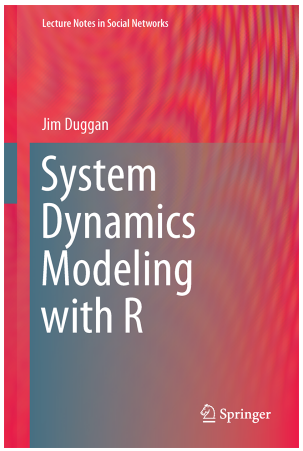
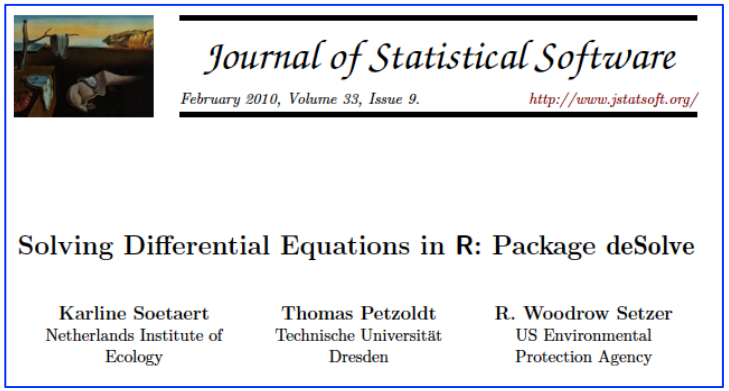
Session 2:
System Dynamics with R

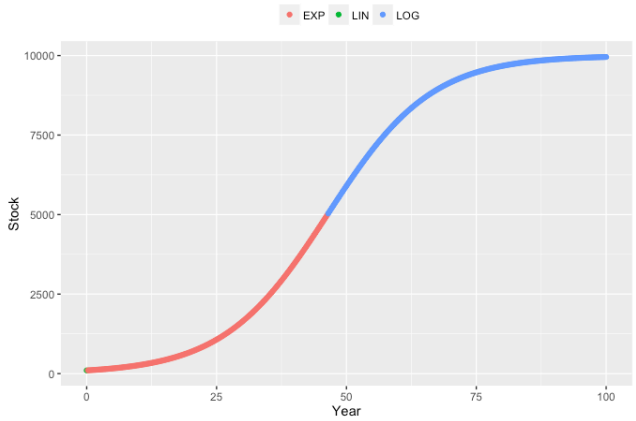


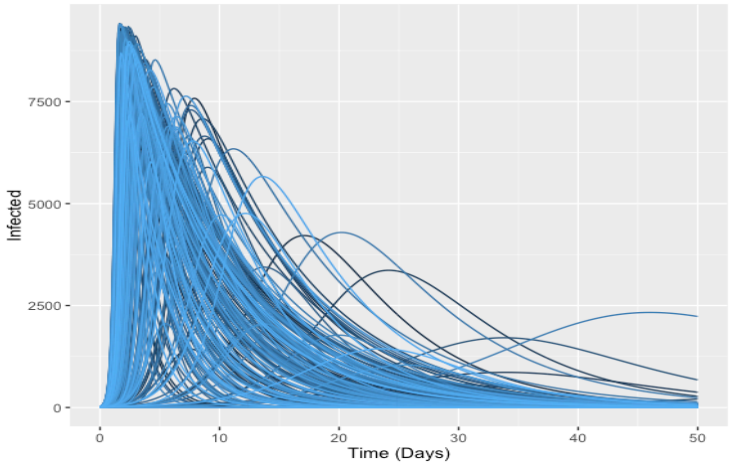
Dr. Jim Duggan,
School of Engineering & Informatics
National University of Ireland Galway.

Overview

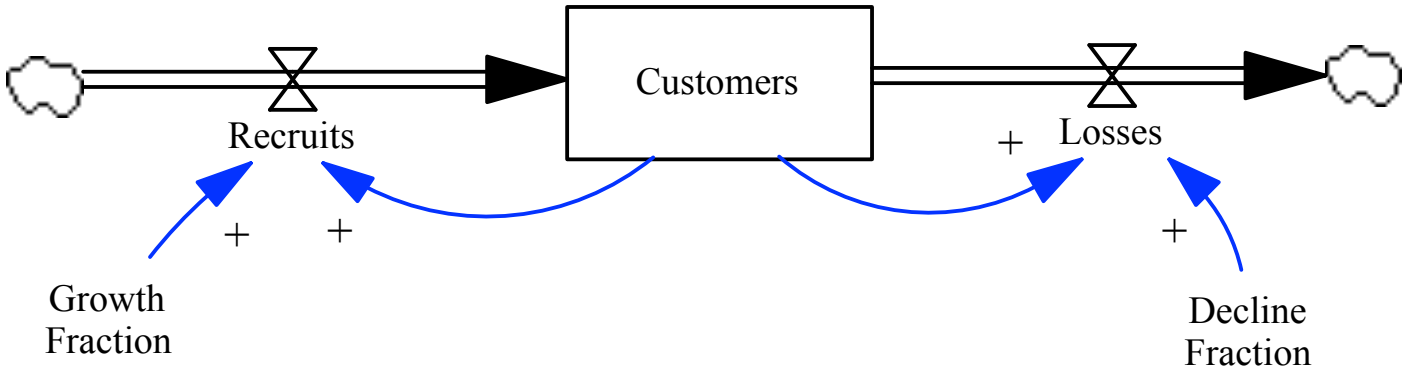
- Introduction to deSolve
- S Shaped Growth & Behaviour Modes
- Sensitivity Analysis
- Statistical Screening







(1) Introduction to deSolve



$$Customers = INTEGRAL(Recruits - Losses, 10000)$$

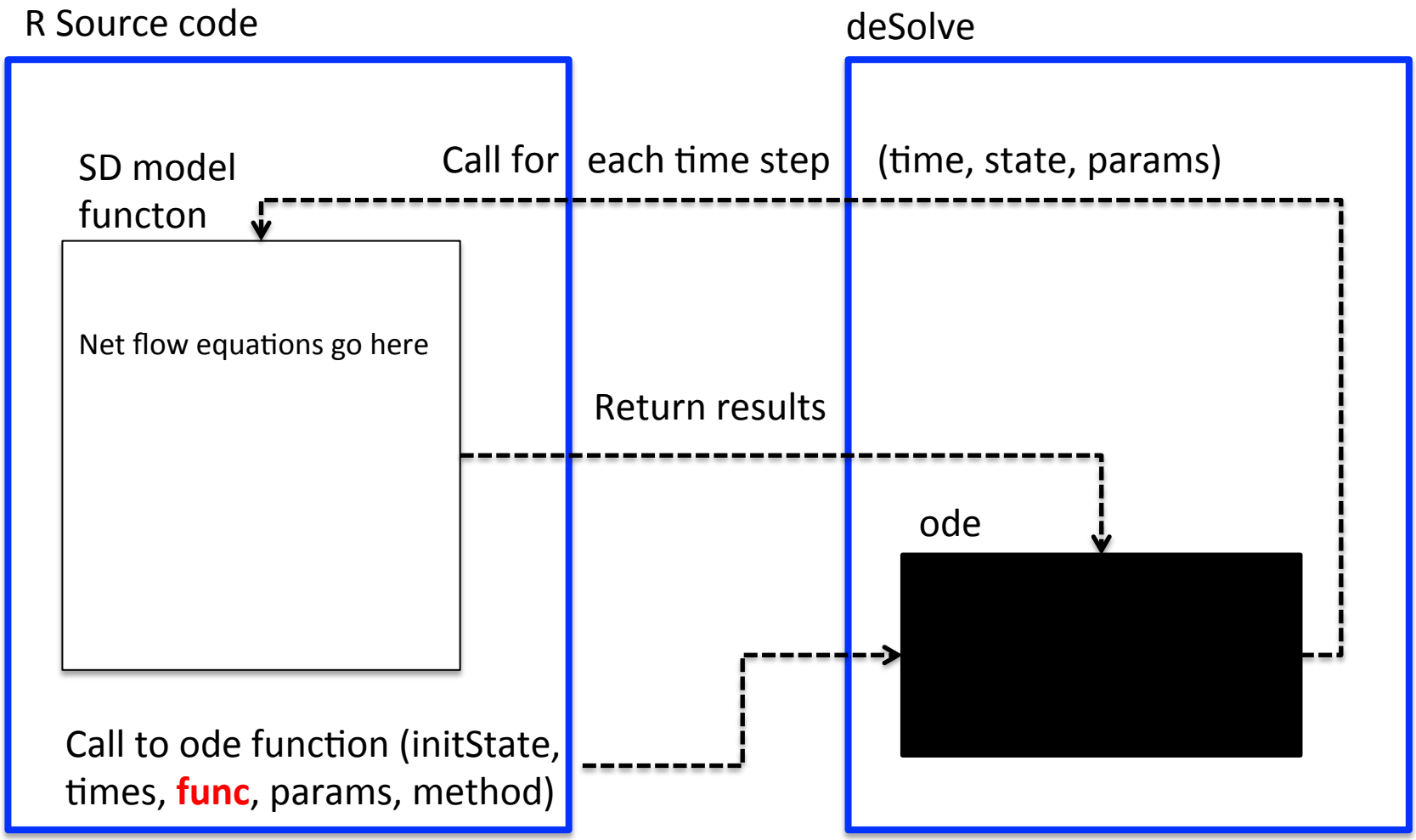
$$Recruits = Customers \times Growth\ Fraction$$

$$Losses = Customers \times Decline\ Fraction$$

$$Growth\ Fraction = 0.07$$

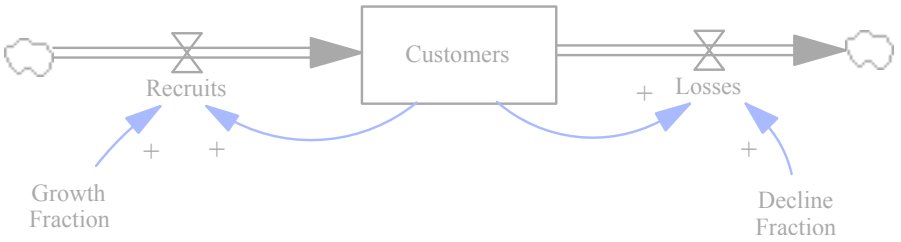
$$Decline\ Fraction = 0.03$$

deSolve Approach (callbacks)



R Code (1/3)

```
library(deSolve)
library(ggplot2)
require(gridExtra)
library(scales)
```




```
# Setup simulation times and time step
START<-2015; FINISH<-2030; STEP<-0.25

# Create time vector
simtime <- seq(START, FINISH, by=STEP)

# Create stock and auxs
stocks  <- c(sCustomers=10000)
auxs    <- c(aGrowthFraction=0.08, aDeclineFraction=0.03)
```

