Review of random networks and Markov models

In this notebook, you will be able to review the generation of different random networks and practice implementing Markov models that allow you to simulate infections that work with the adjacency matrices we generate

Installing and loading libraries If you do not have the following libraries installed, run the following code block to install them.

```
#install.packages(c("deSolve","sna","igraph","network","ggnet2", "intergraph", "GGally"))
```

Load the libraries that we are going to use

```
library(tidyverse)
library(igraph)
library(sna)
library(network)
library(ggplot2)
library(GGally)
```

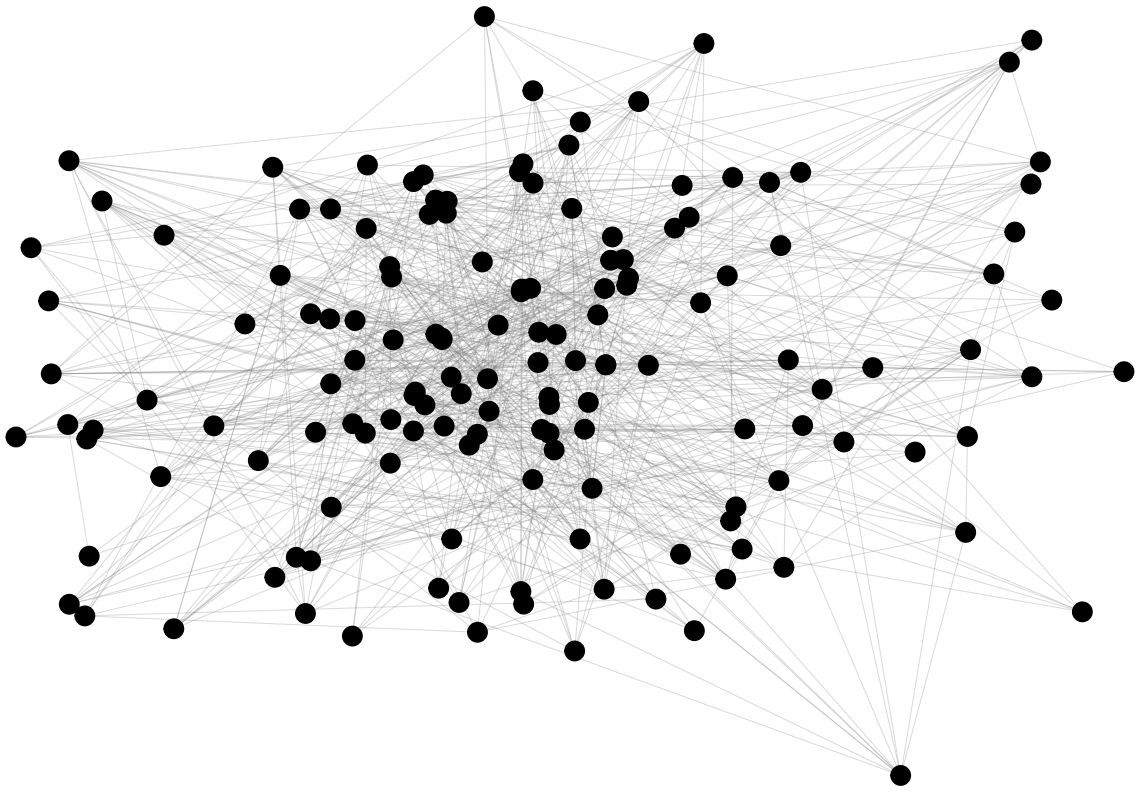
Generating random networks Let's start with the most common networks, starting with the Erdos-Renyi network. For this, we will use the igraph library with its erdos.renyi.game function. This function is as follows:

```
erdos.renyi.game(
  n,
  p.or.m,
  type = c("gnp", "gnm"),
  directed = FALSE,
  loops = FALSE
)
```

It receives different parameters, but for this exercise we will only be interested in n, which corresponds to the number of nodes in the network, p, which is the probability of creating an edge between two nodes, and directed to configure whether the network is directed or undirected

Let's see how to create a network and graph it:

```
g1 <- erdos.renyi.game(n = 150, p = 10 / 150)
ggnet2(g1,mode = "spring", vjust = -1, size = 3, color = 1,edge.alpha = 0.3,edge.size = 0.1)
```



