

# **System Dynamics Modeling with R**

**System Dynamics Collaborative Learning Meeting**

**Worcester Polytechnic Institute (WPI)**

**May 24<sup>th</sup> 2017**

Jim Duggan,  
School of Engineering & Informatics  
National University of Ireland Galway.



NUI Galway  
OÉ Gaillimh

*System Dynamics Modeling with R*

<https://github.com/JimDuggan/SDMR>

# Overview

1. R and Data Science (Overview)
2. System Dynamics Modeling with R
  - Models using deSolve
  - Behaviour Modes
  - Disaggregate Models
  - Sensitivity Runs
  - Statistical Screening
  - Using RShiny



# (2.3) Disaggregate Models

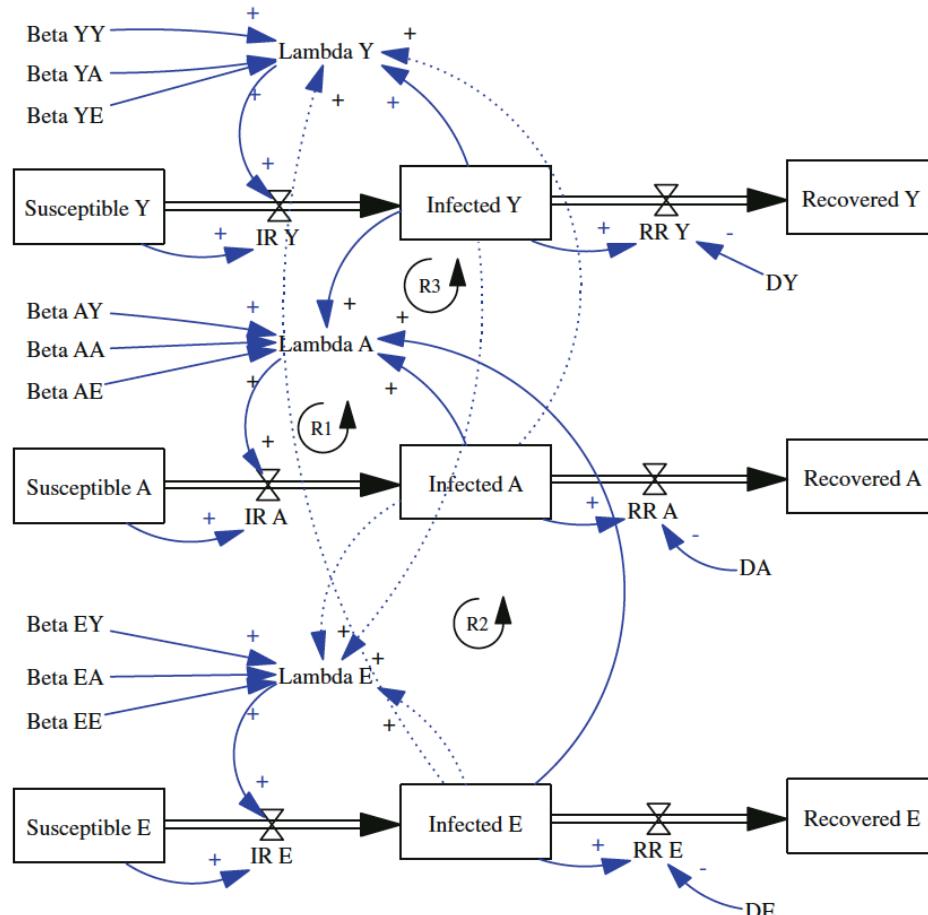


Fig. 5.5 A disaggregated SIR model (three cohorts)

```
model <- function(time, stocks, auxs){
  with(as.list(c(stocks, auxs)),{
    #convert the stocks vector to a matrix
    states<-matrix(stocks,nrow=NUM_COHORTS)

    Susceptible <- states[,1]
    Infected     <- states[,2]
    Recovered   <- states[,3]

    Lambda       <- beta %*% Infected
    IR           <- Lambda * Susceptible
    RR           <- Infected / delays
    dS_dt        <- -IR
    RR_Y         <- RR[1]
    RR_A         <- RR[2]
    RR_E         <- RR[3]
    DY           <- death_rate
    Lambda_Y    <- Lambda[1]
    Lambda_A    <- Lambda[2]
    Lambda_E    <- Lambda[3]
    IR_Y        <- IR[1]
    IR_A        <- IR[2]
    IR_E        <- IR[3]
    R1           <- delay[1]
    R2           <- delay[2]
    R3           <- delay[3]
    Beta_YY     <- beta[1,1]
    Beta_YA     <- beta[1,2]
    Beta_YE     <- beta[1,3]
    Beta_AY     <- beta[2,1]
    Beta_AA    <- beta[2,2]
    Beta_AE     <- beta[2,3]
    Beta_EY     <- beta[3,1]
    Beta_EA     <- beta[3,2]
    Beta_EE     <- beta[3,3]
  })
}
```



# Using matrices in SD Model

$$\begin{bmatrix} \lambda_1 \\ \vdots \\ \lambda_N \end{bmatrix} = \begin{bmatrix} ce_{11}/N_1 & \cdots & ce_{1n}/N_1 \\ \vdots & \ddots & \vdots \\ ce_{n1}/N_N & \cdots & ce_{nn}/N_N \end{bmatrix} \begin{bmatrix} I_1 \\ \vdots \\ I_N \end{bmatrix} \quad (5.64)$$

$$\begin{pmatrix} \lambda_Y \\ \lambda_A \\ \lambda_E \end{pmatrix} = \begin{pmatrix} CE_{YY}/N_Y & CE_{YA}/N_Y & CE_{YE}/N_Y \\ CE_{AY}/N_A & CE_{AA}/N_A & CE_{AE}/N_A \\ CE_{EY}/N_E & CE_{EA}/N_E & CE_{EE}/N_E \end{pmatrix} \begin{pmatrix} I_Y \\ I_A \\ I_E \end{pmatrix} \quad (5.65)$$