R Code (2/3)

```
# Model function
model <- function(time, stocks, auxs){
  with(as.list(c(stocks, auxs)),{

    fRecruits<-sCustomers*aGrowthFraction

    fLosses<-sCustomers*aDeclineFraction

    dC_dt <- fRecruits - fLosses

    return (list(c(dC_dt),
                 Recruits=fRecruits, Losses=fLosses,
                 GF=aGrowthFraction,DF=aDeclineFraction))
  })
}
```

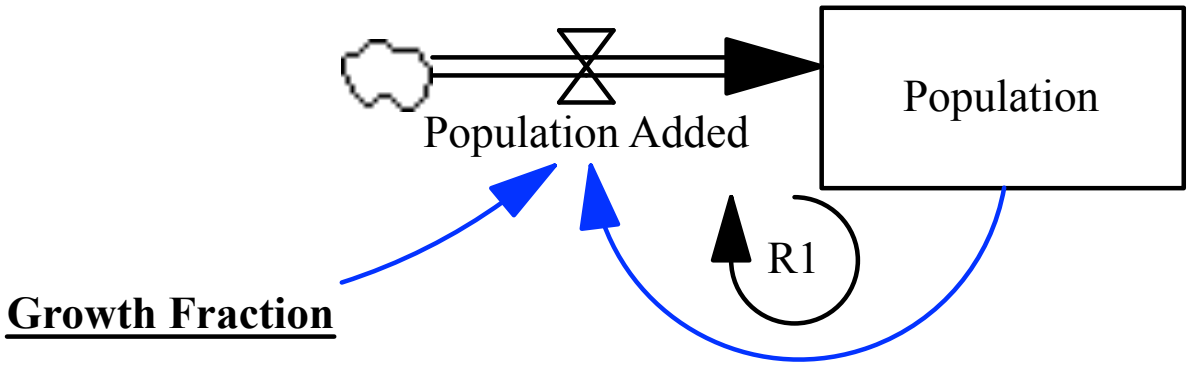
R Code (3/3)

```
# Run simulation
o<-data.frame(ode(y=stocks, times=simtime, func = model,
                  parms=auxs, method="euler"))
```

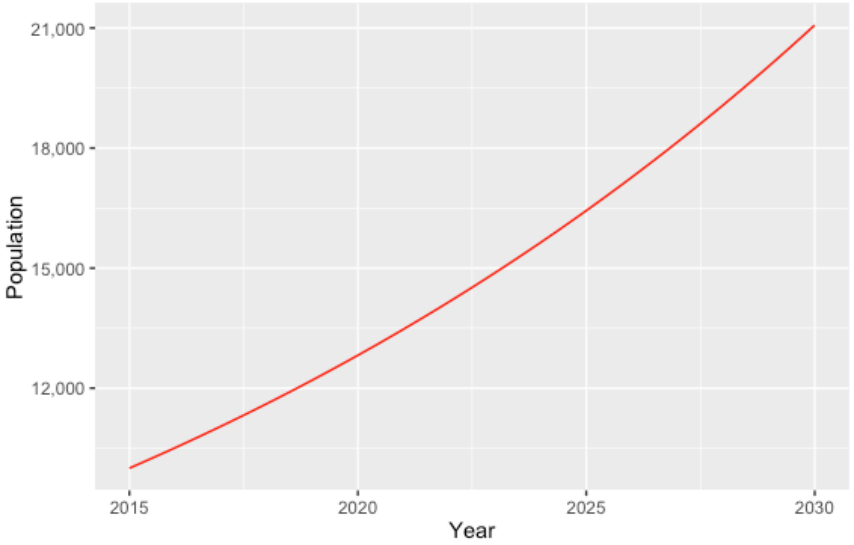
```
> o[1:10,]
    time      sCustomers Recruits  Losses     NetFlow   GF   DF
1  2015.00   10000.00    800.0000 300.0000   500.0000  0.08 0.03
2  2015.25   10125.00    810.0000 303.7500   506.2500  0.08 0.03
3  2015.50   10251.56    820.1250 307.5469   512.5781  0.08 0.03
4  2015.75   10379.71    830.3766 311.3912   518.9854  0.08 0.03
5  2016.00   10509.45    840.7563 315.2836   525.4727  0.08 0.03
6  2016.25   10640.82    851.2657 319.2246   532.0411  0.08 0.03
7  2016.50   10773.83    861.9065 323.2150   538.6916  0.08 0.03
8  2016.75   10908.50    872.6804 327.2551   545.4252  0.08 0.03
9  2017.00   11044.86    883.5889 331.3458   552.2431  0.08 0.03
10 2017.25   11182.92    894.6337 335.4877   559.1461  0.08 0.03
```

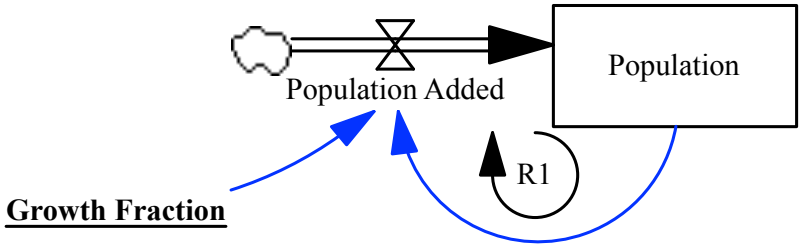
Challenge 1

- Implement the following model in R. Assume an initial population of 100000 and a growth fraction of 0.05. Plot the results, and a phase plot.

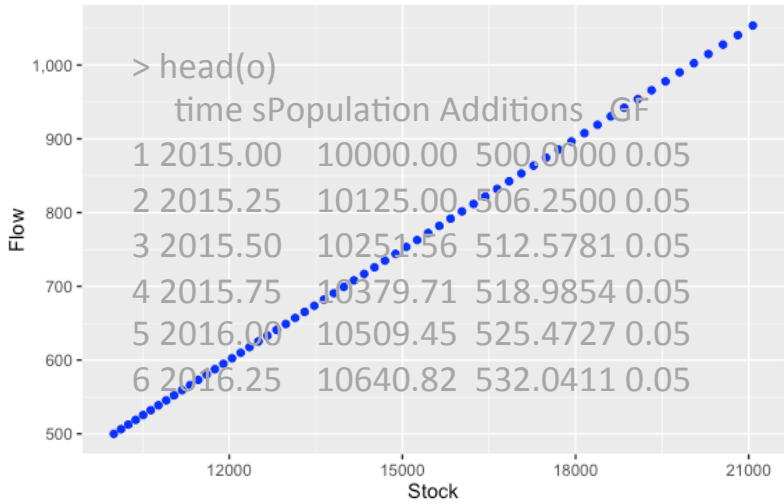


Results



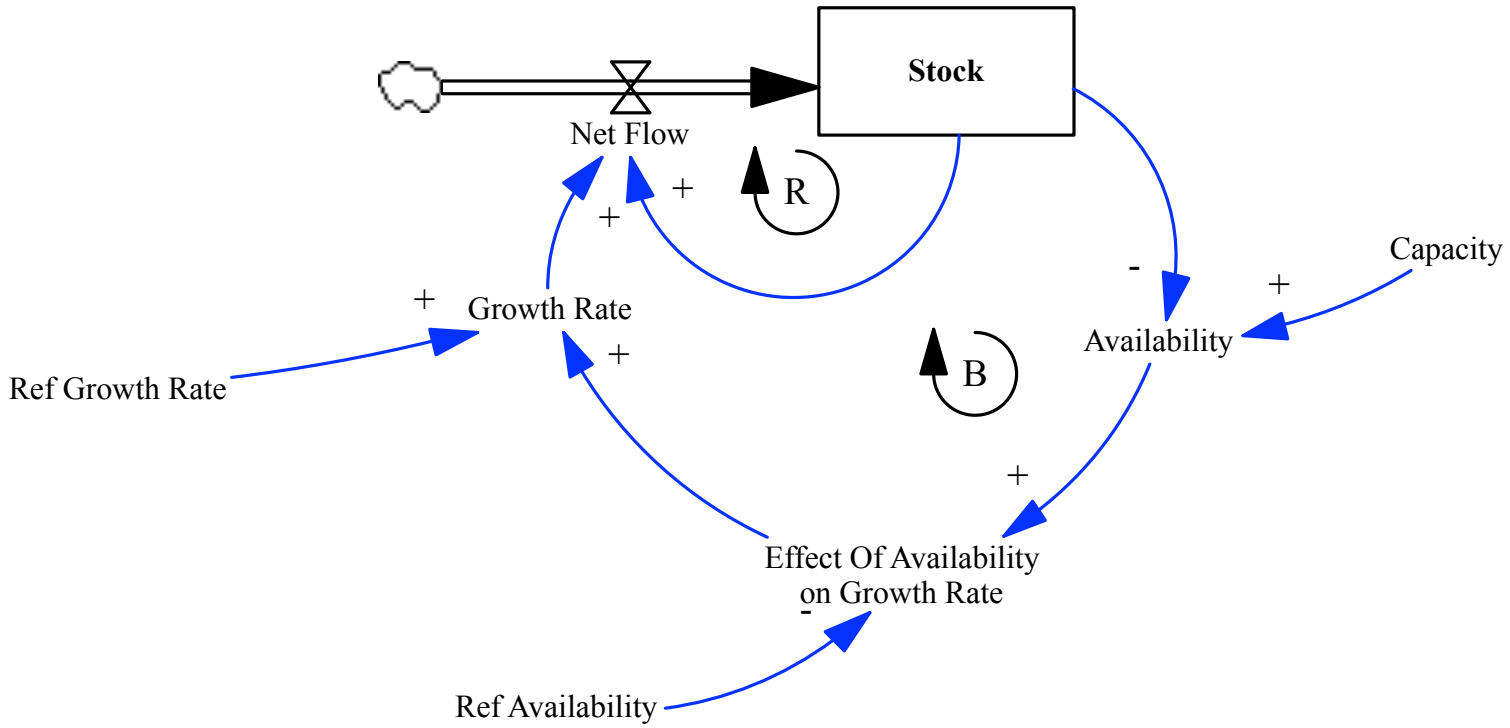


Population= INTEG (Population Added,10000)
Growth Fraction=0.05
Population Added=Population*Growth Fraction



(2) S-Shaped Growth & Behaviour Modes

There will always be limits to growth. They can be self-imposed. If they aren't, they will be system-imposed.
Donella H. Meadows, Thinking in Systems: A Primer (2008), p.103