Now we are going to extract the adjacency matrix from the network we just created. For that, we are going to use the as_adjacency_matrix function, which receives as a parameter the network from which the matrix will be extracted

```
A1 <- as.matrix(as_adjacency_matrix(g1))
```

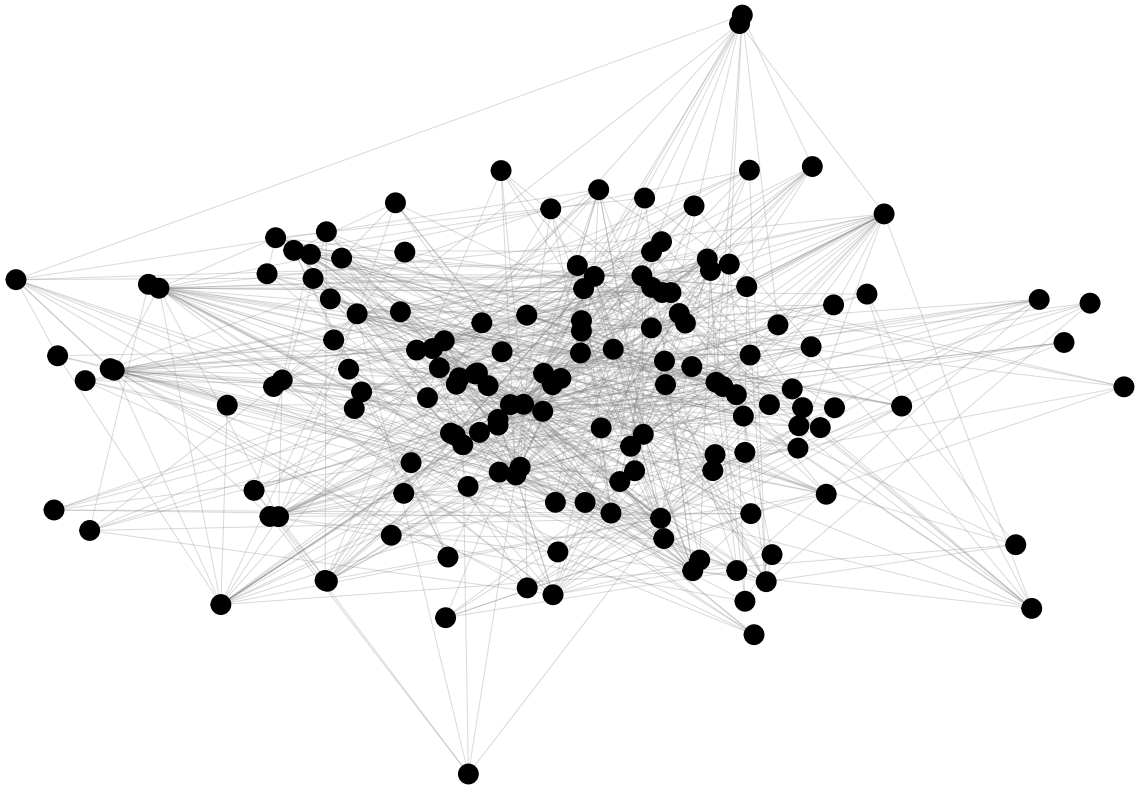
Now let's create a scale free network using the Barabasi-Albert model. The function to use will be **sample_pa**, which has the following form:

```
sample_pa(
  n,
  power = 1,
  m = NULL,
  out.dist = NULL,
  out.seq = NULL,
  out.pref = FALSE,
  zero.appeal = 1,
  directed = TRUE,
  algorithm = c("psumtree", "psumtree-multiple", "bag"),
  start.graph = NULL
)
```

This function receives different parameters, but for this tutorial we will only be interested in n, which corresponds to the number of nodes in the network, *power*, which is the power of the *preferential attachment* and by default is set to 1 (*linear preferential attachment*), and *directed* to configure whether the network is directed or undirected.