# R Code – Representing Stocks

```
#Setup the model variables
stocks <- c(SusceptibleY=24999, SusceptibleA=50000, SusceptibleE=25000,
             InfectedY=1,           InfectedA=0,           InfectedE=0,
             RecoveredY=0,          RecoveredA=0,          RecoveredE=0)

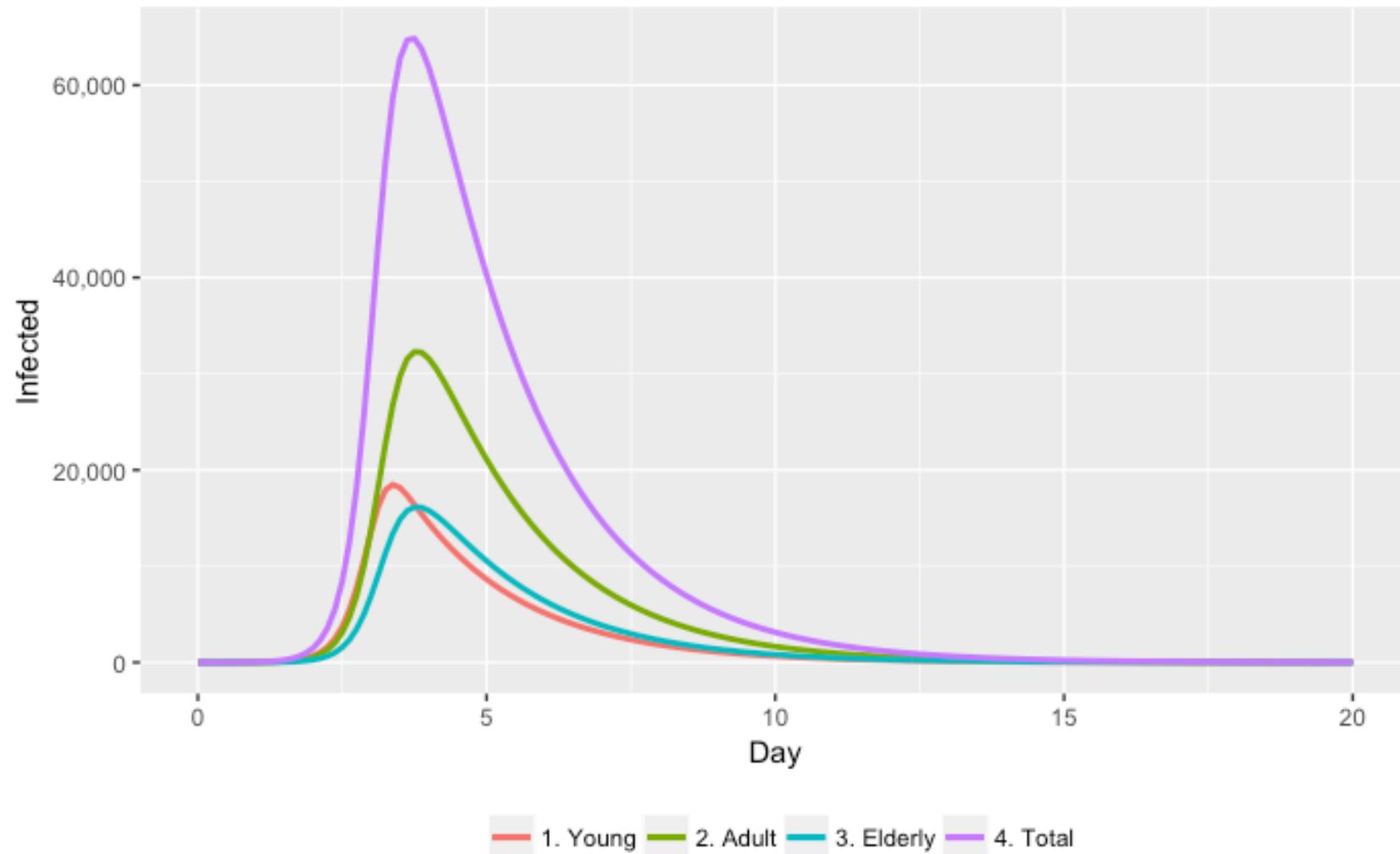
delays <- c(Young=2.0, Adult=2.0, Elderly=2.0)
```



```
29 model <- function(time, stocks, auxs){  
30   with(as.list(c(stocks, auxs)), {  
31     #convert the stocks vector to a matrix  
32     states<-matrix(stocks,nrow=NUM_COHORTS,ncol=NUM_STATES)  
33  
34     Susceptible <- states[,1]  
35     Infected    <- states[,2]  
36     Recovered   <- states[,3]  
37  
38     Lambda      <- beta %*% Infected  
39  
40     IR          <- Lambda * Susceptible  
41     RR          <- Infected / delays  
42  
43     dS_dt       <- -IR  
44     dI_dt       <- IR - RR  
45     dR_dt       <- RR  
46  
47     return (list(c(dS_dt, dI_dt, dR_dt)))
```



# Epi-Curve Output



# (2.4) Sensitivity Analysis (library FME)

- Vary key parameters
  - Effective Contacts
  - Recovery Delay
- Perform many simulation runs
- Analyse output

## Package ‘FME’

February 19, 2015

**Version** 1.3.2

**Title** A Flexible Modelling Environment for Inverse Modelling, Sensitivity, Identifiability, Monte Carlo Analysis.

**Author** Karline Soetaert <[karline.soetaert@nioz.nl](mailto:karline.soetaert@nioz.nl)>, Thomas Petzoldt <[thomas.petzoldt@tu-dresden.de](mailto:thomas.petzoldt@tu-dresden.de)>

**Maintainer** Karline Soetaert <[karline.soetaert@nioz.nl](mailto:karline.soetaert@nioz.nl)>

**Depends** R (>= 2.6), deSolve, rootSolve, coda

**Imports** minpack.lm, MASS

**Suggests** diagram

**Description** Provides functions to help in fitting models to data, to perform Monte Carlo, sensitivity and identifiability analysis. It is intended to work with models be written as a set of differential equations that are solved either by an integration routine from package deSolve, or a steady-state solver from package rootSolve. However, the methods can also be used with other types of functions.



# Overall Idea

R Source code

## Run Simulation Function (params)

### Model function

Net flow equations go here

Call Model Function

Return Results

Setup Params (FME library)

Call Simulation Function

Display Results



# Generating Parameters

```
CE.MIN<-0;           CE.MAX<-7.0
DEL.MIN<-1.0;         DEL.MAX<-10.0
INIT.INF.MIN<-1.0;   INIT.INF.MAX<-25.0;

parRange<-data.frame(
  min=c(CE.MIN, DEL.MIN, INIT.INF.MIN),
  max=c(CE.MAX, DEL.MAX, INIT.INF.MAX)
)

rownames(parRange)<-c("aEffective.Contact.Rate", "aDelay", "initInfected")

NRUNS<-10
p<-data.frame(RunNo=1:NRUNS, Latinhyper(parRange, NRUNS))
```