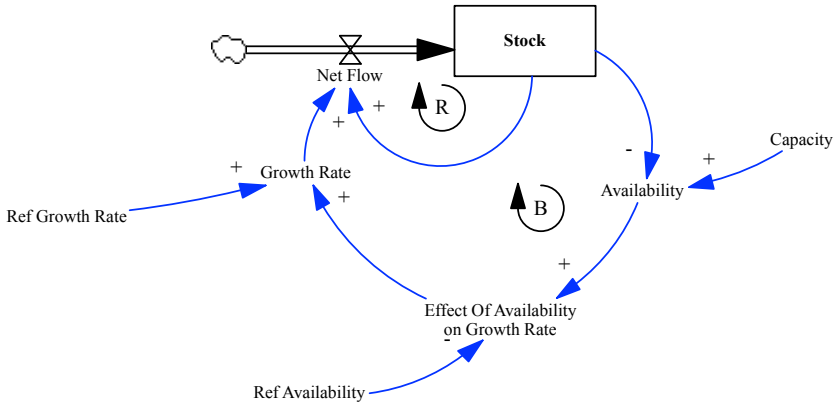


Equations

$$Capacity = 10000$$

$$Stock = INTEGRAL(Net\ Flow, 100)$$

$$Availability = 1 - \frac{Stock}{Capacity}$$

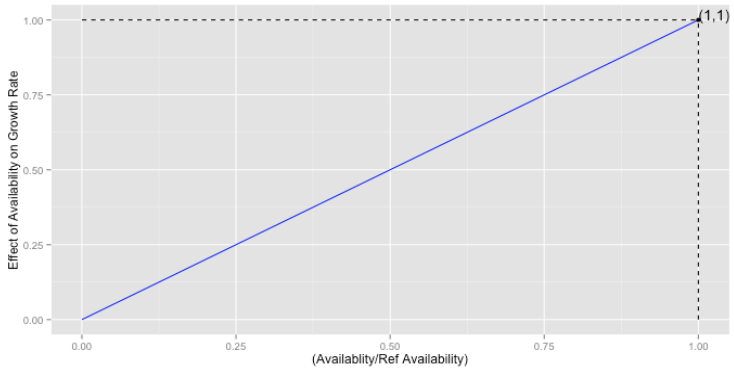


$$Ref\ Availability = 1.0 \qquad Ref\ Growth\ Rate = 0.10$$

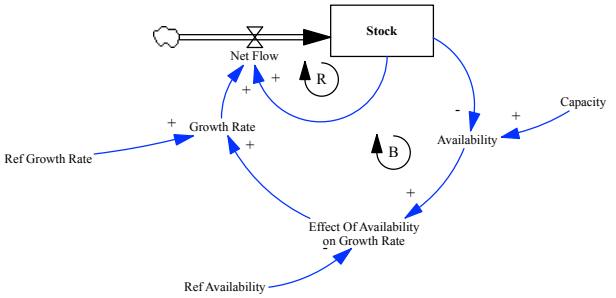
$$Effect\ of\ Availability\ on\ Growth\ Rate = \frac{Availability}{Ref\ Availability}$$

$$Growth\ Rate = Ref\ Growth\ Rate \times Effect\ of\ Availability\ on\ Growth\ Rate$$

$$Net\ Flow = Stock \times Growth\ Rate$$

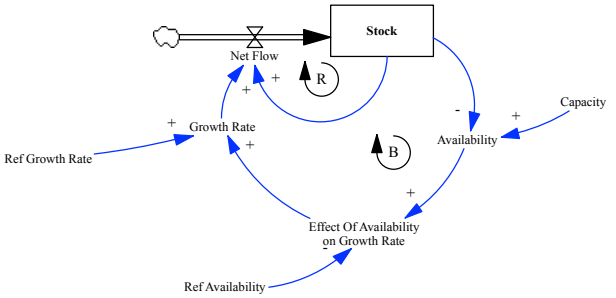


R Code (1/3)




```
library(deSolve)
library(ggplot2)
require(gridExtra)

START<-0; FINISH<-100; STEP<-0.25

simtime <- seq(START, FINISH, by=STEP)

stocks  <- c(sStock=100)

auxs    <- c(aCapacity=10000, aRef.Availability=1, aRef.GrowthRate=0.10)
```

R Code (2/3)



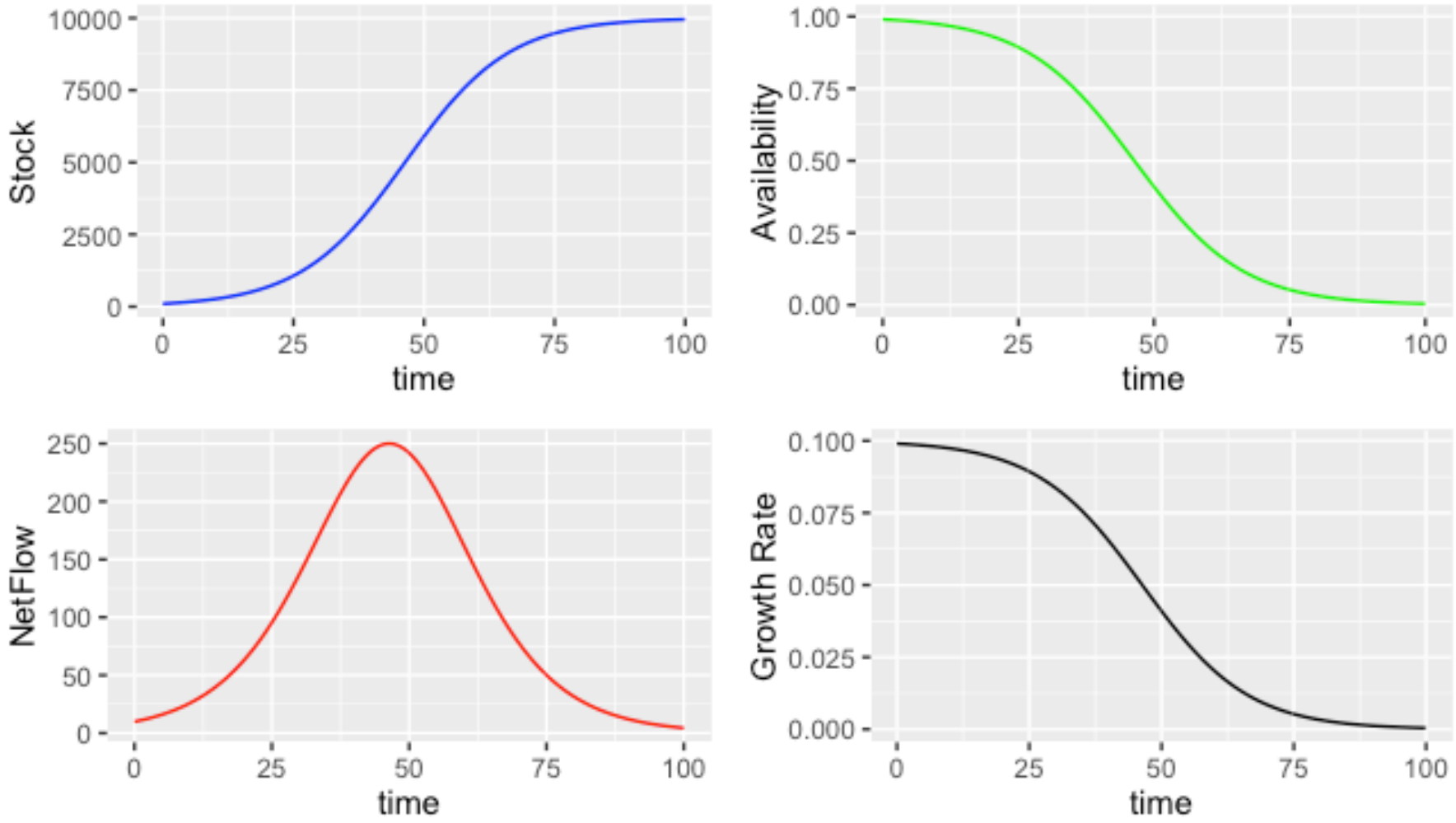

```
model <- function(time, stocks, auxs){
  with(as.list(c(stocks, auxs)),{
    aAvailability <- 1 - sStock / aCapacity
    aEffect <- aAvailability / aRef.Availability
    aGrowth.Rate <- aRef.GrowthRate * aEffect

    fNet.Flow <- sStock * aGrowth.Rate

    d_sStock_dt  <- fNet.Flow
    return (list(c(d_sStock_dt), NetFlow=fNet.Flow,
       GrowthRate=aGrowth.Rate, Effect=aEffect,  Availability=aAvailability))
  })
}
```

R Code (3/3)

```
o<-data.frame(ode(y=stocks, times=simtime, func = model,
                  parms=auxs, method="euler"))
```

```
> head(o[o$time %% 1 == 0,])
  time   sStock     NetFlow   GrowthRate    Effect      Availability
1    0 100.0000    9.90000   0.09900000   0.9900000   0.9900000
5    1 110.2696   10.90536   0.09889730   0.9889730   0.9889730
9    2 121.5812   12.01030   0.09878419   0.9878419   0.9878419
13   3 134.0377   13.22411   0.09865962   0.9865962   0.9865962
17   4 147.7519   14.55689   0.09852248   0.9852248   0.9852248
21   5 162.8467   16.01948   0.09837153   0.9837153   0.9837153
```

Plotting output…

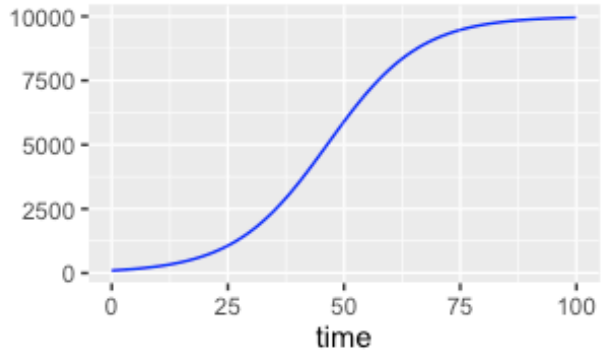


Challenge 2

- When the system reaches saturation point, double the capacity, and run again for another 100 units
- What impact do you think this would have on the system behaviour?
- Modify the model, rerun, and plot the results.