Let's see how to create a network and graph it:

```
g2 <- sample_pa(n = 150, m = 5, directed = FALSE)
A2 <- as.matrix(as_adjacency_matrix(g2))
ggnet2(g2,mode = "spring", vjust = -1, size = 3, color = 1,edge.alpha = 0.3,edge.size = 0.1)
```



Cascade-type contagion

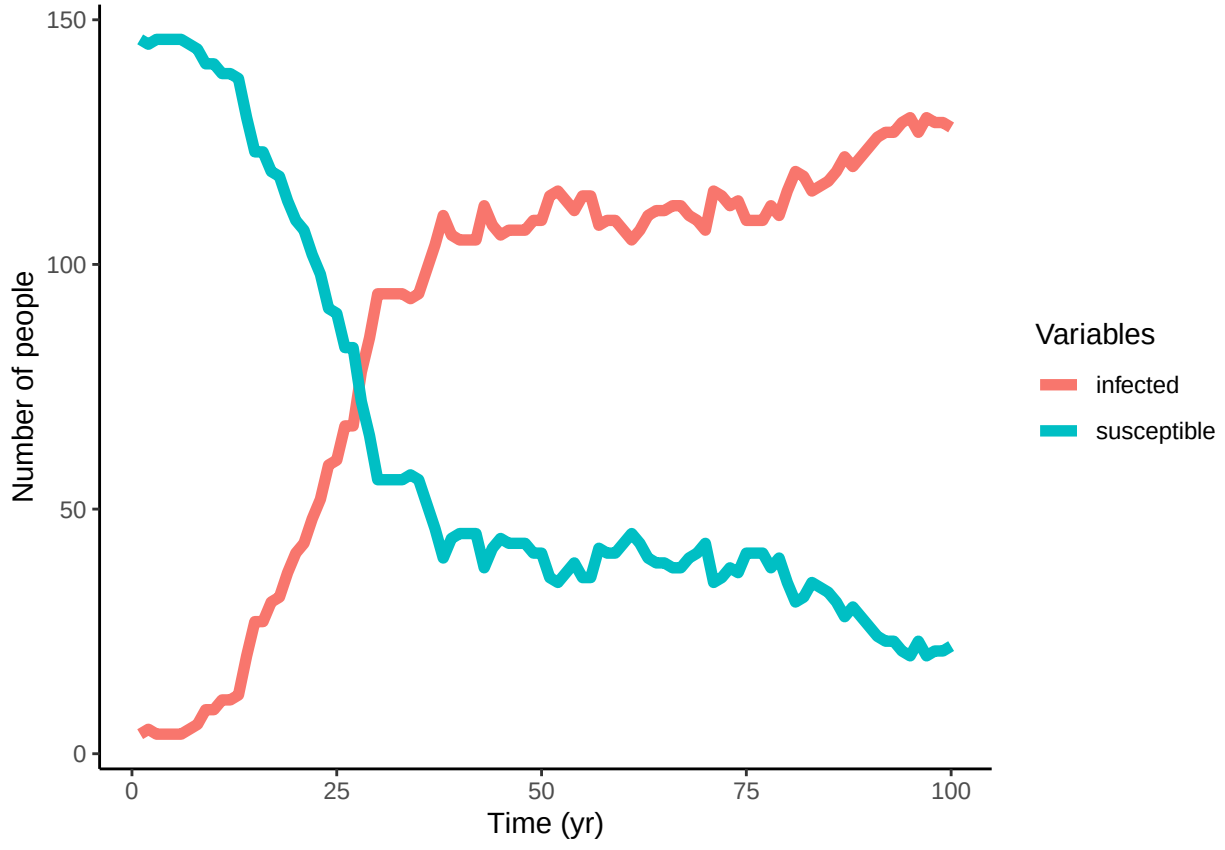
For the contagion, we will roll the dice for infection with all the neighbors of each node, using the adjacency matrix as a reference. Let's start by implementing the contagion for an SIS model

```
Tsim <- 100
lambda <- 0.02 #Contagion rate
gamma <- 0.035 #Recovery rate

#Lists to store the values for each t from infected and susceptible
susceptible <- rep(NA,Tsim)
infected <- rep(NA,Tsim)
state <- rbinom(150,1,5/150)
ids_nodes<-1:150

for(i in 1:Tsim){
  infected[i]<-sum(state==1) #Register the infected
  susceptible[i]<-sum(state==0) #Register the susceptible
```

```
  #Infection dynamics
  infected_nodes<-ids_nodes[state==1] #Extract the infected nodes
  for(j in 1:length(infected_nodes)){
    neighbours <- ids_nodes[A1[infected_nodes[j],]==1] #Extract the neighbours of each infected node
    if(length(neighbours)>0){
      random <- runif(length(neighbours),0,1)
      chance <- as.integer(random<=lambda)
      state[neighbours]<-ifelse(state[neighbours]==0,state[neighbours]+chance,state[neighbours])
    }
  }
  #Recovery dynamics
  random <- runif(length(infected_nodes),0,1)
  chance <- as.integer(random<=gamma)
  state[infected_nodes]<-state[infected_nodes]-chance
}
```