# parRange data frame & LatinHyper()

```
> parRange
```

		min	max
aEffective.Contact.Rate		0	7
aDelay		1	10
initInfected		1	25

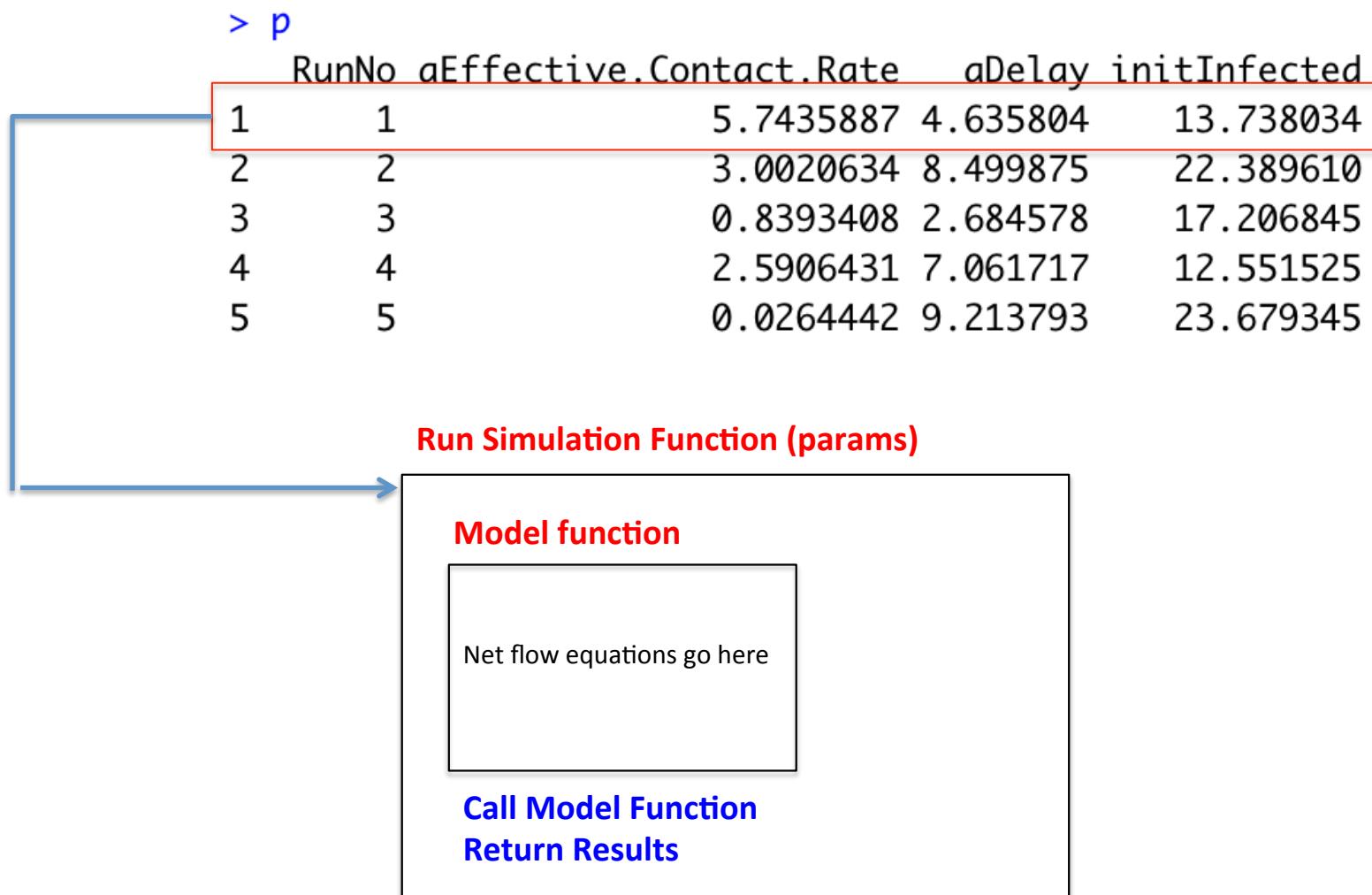
```
>
```

```
> p
```

RunNo	aEffective.Contact.Rate	aDelay	initInfected
1	5.7435887	4.635804	13.738034
2	3.0020634	8.499875	22.389610
3	0.8393408	2.684578	17.206845
4	2.5906431	7.061717	12.551525
5	0.0264442	9.213793	23.679345
6	4.2287664	8.193284	8.757535
7	6.8971697	4.544439	5.548590
8	5.5252311	5.864802	2.606257
9	1.8602947	3.583974	7.451173
10	3.8608469	1.207800	17.882494



# Overall idea... $N$ simulation runs



# The Code (1/3)

```
runsim <- function(rvec){  
  # this is the model function  
  model <- function(time, stocks, auxs){  
    with(as.list(c(stocks, auxs)), {  
      aBeta <- aEffective.Contact.Rate / aTotalPopulation  
      aLambda <- aBeta * sInfected  
  
      fIR <- sSusceptible * aLambda  
      fRR <- sInfected / aDelay  
  
      dS_dt <- -fIR  
      dI_dt <- fIR - fRR  
      dR_dt <- fRR  
  
      return (list(c(dS_dt,dI_dt,dR_dt),  
                  IR=fIR, RR=fRR,Beta=aBeta,Lambda=aLambda,DEL=aDelay,  
                  CE=aEffective.Contact.Rate,InitI=initInfected))  
    })  
  }  
}
```



# The Code (2/3)

```
# setup the individual simulation run
START<-0; FINISH<-20; STEP<-0.01;
simtime <- seq(START, FINISH, by=STEP)

init<-rvec[["initInfected"]]

a<-c(aTotalPopulation=10000,rvec["aEffective.Contact.Rate"],
      rvec["aDelay"],rvec["initInfected"])

stocks  <- c(sSusceptible=10000-init,sInfected=init,sRecovered=0)

o<-data.frame(ode(y=stocks, simtime, func = model,
                     parms=a, method="euler"))
o$RunNumber<-rvec["RunNo"]
o
}
```



# The Code (3/3)

```
CE.MIN<-0;           CE.MAX<-7.0
DEL.MIN<-1.0;         DEL.MAX<-10.0
INIT.INF.MIN<-1.0;   INIT.INF.MAX<-25.0

parRange<-data.frame(
  min=c(CE.MIN, DEL.MIN, INIT.INF.MIN),
  max=c(CE.MAX, DEL.MAX, INIT.INF.MAX)
)

rownames(parRange)<-c("aEffective.Contact.Rate", "aDelay", "initInfected")

NRUNS<-10
p<-data.frame(RunNo=1:NRUNS, Latinhyper(parRange, NRUNS))

out<-apply(p, 1, function(x)runsim(x))

df<-rbind.fill(out)
```



# Initial output is a list of data frames: One for each simulation

```
> str(out[[1]])  
'data.frame': 5001 obs. of 12 variables:  
 $ time      : num  0 0.01 0.02 0.03 0.04 0.05 0.06 0.07 0.08 0.09 ...  
 $ sSusceptible: num  9991 9991 9991 9990 9990 ...  
 $ sInfected   : num  8.65 8.99 9.34 9.71 10.09 ...  
 $ sRecovered   : num  0 0.0103 0.021 0.0321 0.0436 ...  
 $ IR          : num  35.1 36.4 37.9 39.4 40.9 ...  
 $ RR          : num  1.03 1.07 1.11 1.15 1.2 ...  
 $ Beta         : num  0.000406 0.000406 0.000406 0.000406 0.000406 ...  
 $ Lambda       : num  0.00351 0.00365 0.00379 0.00394 0.00409 ...  
 $ DEL          : num  8.41 8.41 8.41 8.41 8.41 ...  
 $ CE           : num  4.06 4.06 4.06 4.06 4.06 ...  
 $ InitI        : num  8.65 8.65 8.65 8.65 8.65 ...  
 $ RunNumber    : num  1 1 1 1 1 1 1 1 1 1 ...
```