Atomic Behaviour Pattern

- We apply an attribute of simulation output: the *atomic behavior pattern* (Ford 1999)



$$\frac{\partial\left(\left|\frac{\partial x}{\partial t}\right|\right)}{\partial t} > 0$$

EXPONENTIAL

$$\frac{\partial\left(\left|\frac{\partial x}{\partial t}\right|\right)}{\partial t} < 0$$

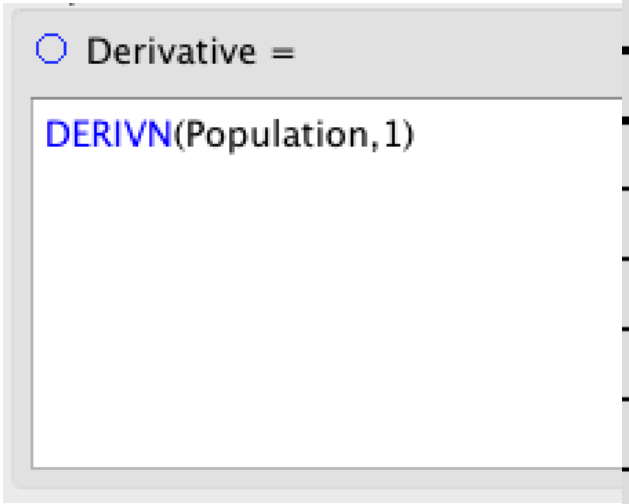
LOGARITHMIC

$$\frac{\partial\left(\left|\frac{\partial x}{\partial t}\right|\right)}{\partial t} = 0$$

LINEAR

DERIVN(<input>,<order>)

- In Stella, the function DERIVN is used to calculate the derivative of the variable of interest.



11:48 AM 7/1/16 Table 1 (Untitled

Time	Derivative	Population	Added
.00	0.00	10,000.00	500.00
.25	500.00	10,125.00	506.25
.50	506.25	10,251.56	512.58
.75	512.58	10,379.71	518.99
1.00	518.99	10,509.45	525.47
1.25	525.47	10,640.82	532.04

In R, diff() can be used

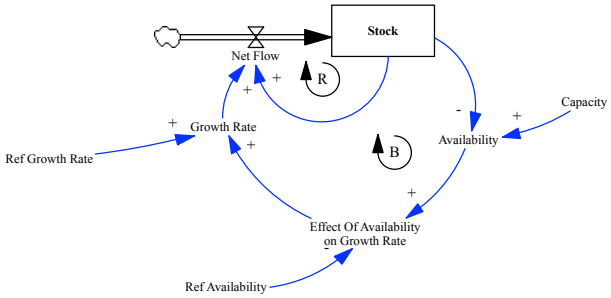
```
> diff(o$sPopulation)/diff(o$time)
 [1]  500.0000  506.2500  512.5781  518.9854  525.4727  532.0411  538.6916  545.4252
 [9]  552.2431  559.1461  566.1354  573.2121  580.3773  587.6320  594.9774  602.4146
[17]  609.9448  617.5691  625.2887  633.1048  641.0186  649.0313  657.1442  665.3585
[25]  673.6755  682.0965  690.6227  699.2555  707.9962  716.8461  725.8067  734.8793
[33]  744.0653  753.3661  762.7831  772.3179  781.9719  791.7466  801.6434  811.6639
[41]  821.8097  832.0824  842.4834  853.0144  863.6771  874.4731  885.4040  896.4715
[49]  907.6774  919.0234  930.5112  942.1426  953.9194  965.8434  977.9164  990.1403
[57] 1002.5171 1015.0486 1027.7367 1040.5834
```

11:48 AM 7/1/16 Table 1 (Untitled

Time	Derivative	Population	Added
.00	0.00	10,000.00	500.00
.25	500.00	10,125.00	506.25
.50	506.25	10,251.56	512.58
.75	512.58	10,379.71	518.99
1.00	518.99	10,509.45	525.47
1.25	525.47	10,640.82	532.04

Challenge 3

- Implement an R function that reproduces the behaviour of DERIVN, assuming the order is 1
- Format:
  - derivn(<variable of interest>)
- Ensure that the returned vector has the same number of elements as the input vector (see output from Stella example)



Calculating

$$\frac{\partial\left(\left|\frac{\partial x}{\partial t}\right|\right)}{\partial t}$$

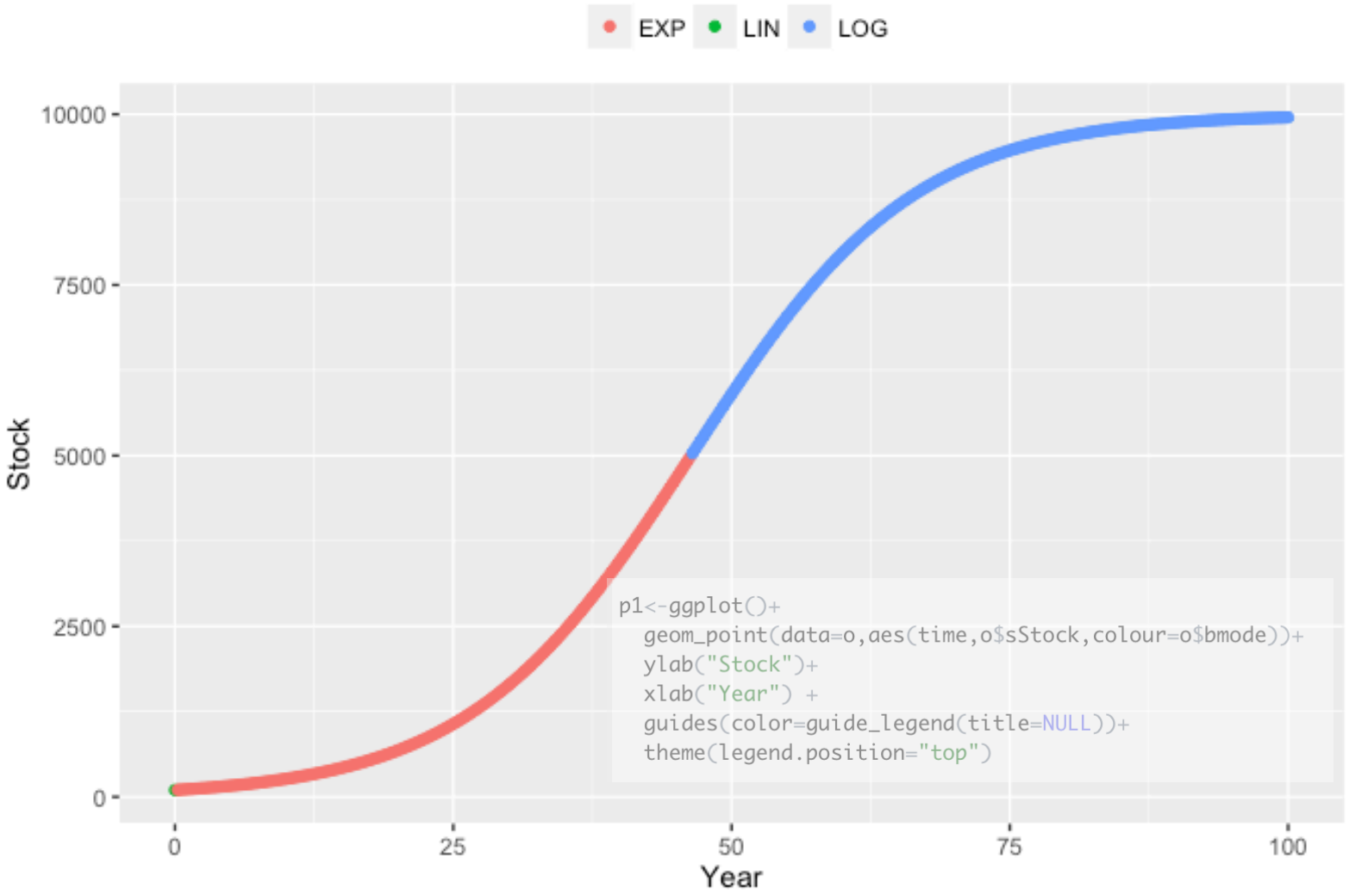
```
> o$abm<-derivn(abs(o$NetFlow),o$time)
> head(o)
  time   sStock  NetFlow GrowthRate    Effect Availability       abm
1 0.00 100.0000  9.90000 0.09900000 0.9900000    0.9900000 0.0000000
2 0.25 102.4750 10.14249 0.09897525 0.9897525    0.9897525 0.9699550
3 0.50 105.0106 10.39079 0.09894989 0.9894989    0.9894989 0.9932047
4 0.75 107.6083 10.64504 0.09892392 0.9892392    0.9892392 1.0169862
5 1.00 110.2696 10.90536 0.09889730 0.9889730    0.9889730 1.0413105
6 1.25 112.9959 11.17191 0.09887004 0.9887004    0.9887004 1.0661885
> tail(o)
      time   sStock  NetFlow   GrowthRate      Effect Availability        abm
396  98.75 9949.767 4.998072 0.0005023306 0.005023306  0.005023306 -0.5072746
397  99.00 9951.016 4.874360 0.0004898354 0.004898354  0.004898354 -0.4948483
398  99.25 9952.235 4.753680 0.0004776495 0.004776495  0.004776495 -0.4827201
399  99.50 9953.423 4.635959 0.0004657653 0.004657653  0.004657653 -0.4708833
400  99.75 9954.582 4.521126 0.0004541754 0.004541754  0.004541754 -0.4593311
401 100.00 9955.713 4.409112 0.0004428726 0.004428726  0.004428726 -0.4480570
```

Classifying Output

```
abm<-function(v){
  ifelse(v==0.0,"LIN",
        ifelse(v<0,"LOG","EXP"))
}
```