Let's graph the behaviour of the curve:

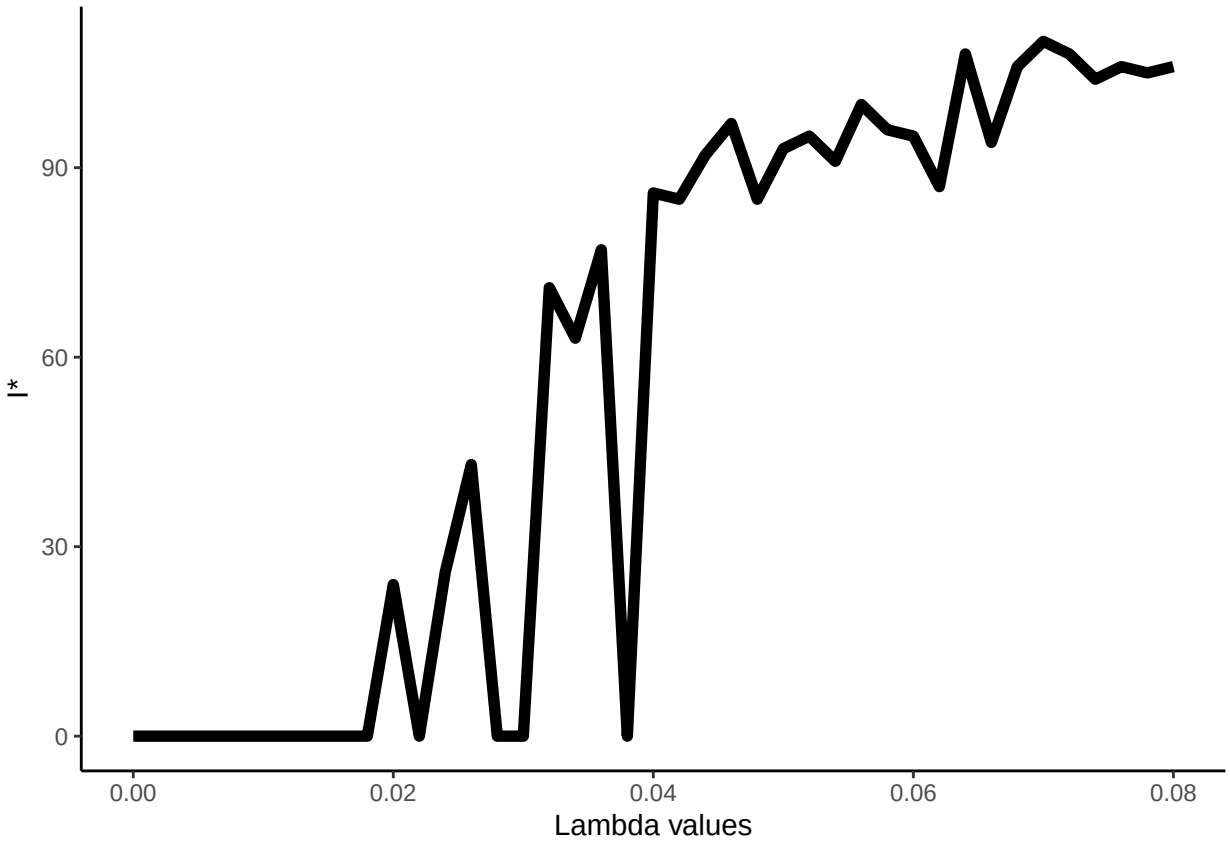
```
output <-data.frame(time = 1:length(susceptible),susceptible,infected)
q<-output %>%
gather(variable,value,-time) %>%
ggplot(aes(x=time,y=value,color=variable))+
geom_line(linewidth=2)+
theme_classic()+
labs(x='Time (yr)',y='Number of people',color = "Variables")
q
```