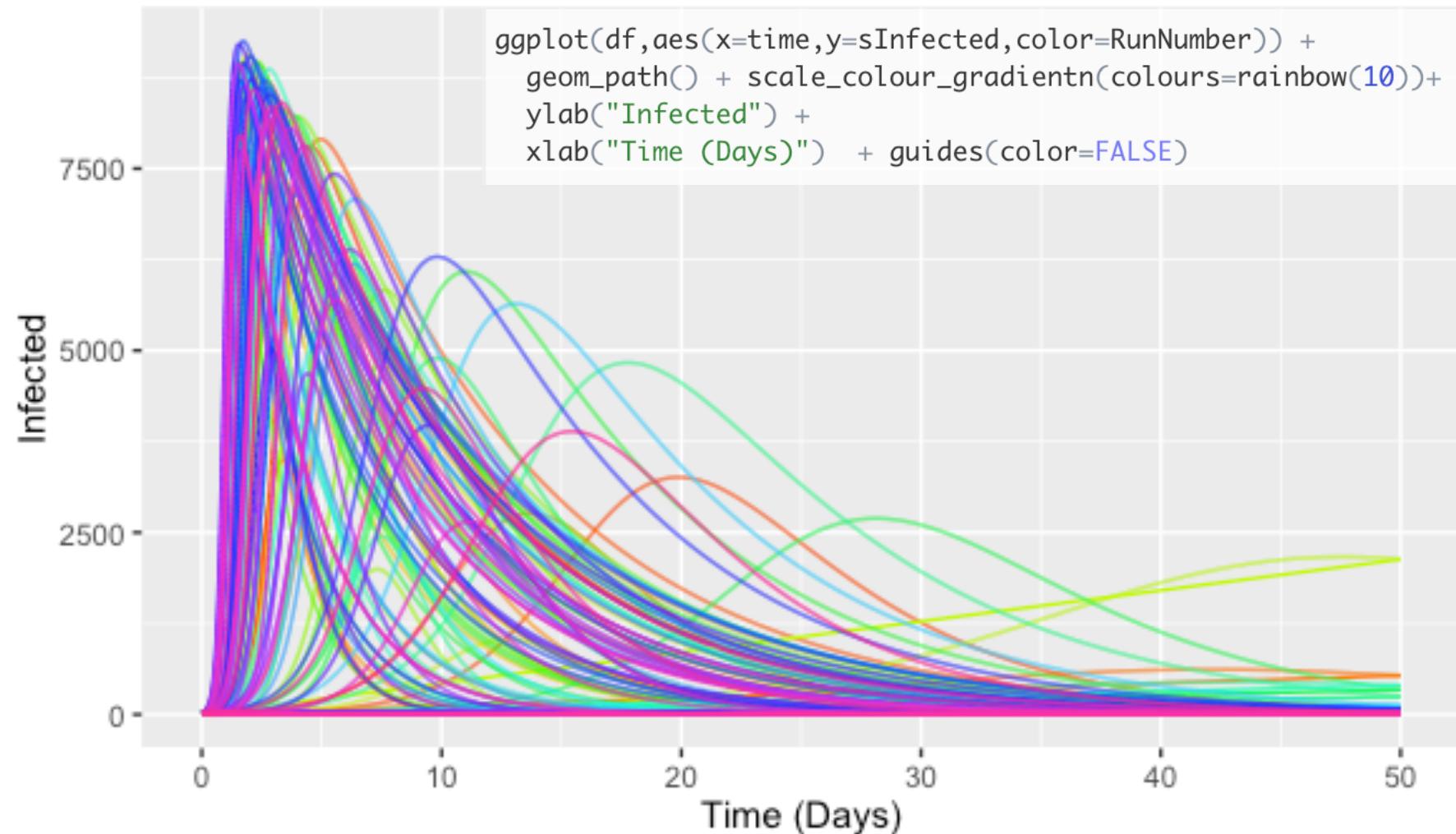


# Converted to a single data frame

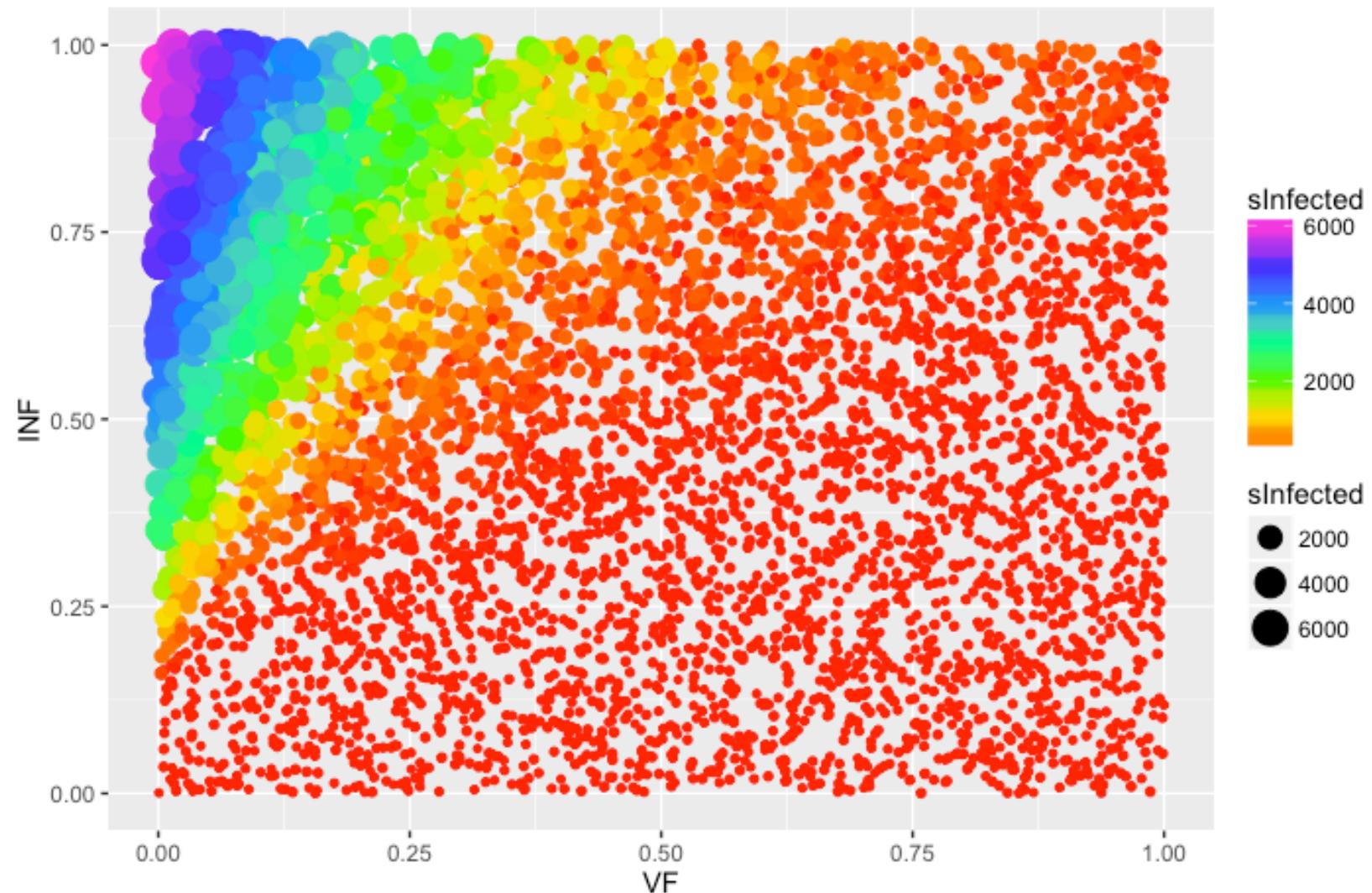
```
> df<-rbind.fill(out)
>
>
> str(df)
'data.frame': 500100 obs. of 12 variables:
 $ time       : num  0 0.01 0.02 0.03 0.04 0.05 0.06 0.07 0.08 0.09 ...
 $ sSusceptible: num 9991 9991 9991 9990 9990 ...
 $ sInfected   : num 8.65 8.99 9.34 9.71 10.09 ...
 $ sRecovered   : num 0 0.0103 0.021 0.0321 0.0436 ...
 $ IR          : num 35.1 36.4 37.9 39.4 40.9 ...
 $ RR          : num 1.03 1.07 1.11 1.15 1.2 ...
 $ Beta         : num 0.000406 0.000406 0.000406 0.000406 0.000406 ...
 $ Lambda       : num 0.00351 0.00365 0.00379 0.00394 0.00409 ...
 $ DEL          : num 8.41 8.41 8.41 8.41 8.41 ...
 $ CE           : num 4.06 4.06 4.06 4.06 4.06 ...
 $ InitI         : num 8.65 8.65 8.65 8.65 8.65 ...
 $ RunNumber    : num 1 1 1 1 1 1 1 1 1 1 ...
```



# Visualise Simulations



# $N = 5000$



## (2.5) Statistical Screening

*The system dynamics approach leads to models with a large number of highly uncertain parameters, so we should ask ourselves which of the parameters are really important.* Ford and Flynn (2005)

Statistical screening utilizes the sensitivity output data to calculate the *correlation coefficients* between parameters and a user-defined system performance variable (Taylor, Ford and Ford 2010).

$$r = \frac{\sum (X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{\sum (X_i - \bar{X})^2 \sum (Y_i - \bar{Y})^2}}$$