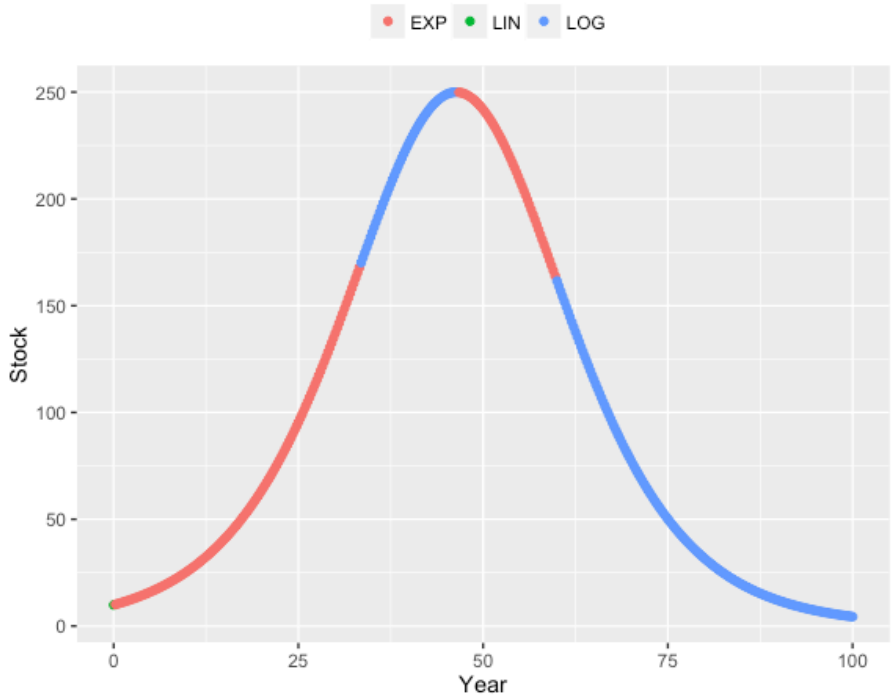
```
> o$bmode<-abm(o$abm)
> head(o)
  time   sStock  NetFlow GrowthRate    Effect Availability       abm bmode
1 0.00 100.0000  9.90000 0.09900000 0.9900000    0.9900000 0.0000000   LIN
2 0.25 102.4750 10.14249 0.09897525 0.9897525    0.9897525 0.9699550   EXP
3 0.50 105.0106 10.39079 0.09894989 0.9894989    0.9894989 0.9932047   EXP
4 0.75 107.6083 10.64504 0.09892392 0.9892392    0.9892392 1.0169862   EXP
5 1.00 110.2696 10.90536 0.09889730 0.9889730    0.9889730 1.0413105   EXP
6 1.25 112.9959 11.17191 0.09887004 0.9887004    0.9887004 1.0661885   EXP
> tail(o)
      time   sStock  NetFlow   GrowthRate      Effect Availability        abm bmode
396  98.75 9949.767 4.998072 0.0005023306 0.005023306  0.005023306 -0.5072746   LOG
397  99.00 9951.016 4.874360 0.0004898354 0.004898354  0.004898354 -0.4948483   LOG
398  99.25 9952.235 4.753680 0.0004776495 0.004776495  0.004776495 -0.4827201   LOG
399  99.50 9953.423 4.635959 0.0004657653 0.004657653  0.004657653 -0.4708833   LOG
400  99.75 9954.582 4.521126 0.0004541754 0.004541754  0.004541754 -0.4593311   LOG
401 100.00 9955.713 4.409112 0.0004428726 0.004428726  0.004428726 -0.4480570   LOG
```

ggplot()


EXP
LIN
LOG
10000
7500
5000
2500
0
Stock
0
25
50
75
100
Year
p1<-ggplot()+
geom_point(data=o,aes(time,o$sStock,colour=o$bmode))+
ylab("Stock")+
xlab("Year") +
guides(color=guide_legend(title=NULL))+
theme(legend.position="top")

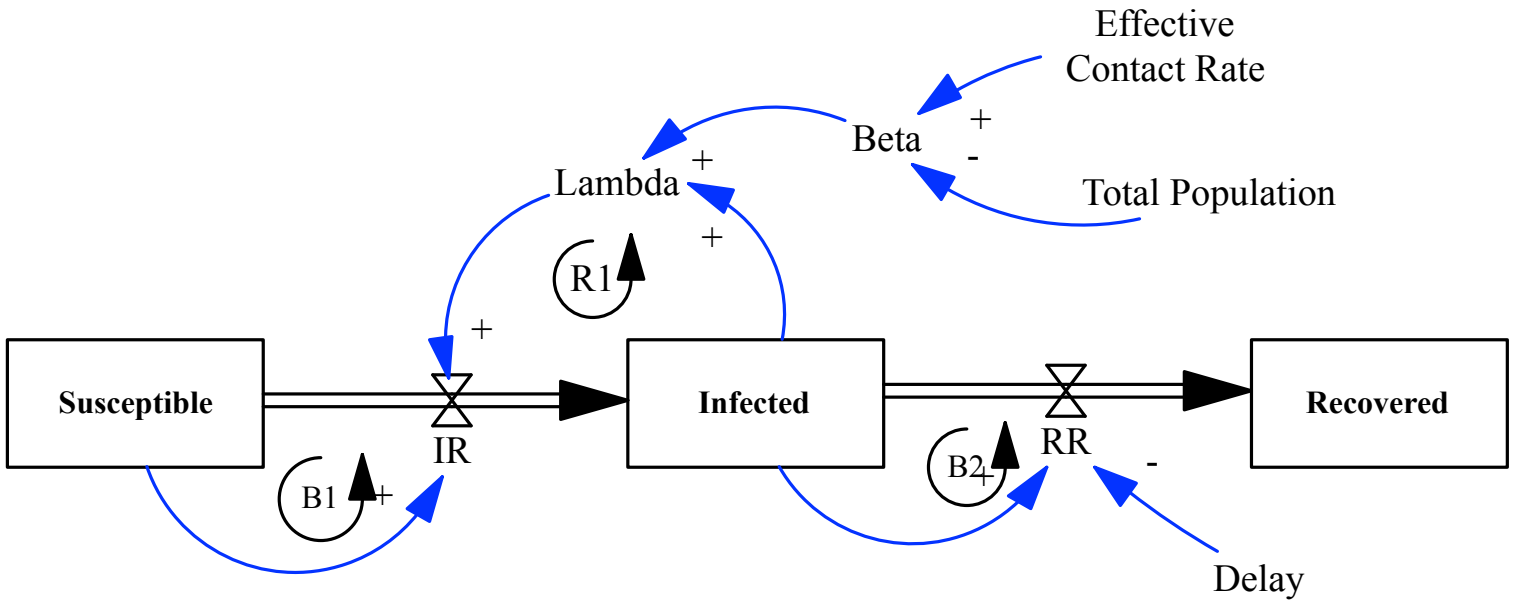
Challenge 4

- Reproduce the behaviour mode analysis with the net flow as the variable of interest



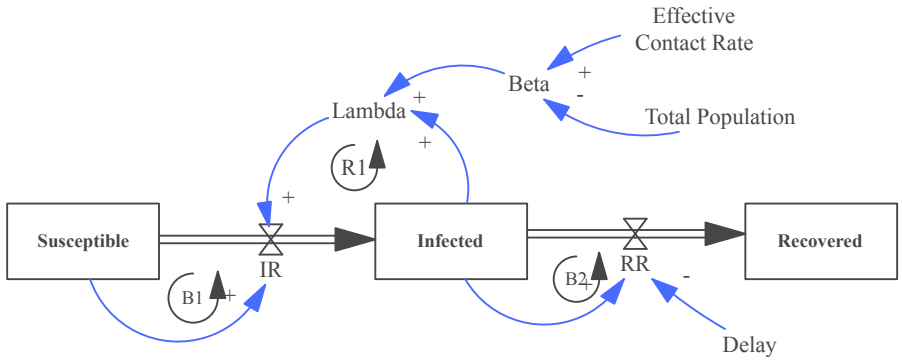
(3) Sensitivity Analysis – SIR Model

Diffusion - a fundamental process in physical, biological, social and economic settings (Rahmandad and Sterman 2008).



Model Equations

```
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))
  })
}
```



Sensitivity Analysis (library FME)

- Vary key parameters
- 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>, Thomas Petzoldt <thomas.petzoldt@tu-dresden.de>

Maintainer Karline Soetaert <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.

(a) Sample parameters – setup in a data frame, with cols (min, max)

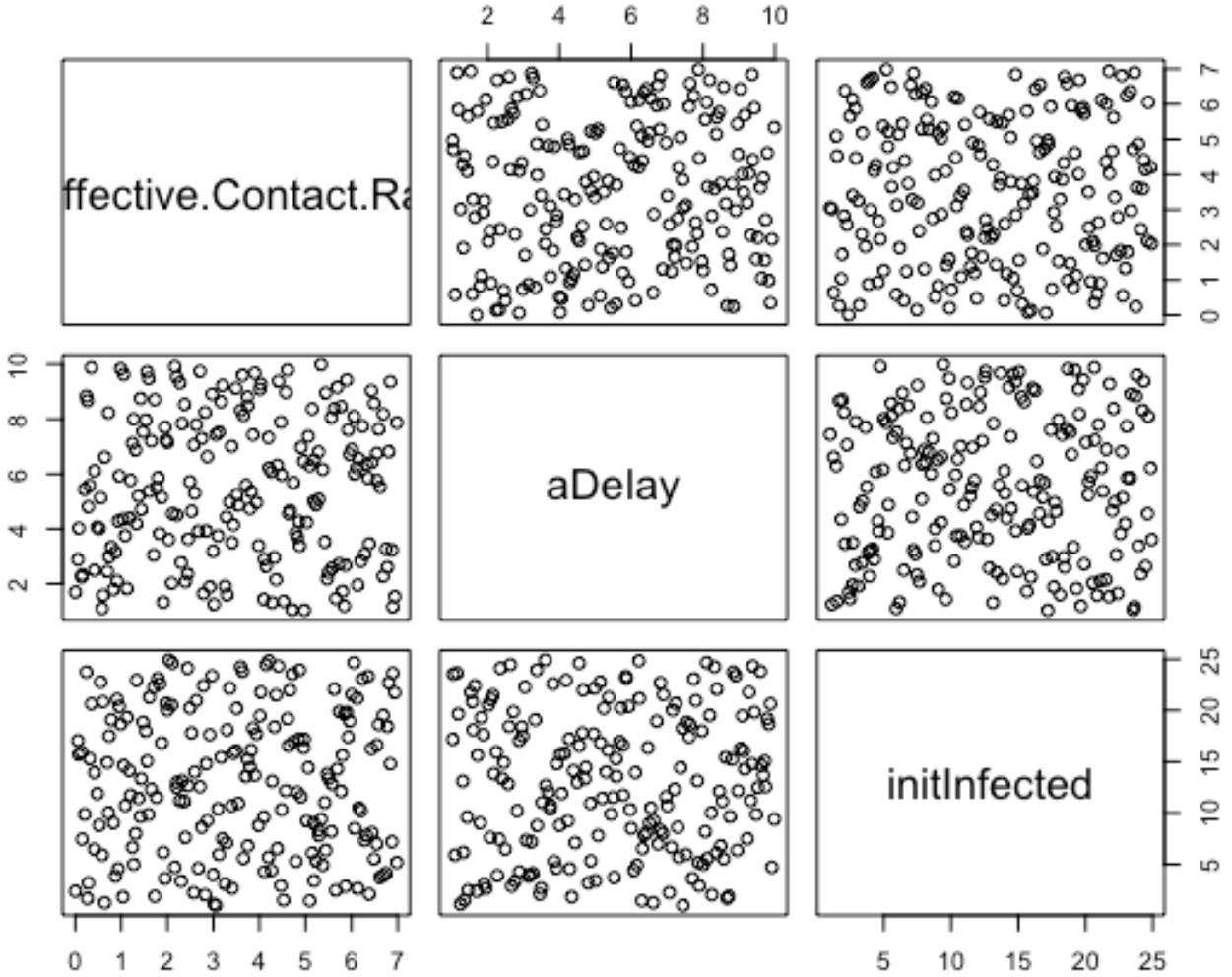
```
CE.MIN<-0;        CE.MAX<-7.0
DEL.MIN<-1.0;     DEL.MAX<-10.0
INIT.INF.MIN<-1.0;  INIT.INF.MAX<-25.0;
parRange<-data.frame(
  min=c(CE.MIN, DEL.MIN, INIT.INF.MIN),
  max=c(CE.MAX, DEL.MAX, INIT.INF.MAX)
)
rownames(parRange)<-c("aEffective.Contact.Rate", "aDelay", "initInfected")
```