Just as we did in the first notebook, we are going to plot how the number of infected individuals changes when the system stabilizes under different variations of lambda.

```
lambdas<-seq(from=0,to=0.08,by =0.002)
gammas<-rep(0.15,length(lambdas))
Istable <- rep(NA,length(lambdas))
```

```
for(t in 1:length(lambdas)){

  Tsim <- 250
  lambda <- lambdas[t] #Contagion rate
  gamma <- gammas[t] #Recovery rate

  #Lists to store the values for each t from the infected and susceptible
  susceptible <- rep(NA,Tsim)
  infected <- rep(NA,Tsim)
  state <- rbinom(150,1,5/150)
  ids_nodes<-1:150

  for(i in 1:Tsim){

    infected[i]<-sum(state==1) #Register the infected
    susceptible[i]<-sum(state==0) #Register the susceptible

    #Infection dynamics
    infected_nodes<-ids_nodes[state==1] #Extract the infected nodes
    if(length(infected_nodes)>0){
      for(j in 1:length(infected_nodes)){

        neighbours <- ids_nodes[A1[infected_nodes[j],]==1] #Extract the neighbours of each infected nod
        if(length(neighbours)>0){

          random <- runif(length(neighbours),0,1)
          chance <- as.integer(random<=lambda)
          state[neighbours]<-ifelse(state[neighbours]==0,state[neighbours]+chance,state[neighbours])
        }
      }
    }

    #Recovery dynamics
    random <- runif(length(infected_nodes),0,1)
    chance <- as.integer(random<=gamma)
    state[infected_nodes]<-state[infected_nodes]-chance