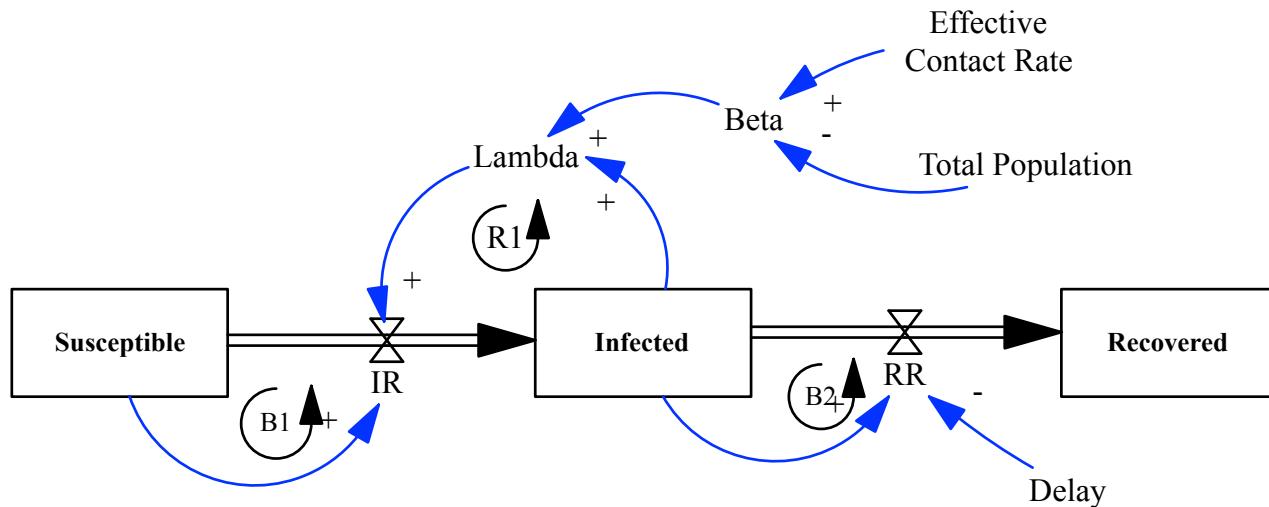
# Steps (based on Taylor et al. 2010)

1. Select a set of exogenous model parameters, and a system performance variable for analysis. Select appropriate parameter ranges.
2. Calculate the correlation coefficients between the selected exogenous model parameters and the system performance variables. Plot the correlation coefficients and the behavior of the performance variable over time.
3. Select the time interval for analysis, by examining the time series data of both the performance variable, and the correlation coefficients.
4. Generate a list of high-leverage parameters, which are those that recorded the highest absolute correlation coefficient values during the selected time period.
5. Based on the parameters selected from step 4, identify the high-leverage model structure(s) that are directly influenced by the parameters..
6. Develop explanations about how each parameter (or set of parameters), and the model structures they influence, drive the overall system behavior.



# Model - SIR

Parameter	Description	Min	Max
Infected <sub>INIT</sub>	The initial value of number infected in the model. A number greater than zero is required in order for the disease to spread.	1.0	25.0
$C_E$	Effective contact rate, where higher values increase the spread of a disease.	0	7.0
$D$	Recovery delay, where a longer delay will result in people spending longer times in the infected stock.	1.0	10.0



# N=200, 40200 Observations

```
> round(head(df),2)
```

	time	sSusceptible	sInfected	sRecovered	IR	RR	Beta	Lambda	DEL	CE	InitI	run
1	0.00	9995.99	4.01	0.00	18.63	0.96	0	0.00	4.16	4.65	4.01	1
2	0.12	9993.66	6.22	0.12	28.87	1.50	0	0.00	4.16	4.65	4.01	1
3	0.25	9990.05	9.64	0.31	44.74	2.32	0	0.00	4.16	4.65	4.01	1
4	0.38	9984.46	14.94	0.60	69.31	3.59	0	0.01	4.16	4.65	4.01	1
5	0.50	9975.80	23.16	1.05	107.32	5.57	0	0.01	4.16	4.65	4.01	1
6	0.62	9962.38	35.88	1.74	166.03	8.63	0	0.02	4.16	4.65	4.01	1

```
> round(tail(df),2)
```

	time	sSusceptible	sInfected	sRecovered	IR	RR	Beta	Lambda	DEL	CE	InitI	run
40195	24.38	0	139.65	9860.35	0	25.94	0	0.08	5.38	5.93	6.7	200
40196	24.50	0	136.41	9863.59	0	25.34	0	0.08	5.38	5.93	6.7	200
40197	24.62	0	133.25	9866.75	0	24.75	0	0.08	5.38	5.93	6.7	200
40198	24.75	0	130.15	9869.85	0	24.17	0	0.08	5.38	5.93	6.7	200
40199	24.88	0	127.13	9872.87	0	23.61	0	0.08	5.38	5.93	6.7	200
40200	25.00	0	124.18	9875.82	0	23.06	0	0.07	5.38	5.93	6.7	200

*Challenge: Need to group the data by time step for correlation analysis*



# split(x,f)

## Arguments

- x vector or data frame containing values to be divided into groups.
- f a ‘factor’ in the sense that `as.factor(f)` defines the grouping, or a list of such factors in which case their interaction is used for the grouping.



# Split the data set

```
> runs<-split(df,df$time)
> round(head(runs[[1]]),2)
  time sSusceptible sInfected sRecovered      IR      RR Beta Lambda DEL CE InitI run
1     0    9995.99     4.01          0 18.63 0.96    0  0.00 4.16 4.65 4.01 1
202   0    9997.29     2.71          0 15.17 0.38    0  0.00 7.20 5.61 2.71 2
403   0    9981.98    18.02          0 102.64 2.76    0  0.01 6.54 5.71 18.02 3
604   0    9977.28    22.72          0 96.75 5.68    0  0.01 4.00 4.27 22.72 4
805   0    9995.46     4.54          0  3.88 1.51    0  0.00 3.00 0.86 4.54 5
1006  0    9989.73    10.27          0 57.92 7.96    0  0.01 1.29 5.65 10.27 6
> round(head(runs[[2]]),2)
  time sSusceptible sInfected sRecovered      IR      RR Beta Lambda DEL CE InitI run
2     0.12    9993.66     6.22     0.12 28.87 1.50    0  0.00 4.16 4.65 4.01 1
203   0.12    9995.40     4.56     0.05 25.54 0.63    0  0.00 7.20 5.61 2.71 2
404   0.12    9969.15    30.50     0.34 173.55 4.66    0  0.02 6.54 5.71 18.02 3
605   0.12    9965.19    34.10     0.71 145.05 8.53    0  0.01 4.00 4.27 22.72 4
806   0.12    9994.97     4.84     0.19  4.13 1.61    0  0.00 3.00 0.86 4.54 5
1007  0.12    9982.49    16.51     1.00 93.08 12.80    0  0.01 1.29 5.65 10.27 6
```

*Challenge: How to calculate the correlation coefficient for each time step?*



# sapply(x,f)

- Another use of user-defined functions in R is as parameters to the *apply* family of functions.
- The general form of the **sapply(x,f,fargs)** function is as follows:
  - **x** is the target vector or list
  - **f** is the function to be called and applied to each element
  - **fargs** are the optional set of arguments that can be applied to the function f.
  - **sapply()** returns a vector