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

(b) Create parameter data set (N runs) with Latinhyper function

```
> NRUNS<-10
> p<-data.frame(Latinhyper(parRange,NRUNS))
> p
   aEffective.Contact.Rate   aDelay initInfected
1                5.9922916 7.490509    12.618530
2                1.9240986 2.891320    21.151183
3                3.4934403 1.759654     4.529824
4                6.5973576 9.479128    20.137005
5                4.3541035 3.732457     9.108786
6                5.2924428 5.531295    16.960297
7                2.2544927 8.981136     1.076673
8                0.4624101 7.014942    24.835231
9                1.2701479 5.258102     8.145877
10               4.0972827 1.927919    15.307400
```

Explore data set plot(p) (N=200)



(c) Write a sensitivity function

```
g.simRuns<-list()
sensRun<-function(p){

  g.simRuns<<-list(length=nrow(p))

  for(i in 1:nrow(p)){
    init<-p[i,"initInfected"]

    a<-c(aTotalPopulation=10000,p[i,1:3])

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

    o<-data.frame(ode(y=stocks, simtime, func = model,
                      parms=a, method="euler"))
    o$run<-i
    g.simRuns[[i]]<<-o
  }
}
```

Exploring the output…

```
NRUNS<-2
p<-data.frame(Latinhyper(parRange,NRUNS))
sensRun(p)
```

```
> str(g.simRuns[[1]])
'data.frame':   401 obs. of  12 variables:
 $ time        : num  0 0.125 0.25 0.375 0.5 ...
 $ sSusceptible: num  9976 9972 9968 9964 9958 ...
 $ sInfected   : num  24.2 27.4 31.1 35.3 40 ...
 $ sRecovered  : num  0 0.361 0.771 1.236 1.762 ..
 $ IR          : num  28.8 32.7 37 42 47.6 ...
 $ RR          : num  2.89 3.28 3.72 4.21 4.78 ...
 $ Beta        : num  0.00012 0.00012 0.00012 0.00
 $ Lambda      : num  0.00289 0.00328 0.00372 0.00
 $ DEL         : num  8.37 8.37 8.37 8.37 8.37 ...
 $ CE          : num  1.2 1.2 1.2 1.2 1.2 ...
 $ InitI       : num  24.2 24.2 24.2 24.2 24.2 ...
 $ run         : int  1 1 1 1 1 1 1 1 1 1 ...
```

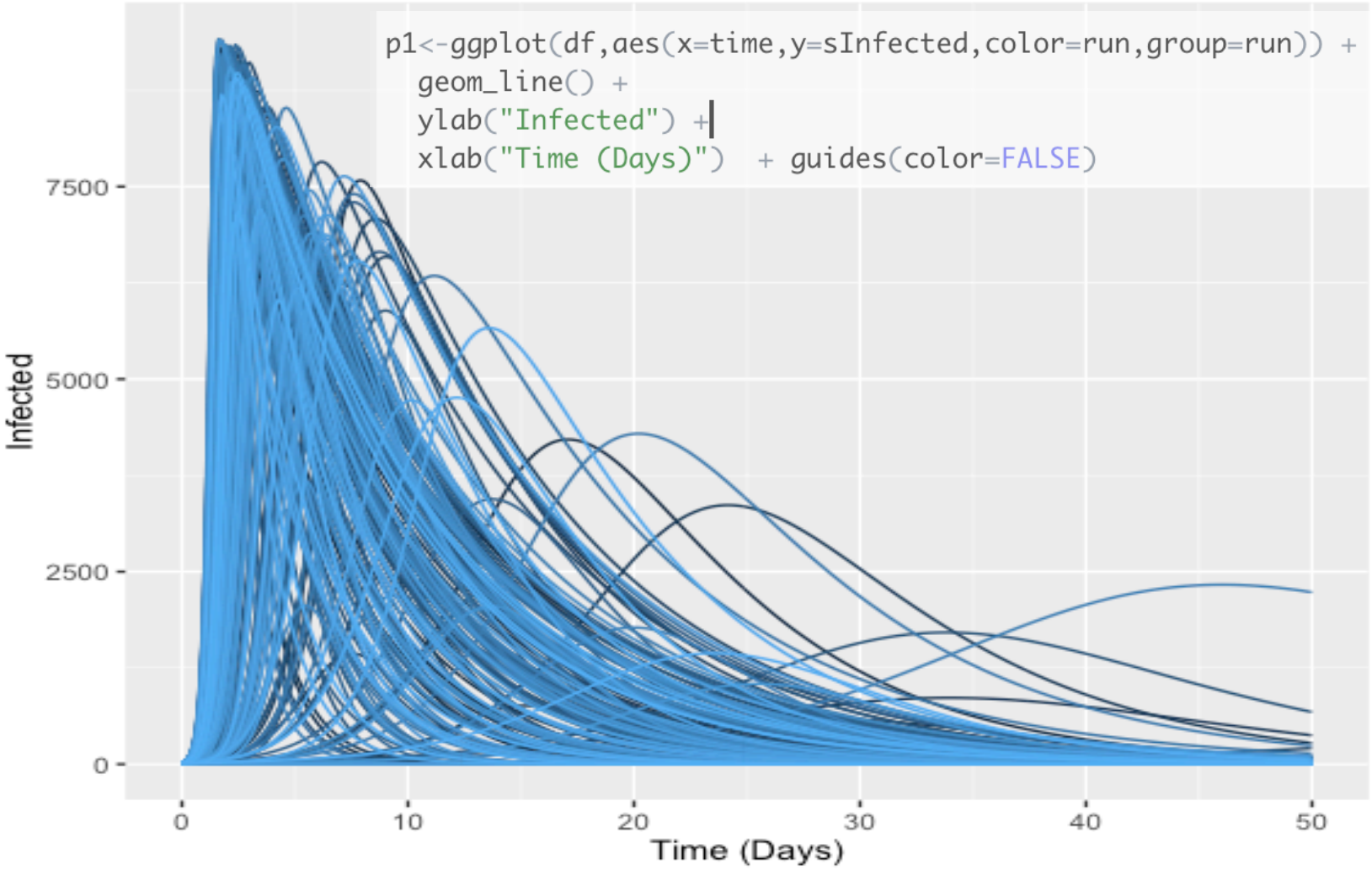
```
> str(g.simRuns[[2]])
'data.frame':   401 obs. of  12 variables:
 $ time        : num  0 0.125 0.25 0.375 0.5 .
 $ sSusceptible: num  9992 9986 9976 9958 9927
 $ sInfected   : num  7.95 13.73 23.7 40.91 70
 $ sRecovered  : num  0 0.226 0.617 1.292 2.45
 $ IR          : num  48 82.9 143 246.4 423.6
 $ RR          : num  1.81 3.13 5.4 9.31 16.06
 $ Beta        : num  0.000605 0.000605 0.0006
 $ Lambda      : num  0.00481 0.0083 0.01434 0
 $ DEL         : num  4.39 4.39 4.39 4.39 4.39
 $ CE          : num  6.05 6.05 6.05 6.05 6.05
 $ InitI       : num  7.95 7.95 7.95 7.95 7.95
 $ run         : int  2 2 2 2 2 2 2 2 2 2 ...
```

Merge into one data frame
rbind.fill {plyr}

```
> df<-rbind.fill(g.simRuns)
> str(df)
'data.frame':   802 obs. of  12 variables:
 $ time        : num  0 0.125 0.25 0.375 0.5 ...
 $ sSusceptible: num  9976 9972 9968 9964 9958 ...
 $ sInfected   : num  24.2 27.4 31.1 35.3 40 ...
 $ sRecovered  : num  0 0.361 0.771 1.236 1.762 ...
 $ IR          : num  28.8 32.7 37 42 47.6 ...
 $ RR          : num  2.89 3.28 3.72 4.21 4.78 ...
 $ Beta        : num  0.00012 0.00012 0.00012 0.00012 0.00012 ...
 $ Lambda      : num  0.00289 0.00328 0.00372 0.00421 0.00478 ...
 $ DEL         : num  8.37 8.37 8.37 8.37 8.37 ...
 $ CE          : num  1.2 1.2 1.2 1.2 1.2 ...
 $ InitI       : num  24.2 24.2 24.2 24.2 24.2 ...
 $ run         : int  1 1 1 1 1 1 1 1 1 1 ...
```

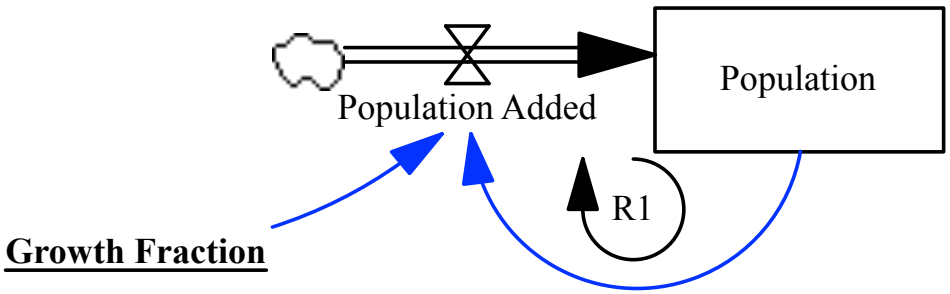
Plot Results

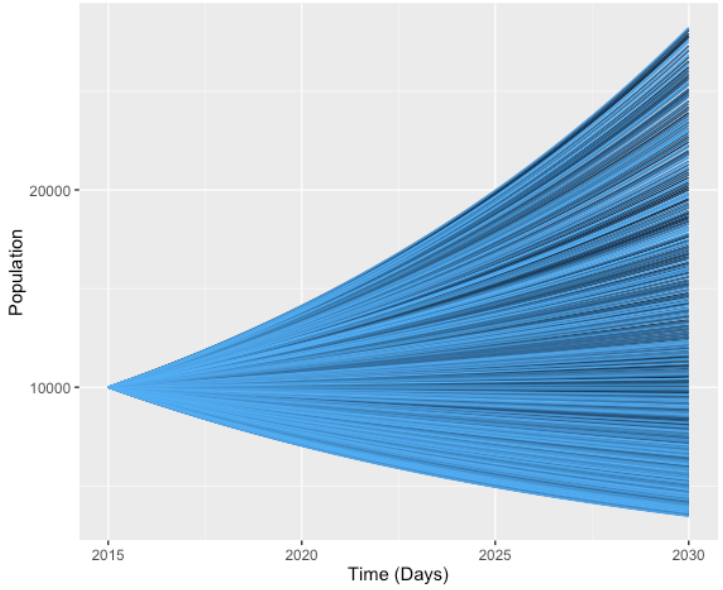
```
p1<-ggplot(df,aes(x=time,y=sInfected,color=run,group=run)) +
  geom_line() +
  ylab("Infected") +
  xlab("Time (Days)")  + guides(color=FALSE)
```



Challenge 5

- For the problem specified in Challenge 1, perform a sensitivity analysis (N=200) with the growth fraction sampled between the range [-0.07 and +0.07]





(4) 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).

