandomVariate[NormalDistribution[Log[3.9], 0.58]]]]/talk;
     RcoefOUT = RcoefOUT^((1/((52*Sum[fatalMean[[i]], {i, 5, 10}])/6 + 3*Sum[nonFatalMean[[1]], {i, 1, 4}])/4 + 3*Sum[nonFatalMean[[1]], {i, 1, 4}]/4)/55 - ((52*Sum[fatalMean[[i]], {i, 1, 4}])/4 + 3*Sum[nonFatalMean[[1]], {i, 1, 4}]/4)/55));
     talk = Max[1, Exp[RandomVariate[NormalDistribution[Log[0.7], 0.525]]]]/bed;
     RcoefOUT = RcoefOUT^((1/((52*Sum[fatalMean[[i]], {i, 5, 10}])/6 + 3*Sum[nonFatalMean[[1]], {i, 1, 4}])/4 + 3*Sum[nonFatalMean[[1]], {i, 1, 4}]/4)/55));
    ];
   ];

RcoefOUT  (*generateR*)

Analysis

Defining distributions

define[iterationsIN_] :=
   {numOfIterations = iterationsIN;
    maxSickPeriod = 14;
    minSickPeriod = 5;
    maxIncubationPeriod = 5;
    minIncubationPeriod = 5;
    incubationDist = Round[RandomVariate[TriangularDistribution[{minIncubationPeriod - 0.5, maxIncubationPeriod + 0.5}, 8], {numOfIterations, Length[id]}]]; 
    sickPeriodDist = Round[RandomVariate[TriangularDistribution[{minSickPeriod - 0.5, maxSickPeriod + 0.5}, 8], {numOfIterations, Length[id]}]]; 
    contactFirstStageDist = Round[RandomVariate[HistogramDistribution[numContacts], {numOfIterations, Length[id]}]]; 
    contactSecondStageDist = Round[RandomVariate[TruncatedDistribution[{0.5, 5.5}, HistogramDistribution[numContacts]], {numOfIterations, Length[id]}]]; 
    stageCutoffDist = Round[RandomVariate[UniformDistribution[{0.5, 5.5}], {numOfIterations, Length[id]}]];
   ];

Rcoef = Table[generateR[viralLoad], {i, 1, numOfIterations}];

viralLoadFatalDist = Table[RandomVariate[TruncatedDistribution[{0, Infinity}, NormalDistribution[fatalMean[[1]], fatalSD[[i]]]], {k, 1, Length[id]}], {j, 1, numOfIterations}];

viralLoadNonFatalDist = Table[RandomVariate[TruncatedDistribution[{0, Infinity}, NormalDistribution[nonFatalMean[[1]], nonFatalSD[[1]]]], {k, 1, Length[id]}], {j, 1, numOfIterations}];

(*define*)