# Get the average at each time

*using an anonymous function*

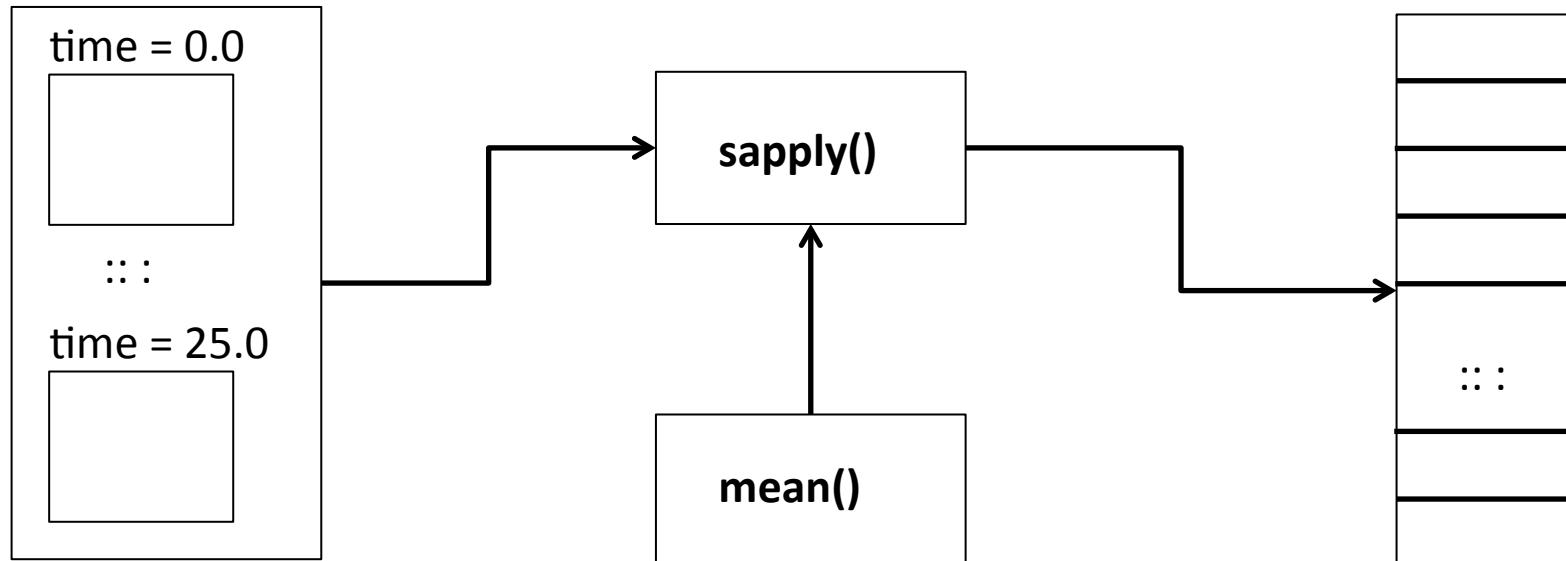
```
av.Infected<-sapply(runs,function(l){mean(l$Infected)})
```

```
> round(av.Infected[1:10],3)
```

0	0.125	0.25	0.375	0.5	0.625	0.75	0.875	1	1.125
12.997	18.196	26.327	39.230	59.956	93.535	148.129	236.543	377.490	594.582

**INPUT**  
(List of Data Frames)

**OUTPUT**  
(Vector of means)



# Calculating cor()

```

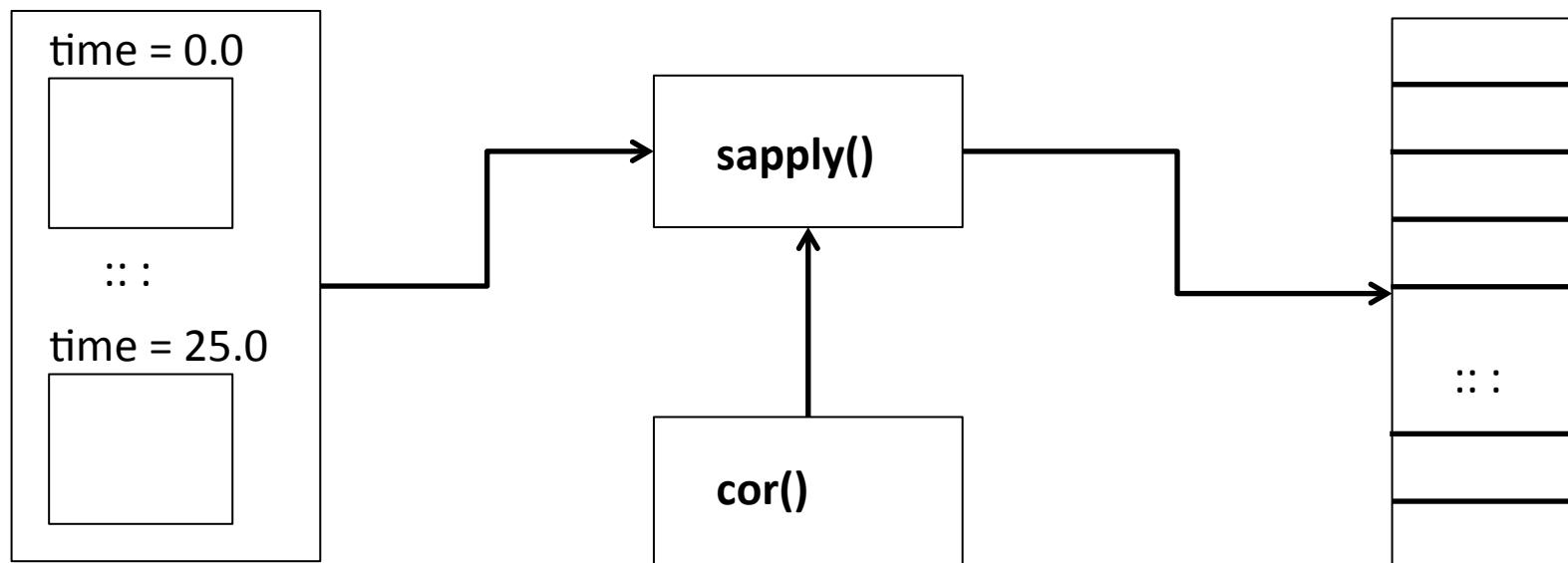
> cor.CE<-sapply(runs,function(l){cor(l$sInfected, l$CE)})
> round(cor.CE[1:10],3)
      0 0.125 0.25 0.375    0.5 0.625 0.75 0.875      1 1.125
0.069 0.388 0.583 0.683 0.730 0.750 0.755 0.755 0.753 0.755

```

# **INPUT**

(List of Data Frames)

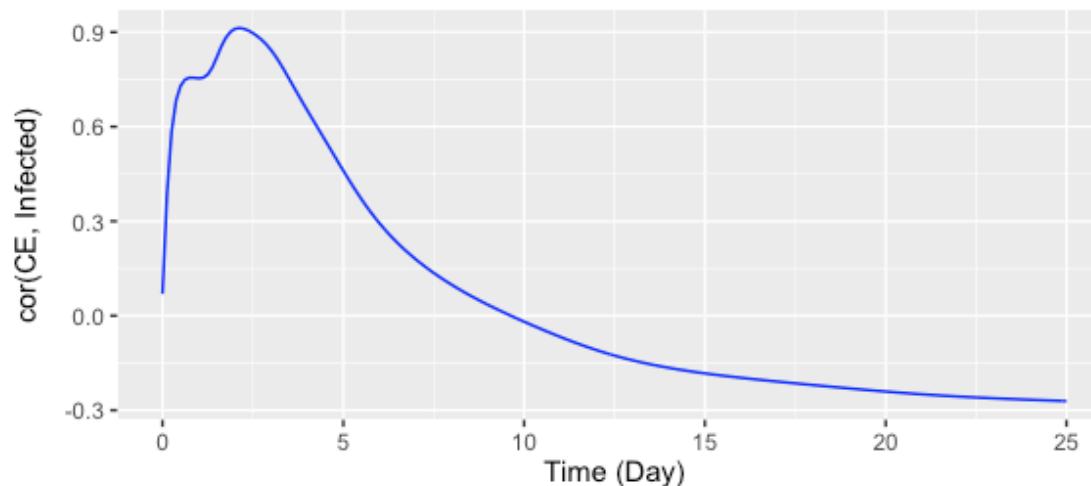
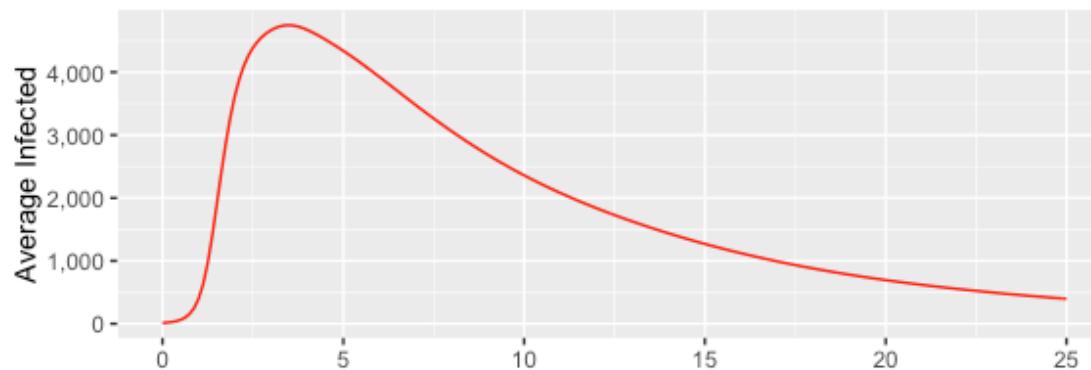
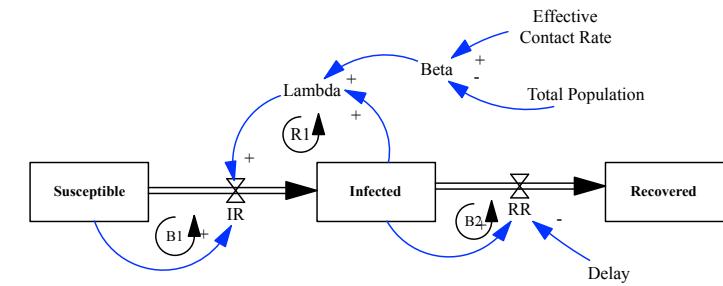
## **OUTPUT**