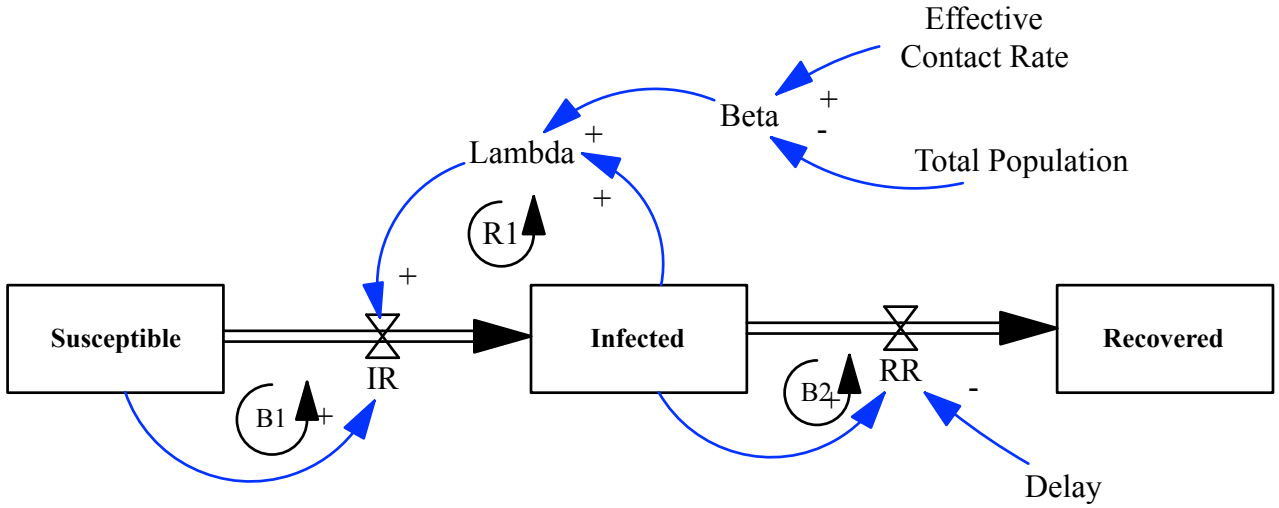
$$r = \frac{\sum (X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{\sum (X_i - \bar{X})^2 \sum (Y_i - \bar{Y})^2}}$$

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
$Infected_{INIT}$	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)

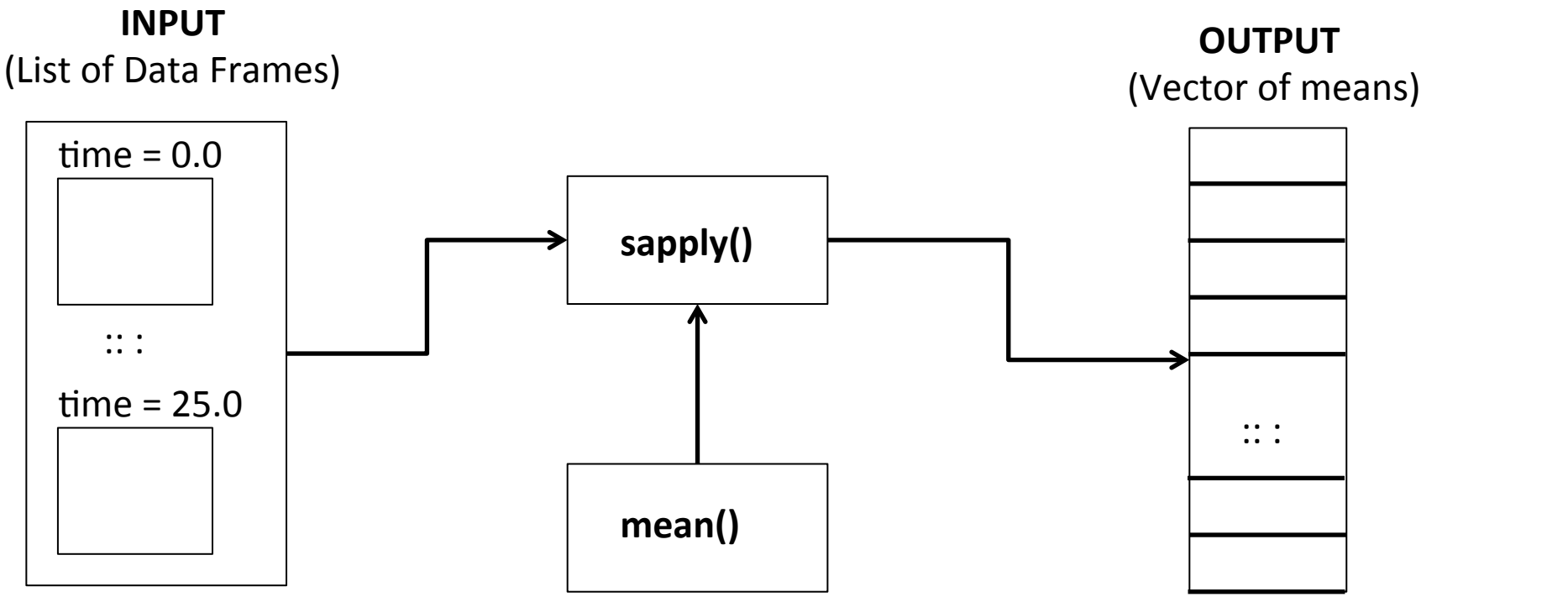
- Another use of user-defined functions in R is as parameters to the *apply* family of functions.
- The general form of the **sapply(x,f,fargs)** function is as follows:
  - **x** is the target vector or list
  - **f is the function to be called and applied to each element**
  - **fargs** are the optional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

Get the average at each time

using an anonymous function

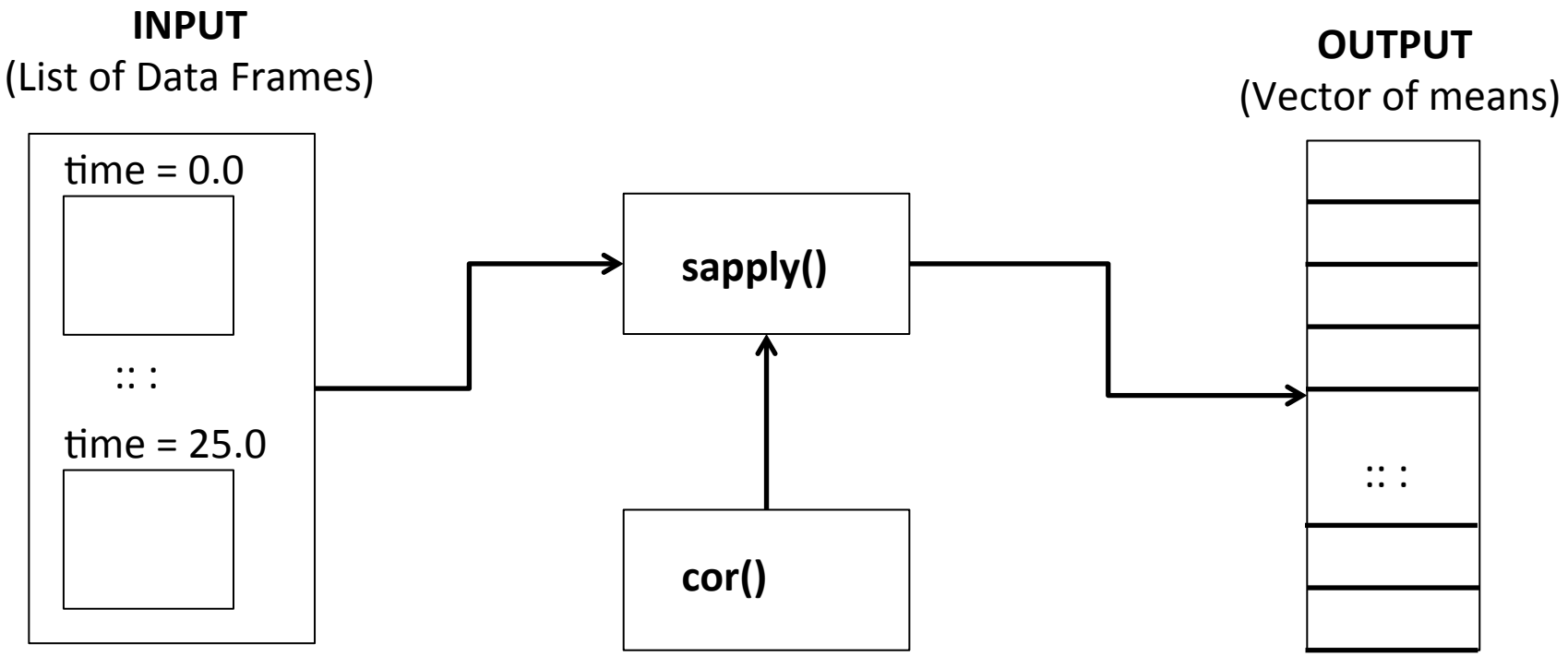
```
av.Infected<-sapply(runs,function(l){mean(l$sInfected)})
```

```
> 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
```

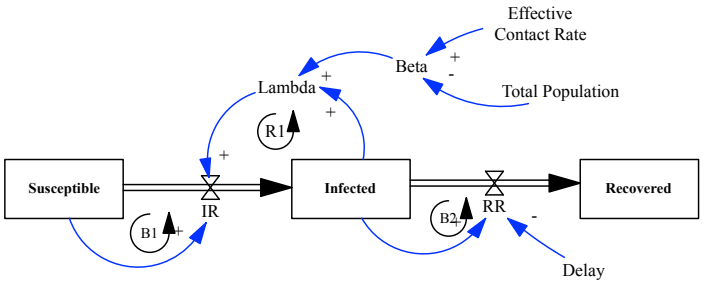


Calculating cor()