Single draw from distributions

singleDraw =
   {Rcoef = generateR[viralLoad];
    define[1];
    coverage = 0;
    propContacts = 1;
sickVectorsForMatrix = Table[0, {i, 1, Length[id]}];
sickLengthForMatrix = Table[0, {i, 1, Length[id]}];
incubationAliveForMatrix = Table[0, {i, 1, Length[id]}];
iter = 1;
For[person = 1, person <= Length[id], person++,
    incubationDead = incubationDist[[iter, person]];
    incubationAlive = incubationDead;
    sickLengthDead = sickPeriodDist[[iter, person]];
    sickLengthAlive = sickLengthDead;
    contactFirstStageDead = contactFirstStageDist[[iter, person]];
    contactSecondStageDead = contactSecondStageDist[[iter, person]];
    contactFirstStageAlive = contactFirstStageDist[[iter, person]];
    contactSecondStageAlive = contactFirstStageAlive;
    stageCutoff = stageCutoffDist[[iter, person]]; (* Days counted from end *)
    viralLoadDead = Rcoef[iter]^viralLoadFatalDist[[iter, person]];
    viralLoadAlive = Rcoef[iter]^viralLoadNonFatalDist[[iter, person]];
    constructSickVector[];
    If[status[[person]] == "Dead",
        sickVectorsForMatrix[[person]] = sickVectorDead;
        sickLengthForMatrix[[person]] = sickLengthDead,
        sickVectorsForMatrix[[person]] = sickVectorAlive;
        sickLengthForMatrix[[person]] = sickLengthAlive;
        incubationAliveForMatrix[[person]] = incubationAlive;];
    ];
constructMatrix[];
rAllSingle = rAll;
rDeadSingle = rDead;
rAliveSingle = rAlive;
(*singleDraw*)
Sensitivity analysis (Iterations, Coverage, Proportion of contacts)
sensAnal[iterationsIN_, coverageIN_, propContactsIN_] := 
{
    coverage = coverageIN;
    propContacts = propContactsIN;
    define[iterationsIN];
rAllSensitivity = Table[0, {i, 1, numOfIterations}];
rDeadSensitivity = Table[0, {i, 1, numOfIterations}];
rAliveSensitivity = Table[0, {i, 1, numOfIterations}];
rPerDayAllSensitivity = Table[0, {i, 1, numOfIterations}];
rPerDayDeadSensitivity = Table[0, {i, 1, numOfIterations}];
rPerDayAliveSensitivity = Table[0, {i, 1, numOfIterations}];
    personPickedIndex = Table[[], {i, 1, numOfIterations}];
    personPickedRVector = Table[[], {i, 1, numOfIterations}];
    For[iter = 1, iter <= numOfIterations, iter++,
        sickVectorsForMatrix = Table[0, {i, 1, Length[id]}];
        sickLengthForMatrix = Table[0, {i, 1, Length[id]}];
        incubationAliveForMatrix = Table[0, {i, 1, Length[id]}];
        tracedCount = 1;
        For[person = 1, person <= Length[id], person++,
            incubationDead = incubationDist[[iter, person]];
            incubationAlive = incubationDead;
            sickLengthDead = sickPeriodDist[[iter, person]];
            sickLengthAlive = sickLengthDead;
            If[NumberQ[id[[person]]],
            ];
contactFirstStageDead = numContacts[[tracedCount]]; 
contactSecondStageDead = Round[RandomVariate[TruncatedDistribution[{0.5, Min[{numContacts[[tracedCount]] + 0.5, 5.5}], HistogramDistribution[numContacts]}]]]; 
contactFirstStageAlive = numContacts[[tracedCount]]; 
contactSecondStageAlive = numContacts[[tracedCount]]; 
tracedCount++;

contactFirstStageDead = contactFirstStageDist[[iter, person]]; 
contactSecondStageDead = contactSecondStageDist[[iter, person]]; 
contactFirstStageAlive = contactFirstStageDist[[iter, person]]; 
contactSecondStageAlive = contactFirstStageAlive; 

stageCutoff = stageCutoffDist[[iter, person]]; (*Days counted from end*) 
viralLoadDead = Rcoef[[iter]]^viralLoadFatalDist[[iter, person]]; 
viralLoadAlive = Rcoef[[iter]]^viralLoadNonFatalDist[[iter, person]]; 

constructSickVector[];

If[status[[person]] == "Dead", 
sickVectorsForMatrix[[person]] = sickVectorDead; 
sickLengthForMatrix[[person]] = sickLengthDead, 
sickVectorsForMatrix[[person]] = sickVectorAlive; 
sickLengthForMatrix[[person]] = sickLengthAlive; 
incubationAliveForMatrix[[person]] = incubationAlive ]; 

constructMatrix[];

rand = RandomInteger[{cropTop, cropBottom}]; 
While[Not[cropLeft <= symptomDateIndex[[rand]] && symptomDateIndex[[rand]] + sickLengthAlive - 1 <= cropRight], 
rand = RandomInteger[{cropTop, cropBottom}]; ]; 