  }
  Istable[t]<-infected[length(infected)]
}

I_graph<-Istable
output<-data.frame(lambdas,Istable)

ggplot(data = output,aes(x=lambdas,y=Istable))+
  geom_line(linewidth=2)+
  theme_classic()+
  labs(x='Lambda values',y='I*')
```



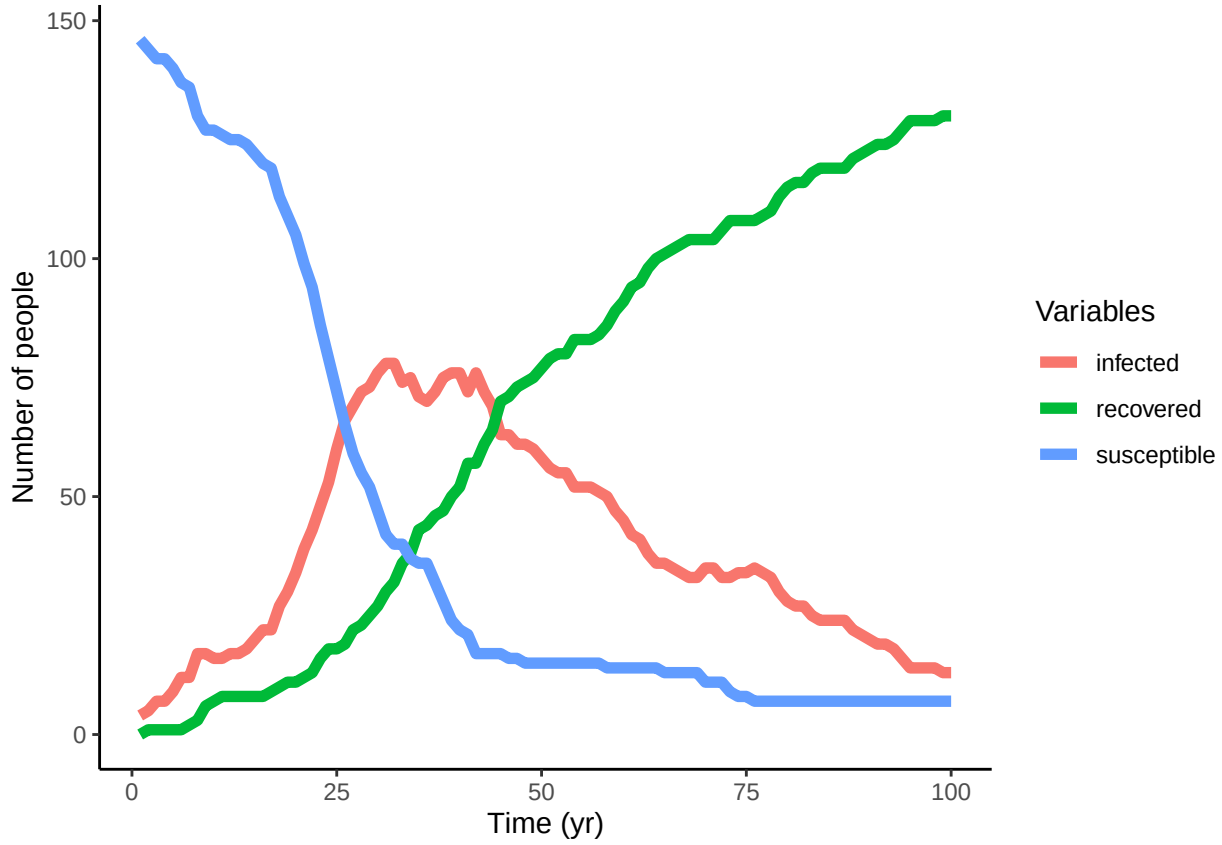
Exercise: now try to modify the two previous algorithms to obtain the plots for an SIR model.

```
Tsim <- 100
lambda <- 0.02 #Contagion rate
gamma <- 0.035 #Recovery rate

#Lists to store the values for each t from infected and susceptible
susceptible <- rep(NA,Tsim)
infected <- rep(NA,Tsim)
recovered <- rep(NA,Tsim)
state <- rbinom(150,1,5/150)
ids_nodes<-1:150

for(i in 1:Tsim){
  infected[i]<-sum(state==1) #Register the infected
  susceptible[i]<-sum(state==0) #Register the susceptible
  recovered[i]<-sum(state==2)

  #Infection dynamics
  infected_nodes<-ids_nodes[state==1] #Extract the infected nodes
  for(j in 1:length(infected_nodes)){
    neighbours <- ids_nodes[A1[infected_nodes[j],]==1] #Extract the neighbours of each infected node
    if(length(neighbours)>0){
      random <- runif(length(neighbours),0,1)
      chance <- as.integer(random<=lambda)
      state[neighbours]<-ifelse(state[neighbours]==0,state[neighbours]+chance,state[neighbours])
```

```
    }
  }
  #Recovery dynamics
  random <- runif(length(infected_nodes),0,1)
  chance <- as.integer(random<=gamma)
  state[infected_nodes]<-state[infected_nodes]+chance
}
```