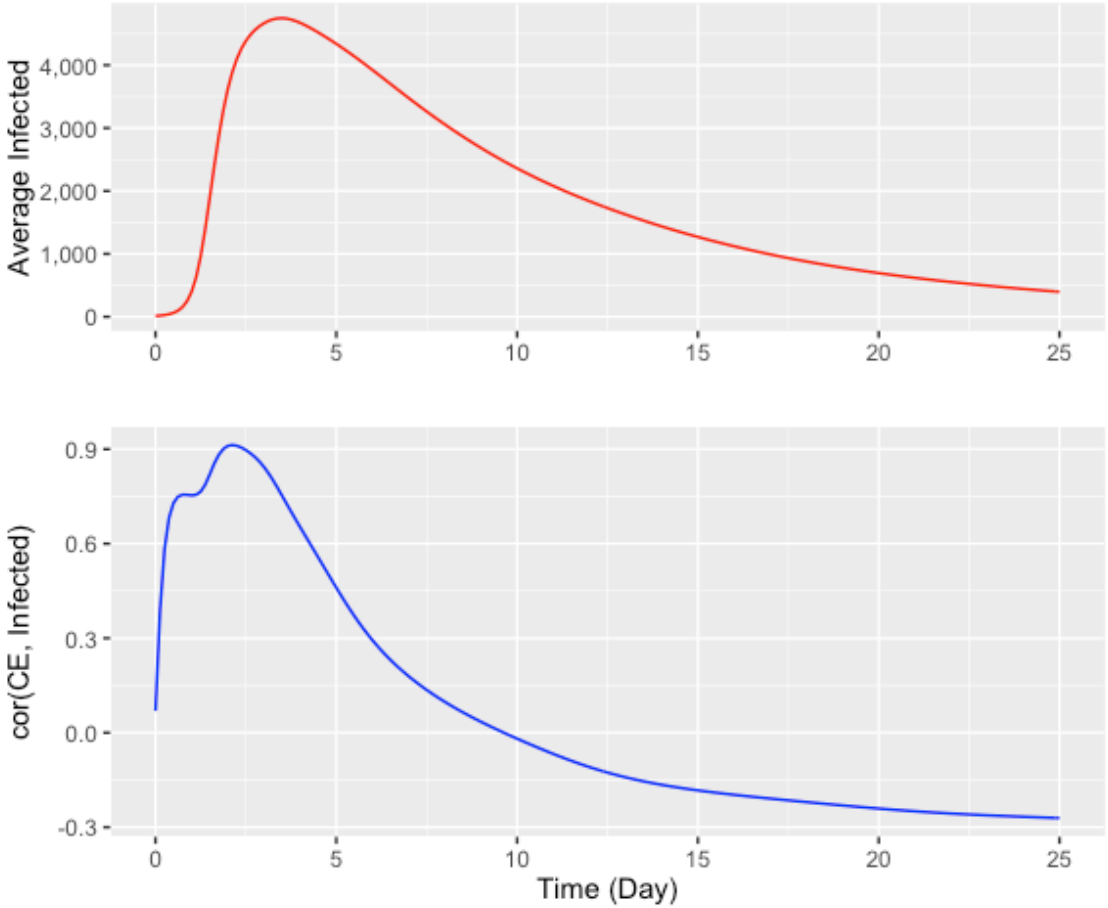
```
> cor.CE<-sapply(runs,function(l){cor(l$sInfected, l$CE)})
> round(cor.CE[1:10],3)
    0 0.125  0.25 0.375   0.5 0.625  0.75 0.875     1 1.125
0.069 0.388 0.583 0.683 0.730 0.750 0.755 0.755 0.753 0.755
```



Visualising Output



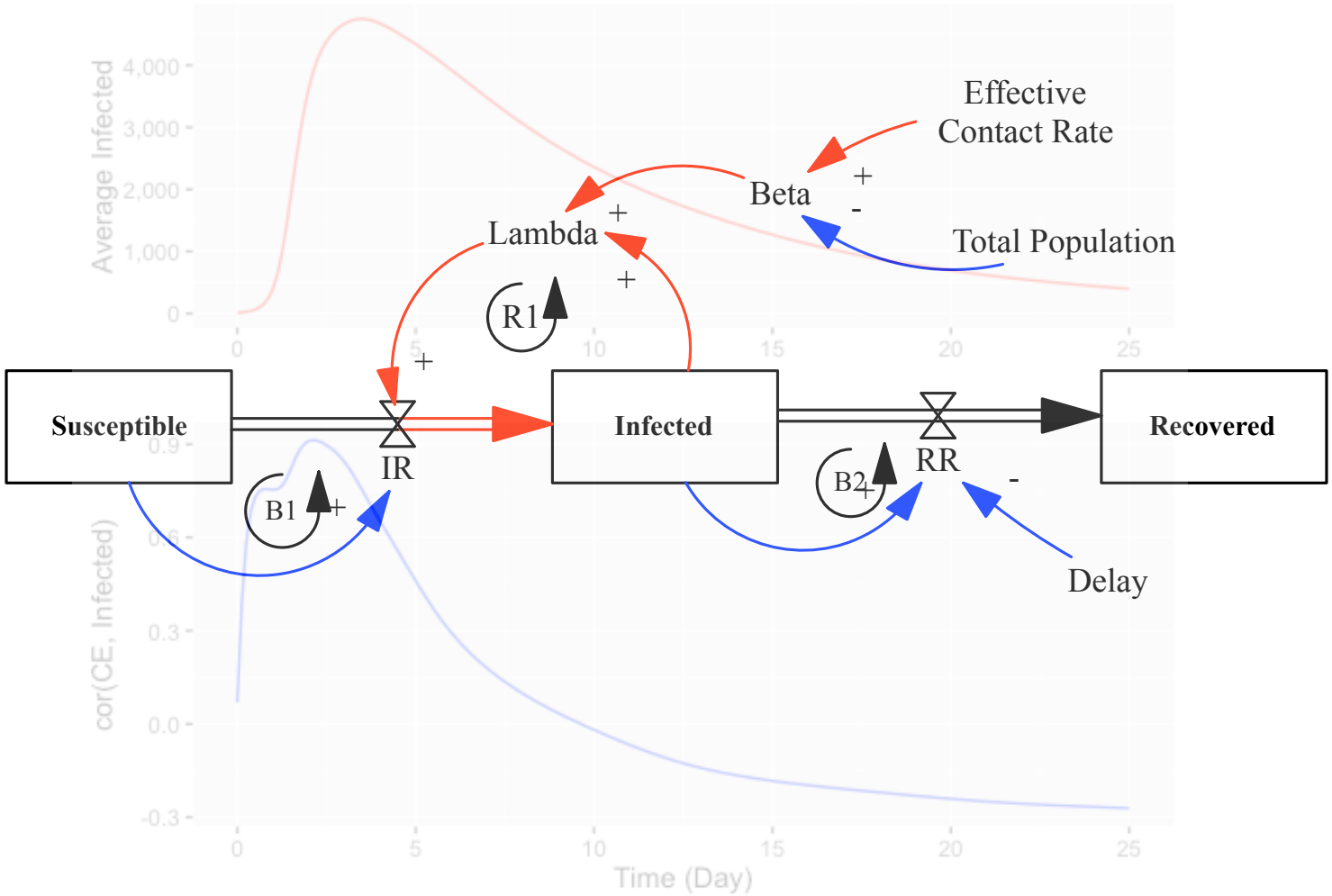

```
> summary(cor.CE)
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
-0.27090 -0.22810 -0.12640  0.06012  0.24550  0.91310
```



Steps (based on Taylor et al. 2009)

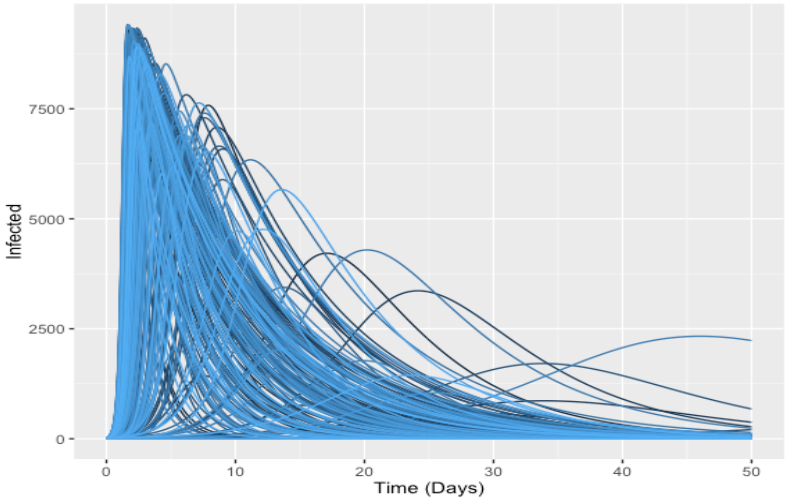
1. Select a set of exogenous model parameters, and a system performance variable for analysis. Select appropriate parameter ranges.
2. Calculate the correlation coefficients between the selected exogenous model parameters and the system performance variables. Plot the correlation coefficients and the 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.**

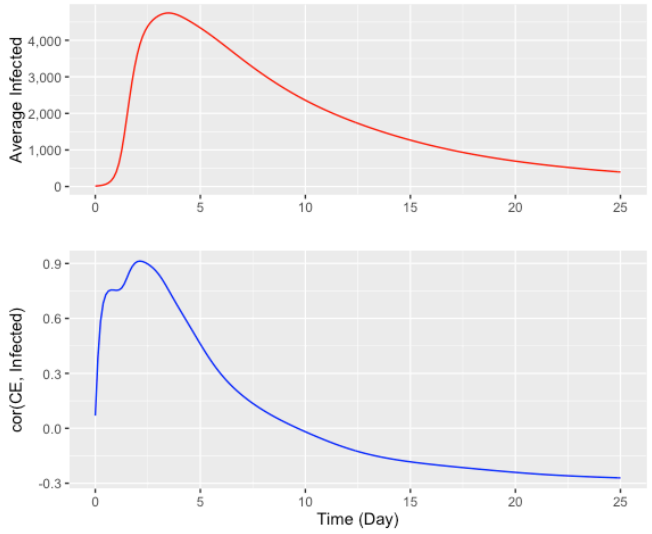
High leverage model structure (Contagion Loop)



Challenge 6

- Perform statistical screening in R on your model of choice (either with deSolve in R or importing model results from another SD tool).





References

- Ford, A., & Flynn, H. (2005). Statistical screening of system dynamics models. *System Dynamics Review*, *21*(4), 273-303.
- Meadows, D. H. (2008). *Thinking in systems: A primer*. Chelsea Green Publishing.
- Taylor, T. R., Ford, D. N., & Ford, A. (2010). Improving model understanding using statistical screening. *System Dynamics Review*, 26(1), 73-87.