personPickedIndex[[iter]] = rand; 
If[status[[rand]] == "Dead", 
personPickedRVector[[iter]] = rMatrix[[rand, symptomDateIndex[[rand]] ;; symptomDateIndex[[rand]] + sickLengthDead - 1]], 
personPickedRVector[[iter]] = rMatrix[[rand, symptomDateIndex[[rand]] ;; symptomDateIndex[[rand]] + sickLengthAlive - 1]]; ]; 

rAllSensitivity[[iter]] = rAll; 
rDeadSensitivity[[iter]] = rDead; 
rAliveSensitivity[[iter]] = rAlive; 
rPerDayAllSensitivity[[iter]] = rPerDayAll; 
rPerDayDeadSensitivity[[iter]] = rPerDayDead; 
rPerDayAliveSensitivity[[iter]] = rPerDayAlive; ]; (*sensAnal*)

Secondary cases distribution (one person per iteration) 
secondaryCases[] := 
{ 
personPickedDeadRVector = {}; 
personPickedAliveRVector = {}; 
For[i = 1, i <= Length[personPickedIndex], i++, 
If[status[[personPickedIndex[[i]]]] == "Dead", 
personPickedDeadRVector = Insert[personPickedDeadRVector, personPickedRVector[[i]], -1]; 
personPickedAliveRVector = Insert[personPickedAliveRVector, personPickedRVector[[i]], -1]; ]; ]; 
Print["Overall ", Length[personPickedIndex], " picked. "];
Print["Survived: ", Length[personPickedAliveRVector], ", Deceased: ", Length[personPickedDeadRVector]]; 
plotPersonPickedBothRVectorHist = Histogram[Table[Total[personPickedDeadRVector[[i]]], {i, 1, Length[personPickedDeadRVector]}], Table[Total[personPickedAliveRVector[[i]]], {i, 1, Length[personPickedAliveRVector]}]], 

Downloaded From: http://annals.org/pdfaccess.ashx?url=/data/journals/aim/931895/ on 06/21/2017
Secondary cases per day (one person per iteration)
secondaryCasesPerDay[] :=
    personPickedDeadPerDay[] = Table[0, {i, 1, Length[Flatten[personPickedDeadRVector[], 2]]}];
    confIntDeadPerDay[] = Table[{0, 0}, {i, 1, Length[Flatten[personPickedDeadRVector[], 2]]}];
    For[i = 1, i <= Length[personPickedDeadPerDay[]], i++,
        personPickedDeadPerDay()[[i]] = Mean[Flatten[personPickedDeadRVector[], 2][[i]]];
        confIntDeadPerDay()[[i, 1]] = Sort[Flatten[personPickedDeadRVector[], 2][[i]]][Round[0.975*Length[Flatten[personPickedDeadRVector[], 2][[i]]] + 1];
        confIntDeadPerDay()[[i, 2]] = Sort[Flatten[personPickedDeadRVector[], 2][[i]]][Max[Round[0.025*Length[Flatten[personPickedDeadRVector[], 2][[i]]] + 1]]];
    ];

    personPickedAlivePerDay[] = Table[0, {i, 1, Length[Flatten[personPickedAliveRVector[], 2]]}];
    confIntAlivePerDay[] = Table[{0, 0}, {i, 1, Length[Flatten[personPickedAliveRVector[], 2]]}];
    For[i = 1, i <= Length[personPickedAlivePerDay[]], i++,
        personPickedAlivePerDay()[[i]] = Mean[Flatten[personPickedAliveRVector[], 2][[i]]];
        confIntAlivePerDay()[[i, 1]] = Sort[Flatten[personPickedAliveRVector[], 2][[i]]][Round[0.975*Length[Flatten[personPickedAliveRVector[], 2][[i]]] + 1];
        confIntAlivePerDay()[[i, 2]] = Sort[Flatten[personPickedAliveRVector[], 2][[i]]][Max[Round[0.025*Length[Flatten[personPickedAliveRVector[], 2][[i]]] + 1]]];
    ];