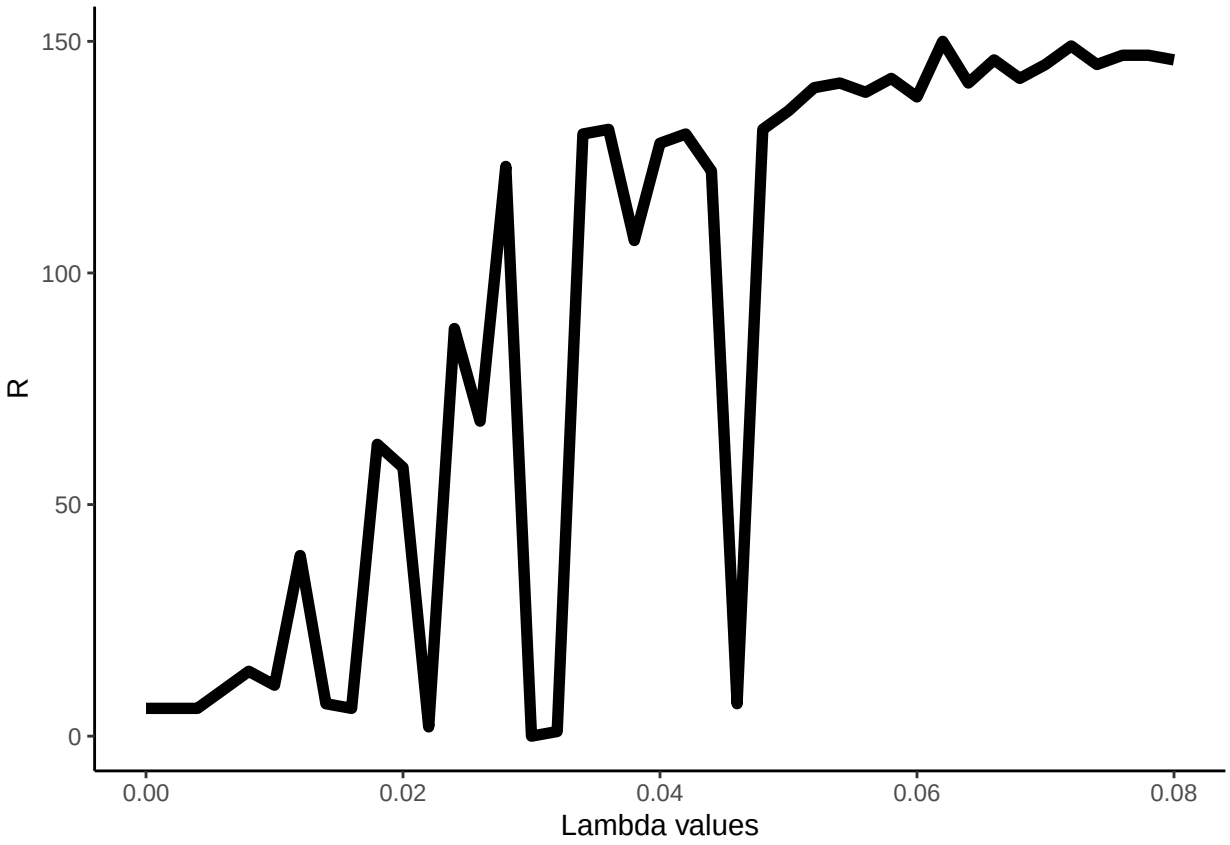
Let's graph the behaviour of the curve:

```
output <-data.frame(time = 1:length(susceptible),susceptible,infected,recovered)
q<-output %>%
gather(variable,value,-time) %>%
ggplot(aes(x=time,y=value,color=variable))+
geom_line(linewidth=2)+
theme_classic()+
labs(x='Time (yr)',y='Number of people',color = "Variables")
q
```




```
for(t in 1:length(lambdas)){

  Tsim <- 250
  lambda <- lambdas[t] #Contagion rate
  gamma <- gammas[t] #Recovery rate
```

```r
  #Lists to store the values for each t from the infected and susceptible
  susceptible <- rep(NA,Tsim)
  infected <- rep(NA,Tsim)
  recovered <- rep(NA,Tsim)
  state <- rbinom(150,1,5/150)
  ids_nodes<-1:150

  for(i in 1:Tsim){

    infected[i]<-sum(state==1) #Register the infected
    susceptible[i]<-sum(state==0)
    recovered[i]<-sum(state==2)#Register the susceptible

    #Infection dynamics
    infected_nodes<-ids_nodes[state==1] #Extract the infected nodes
    if(length(infected_nodes)>0){
      for(j in 1:length(infected_nodes)){

        neighbours <- ids_nodes[A1[infected_nodes[j],]==1] #Extract the neighbours of each infected nod
        if(length(neighbours)>0){

          random <- runif(length(neighbours),0,1)
          chance <- as.integer(random<=lambda)
          state[neighbours]<-ifelse(state[neighbours]==0,state[neighbours]+chance,state[neighbours])
        }
      }
    }

    #Recovery dynamics
    random <- runif(length(infected_nodes),0,1)
    chance <- as.integer(random<=gamma)
    state[infected_nodes]<-state[infected_nodes]+chance
  }
  Istable[t]<-recovered[length(recovered)]
}

I_graph<-Istable
output<-data.frame(lambdas,Istable)

ggplot(data = output,aes(x=lambdas,y=Istable))+
  geom_line(linewidth=2)+
  theme_classic()+
  labs(x='Lambda values',y='R')
```