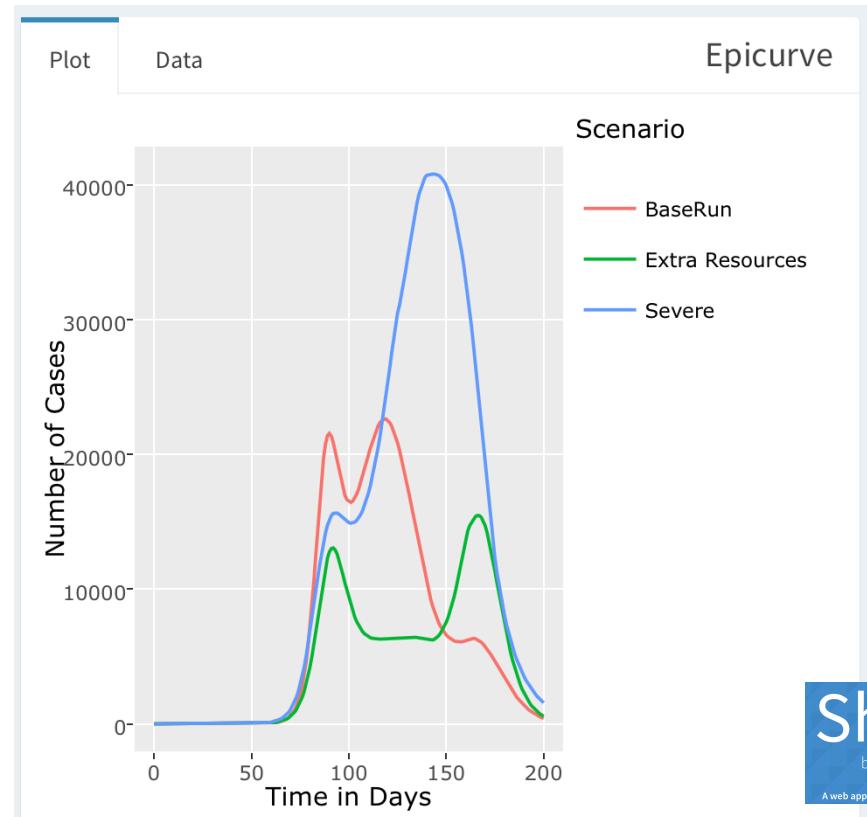
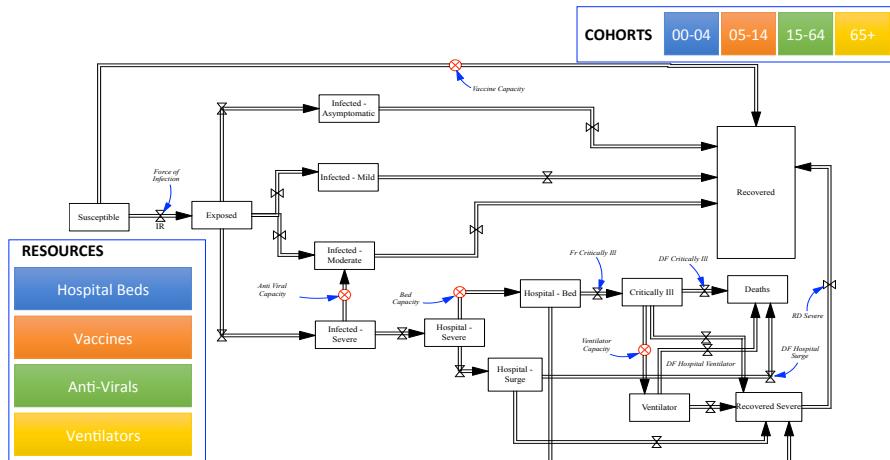


# Visualising Output

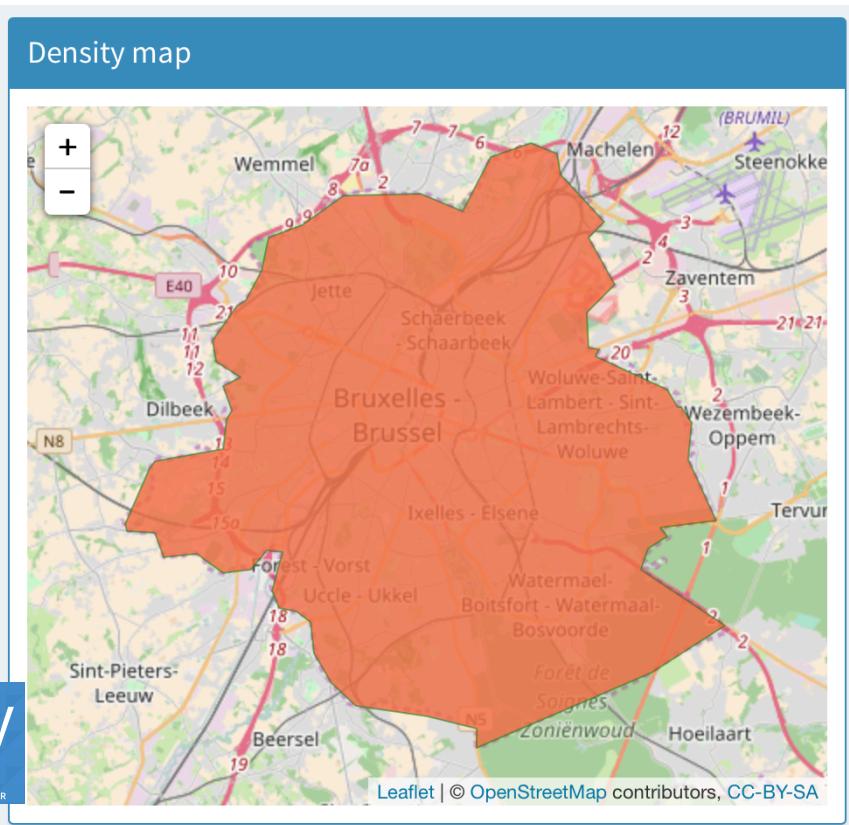
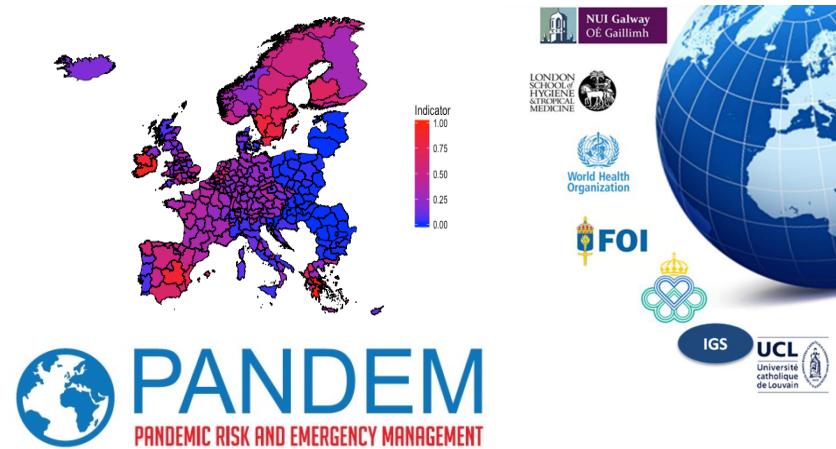
```
> summary(cor.CE)
```