    Print["Secondary cases total - Deceased: ", N[Total[personPickedDeadPerDay[]]]];
    Print["Secondary cases total - Survived: ", N[Total[personPickedAlivePerDay[]]]];

    errorPerDayDeadHi[] = Table[confIntDeadPerDay()[[i, 1]], {i, 1, Length[confIntDeadPerDay[]]}] - personPickedDeadPerDay[];
    errorPerDayDeadLo[] = Table[confIntDeadPerDay()[[i, 2]], {i, 1, Length[confIntDeadPerDay[]]}] - personPickedDeadPerDay[];
    errorPerDayAliveHi[] = Table[confIntAlivePerDay()[[i, 1]], {i, 1, Length[confIntAlivePerDay[]]}] - personPickedAlivePerDay[];
    errorPerDayAliveLo[] = Table[confIntAlivePerDay()[[i, 2]], {i, 1, Length[confIntAlivePerDay[]]}] - personPickedAlivePerDay[];

    Needs["ErrorBarPlots`"];
    confIntDead[] = Table[ErrorBar[errorPerDayDeadLo[], errorPerDayDeadHi[]], {i, 1, Length[personPickedDeadPerDay[]]}];
    confIntAlive[] = Table[ErrorBar[errorPerDayAliveLo[], errorPerDayAliveHi[]], {i, 1, Length[personPickedAlivePerDay[]]}];

    plotSecondaryPerDay[] = GraphicsGrid[{{ErrorListPlot[Table[personPickedAlivePerDay[], confIntAlive[]], PlotLabel -> "Secondary cases per day - Survived", PlotRange -> {{0, 4}, PlotStyle -> Red, AxesOrigin -> {0, 0}}], ErrorListPlot[Table[personPickedDeadPerDay[], confIntDead[]], PlotLabel -> "Secondary cases per day - Deceased", PlotRange -> {{0, 4}, PlotStyle -> Blue, AxesOrigin -> {0, 0}}]]}, ImageSize -> 1000];
}

("secondaryCasesPerDay")
Removing from all population
removeAll[] :=
{ rPerPersonAllSensitivity = Table[Flatten[rPerDayAllSensitivity[[i]], 2]], {i, 1, Length[rPerDayAllSensitivity]}];

rUntilDayAll = Table[{j, 1, Length[rPerDayAllSensitivity], {j, 1, maxSickPeriod}], {k, 1, Length[coverageRemoveScenarios]}];

For[k = 1, k <= Length[coverageRemoveScenarios], k++,
For[i = 1, i <= Length[rPerPersonAllSensitivity], i++,
For[p = 1, p <= Length[rPerPersonAllSensitivity[[i]]], p++,
   rPerPersonAllSensitivity[[i]][[p]] = PadRight[rPerPersonAllSensitivity[[i]][[p]], maxSickPeriod, 0];]
]; For[j = 1, j <= maxSickPeriod, j++,
   tempCopy = rPerPersonAllSensitivity[[i]]; For[p = 1, p <= Length[rPerPersonAllSensitivity[[i]]], p++,
   If[RandomReal] < coverageRemoveScenarios[[k]],
   tempCopy[[p]][[j ;; Length[rPerPersonAllSensitivity[[i]]][[p]]]] = 0;
   If[j == maxSickPeriod,
   tempCopy[[p]][[j]] = 0;
   ];
   ];
];

rUntilDayAll[[k, i, j]] = Total[Mean[tempCopy]]; ];
]
];

plotRemoveAll = GraphicsGrid[{Table[ListLinePlot[Table[Count[Flatten[rUntilDayAll[[k]], 2]][[j]], x_/;x<=1]/Length[Flatten[rUntilDayAll[[k]], 2]][[j]], {j, 1, maxSickPeriod}], PlotRange -> {0, 1.1}, AxesLabel -> "days", PlotLabel -> "Removing from all, coverage: ", coverageRemoveScenarios[[k]]], {k, 1, Length[coverageRemoveScenarios]}], ImageSize -> 1200];
}

Removing deceased from population
removeDead[] :=
{ rPerPersonDeadSensitivity = Table[Flatten[rPerDayDeadSensitivity[[i]], 2]], {i, 1, Length[rPerDayDeadSensitivity]}];

rPerPersonAliveSensitivity = Table[Flatten[rPerDayAliveSensitivity[[i]], 2]], {i, 1, Length[rPerDayAliveSensitivity]}];

rUntilDayDead = Table[{j, 1, Length[rPerDayDeadSensitivity], {j, 1, maxSickPeriod}], {k, 1, Length[coverageRemoveScenarios]}];

For[k = 1, k <= Length[coverageRemoveScenarios], k++,
For[i = 1, i <= Length[rPerPersonDeadSensitivity], i++,
For[p = 1, p <= Length[rPerPersonDeadSensitivity[[i]]], p++,
   rPerPersonDeadSensitivity[[i]][[p]] = PadRight[rPerPersonDeadSensitivity[[i]][[p]], maxSickPeriod, 0];]
]; For[p = 1, p <= Length[rPerPersonAliveSensitivity[[i]]], p++,
   rPerPersonAliveSensitivity[[i]][[p]] = PadRight[rPerPersonAliveSensitivity[[i]][[p]], maxSickPeriod, 0];
];

For[j = 1, j <= maxSickPeriod, j++,
   tempCopy = rPerPersonDeadSensitivity[[i]]; For[p = 1, p <= Length[rPerPersonDeadSensitivity[[i]]], p++,
   If[RandomReal] < coverageRemoveScenarios[[k]],
   tempCopy[[p]][[j ;; Length[rPerPersonAliveSensitivity[[i]]][[p]]]] = 0;
   If[j == maxSickPeriod,
   tempCopy[[p]][[j]] = 0;
   ];
   ];
   tempSum = Length[tempCopy] + Length[rPerPersonAliveSensitivity[[i]]];
   rUntilDayDead[[k, i, j]] = Total[Mean[tempCopy]]*Length[tempCopy] + Total[Mean[rPerPersonAliveSensitivity[[i]]]]*Length[rPerPersonAliveSensitivity[[i]]]/tempSum;
];
]

plotRemoveDead = GraphicsGrid[{Table[ListLinePlot[Table[Count[Flatten[rUntilDayDead[[k]], 2]][[j]], x_/;x<=1]/Length[Flatten[rUntilDayDead[[k]], 2]][[j]], {j, 1, maxSickPeriod}],
   ];
};
]

Downloaded From: http://annals.org/pdfaccess.ashx?url=/data/journals/aim/931895/ on 06/21/2017
PlotRange->[0,1.1],AxesLabel->{"days"},PlotLabel->{"Removing from deceased, coverage: ",coverageRemoveScenarios[[k]]},{k,1,Length[coverageRemoveScenarios]}],ImageSize->1200];}
(*removeDead*)

Reducing contacts (Iterations per scenario)
reduceContacts[iterUsed_]:= 
{ probEradicate[]=Count[Table[Mean[rAllSensitivity[[iii]]],{iii,1,Length[rAllSensitivity]]],x_/;x<=1]/Length[rAllSensitivity];

reduceContactsResults=Table[Table[0,{j,1,Length[contactScenarios]}],{i,1,Length[coverageContactsScenarios]}];

For[ii=1,ii<=Length[coverageContactsScenarios],ii++,
For[jj=1,jj<=Length[contactScenarios],jj++,
sensAnal[iterUsed,coverageContactsScenarios[[ii]],contactScenarios[[jj]]];
reduceContactsResults[[ii,jj]]=probEradicate[];
];
];

plotReduceContacts=Table[ListLinePlot[Transpose[{contactScenarios,reduceContactsResults[[ii]]}],PlotRange->[0,1.1],AxesOrigin->[0,0],ImageSize->Medium,AxesLabel->{"% contacts"},PlotLabel->{"Reducing contacts, coverage: ",coverageContactsScenarios[[ii]]},{ii,1,Length[coverageContactsScenarios]}]];(*reduceContacts*