	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
	-0.27090	-0.22810	-0.12640	0.06012	0.24550	0.91310





**Shiny**  
by RStudio  
A web application framework for R



NUI Galway  
OÉ Gaillimh

*System Dynamics Modeling with R*

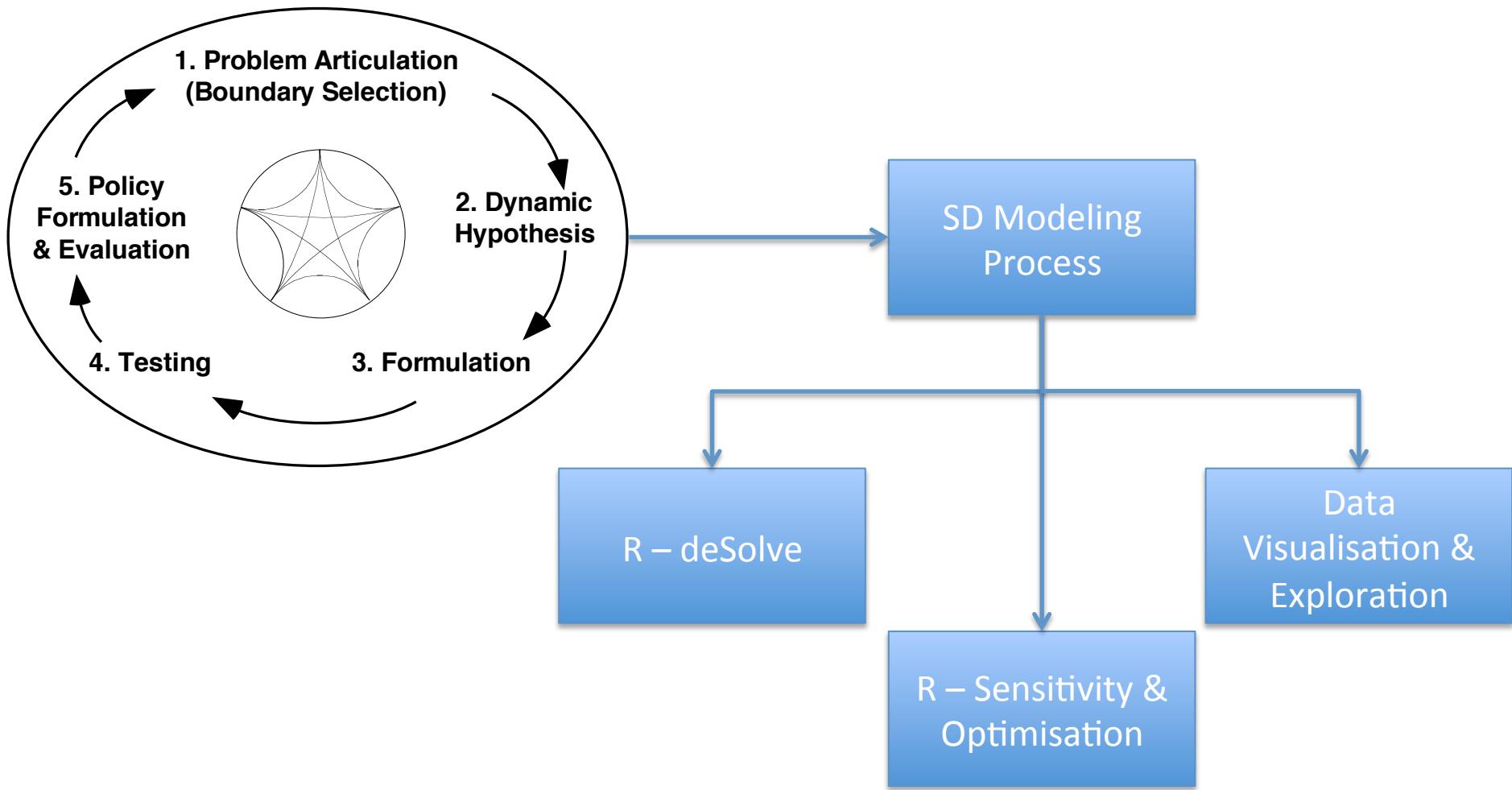
<https://github.com/JimDuggan/SDMR>

# Parameters

Parameters				
WAIFW Matrix (4x4)	VaccinationDay	AVAmount	Ventilators	SevereRecoveryDelay
Country	VaccinationAmount	AVSevModEff	HospitalBedsTxMod	VentilatorRecoveryDelay
GeoLabel	VaccinationDelay	CloseSchools	AverageRecoveryDelay	WorkforPercent
Region	Population	ReportingFraction	Vaccinate_0004	EconomicValuePerYear
SimulationDate	Population_0004	CloseThreshold	Vaccinate_0514	CostPerUnitVaccine
InjectionDay	Population_0514	CloseDuration	Vaccinate_1564	CostPerUnitAntiViral
InjectionAmount	Population_1564	ResAVMultiplier	Vaccinate_65P	CostPerUnitVentilator
InjectionIndex	Population_65P	ResVAccMultiplier	FRCriticallyIll	
PropAsm	Pr_Immune_0004	ResBedsMultiplier	DFHospitalSurge	
PropMild	Pr_Immune_0514	InfectivityMultiplier	DFHospitalVentilator	
PropMod	Pr_Immune_1564	SeasonalForcing	DFCriticallyIll	
PropSev	Pr_Immune_65P	HospitalBeds		



# Summary



Sterman (2000). Business Dynamics. McGraw-Hill.



# Further Reading/Resources

JimDuggan / SDMR

Code Issues 0 Pull requests 0 Projects 0 Wiki Pulse Graphs Settings

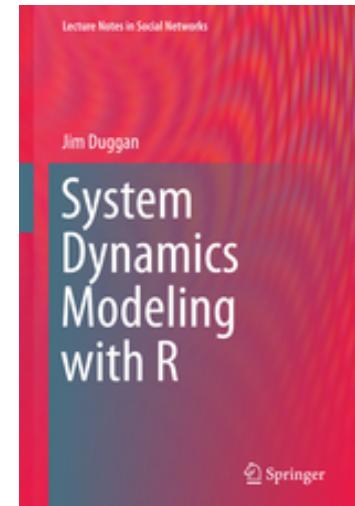
Resources for text book "System Dynamics Modeling with R"

Add topics

101 commits 1 branch 0 releases 1 contributor MIT

Branch: master ▾ New pull request Create new file Upload files Find file Clone or download ▾

Commit	Message	Date
	JimDuggan Update of files...	Latest commit d0b4a00 2 days ago
	R programming/examples Update from lecture...	6 months ago
	RShiny Adding RShiny Tutorial (R Studio Content)	2 months ago
	archive 2015 Adding lecturing archive from 2015	9 months ago
	images First draft of README summary	11 months ago
	lectures/CT561 Update for energy analysis	2 months ago
	models Update for energy analysis	2 months ago
	reader Adding a pipeline delay example	4 months ago
	workshops Update of files...	2 days ago



<http://link.springer.com/book/10.1007%2F978-3-319-34043-2>