Markovian approach

Now let's try to model the same SIS model using the theoretical equations and compare them. For this, we will define two functions: $P_i(t)$ and $q_i(t)$. We have that P_i is defined as:

$$P_i(t+1) = P_i(t) * (1-\mu) + (1 - P_i(t)) * q_i(t)$$

defined as the probability of infection of node i for each time step t. On the other hand, the function q_i(t) is defined as:

$$q_i(t) = 1 - \prod_{j=1}^{n} [1 - \lambda A_{ij} P_j(t)]$$

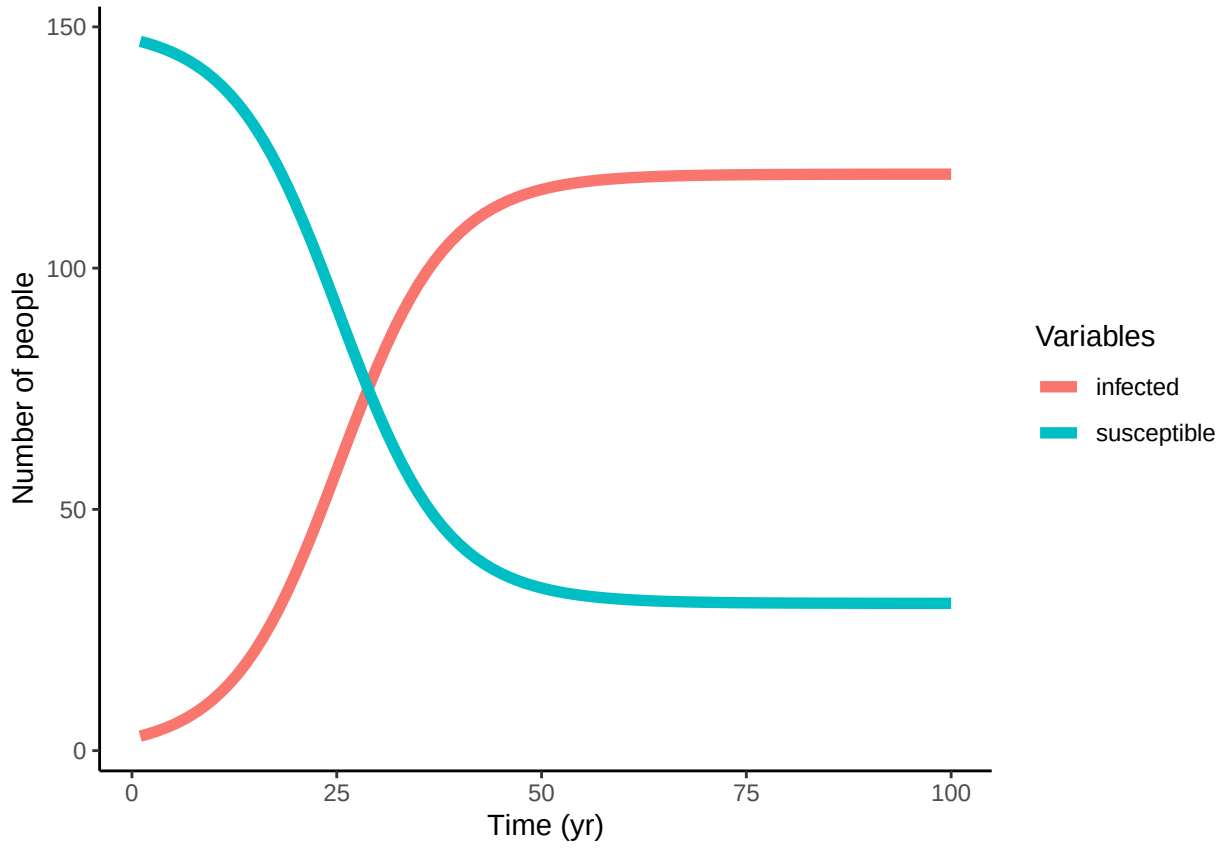
For the initial conditions of P_i(t), we equate all the values of the vector to the number of initial infected nodes divided by the total number of nodes. Let's start implementing these formulas for the SIS model

```
Tsim <- 100
lambda <- 0.02 #Contagion rate
mu <- 0.035 #Recovery rate

#Lists to store the values for each t from the infected and susceptible
```

```
susceptible <- rep(NA,Tsim)
infected <- rep(NA,Tsim)
ids_nodes <-1:150

#P_i(t) and q_i(t)
q_i <- rep(0,150)

#Initial conditions for P_i(0)
P_i <- rep(3/150,150)

for(t in 1:Tsim){
  #Storing the infection percentages
  infected[t] <- sum(P_i)/150
  susceptible[t] <- 1-infected[t]
  for(i in 1:150){
    #Updating q_i(t)
    neighbours <- ids_nodes[A1[i,]==1]
    if(length(neighbours)>0){
      q_i[i]<-1
      for(j in 1:length(neighbours)){
        q_i[i]<-q_i[i]*(1-lambda*P_i[j])
      }
      q_i[i]<-1-q_i[i]
    }
  }
  #Updating P_i(t)
  P_i <- P_i*(1-mu)+(1-P_i)*q_i
}
rm(P_i,q_i,mu,lambda,i,ids_nodes,t,Tsim,neighbours,j)
```

Let's see the behaviour of the curves:

```
output <-data.frame(time = 1:length(susceptible),susceptible = susceptible*150,infected = infected*150)
q<-output %>%
  gather(variable,value,-time) %>%
  ggplot(aes(x=time,y=value,color=variable))+
  geom_line(linewidth=2)+
  theme_classic()+
  labs(x='Time (yr)',y='Number of people',color = "Variables")
q
```

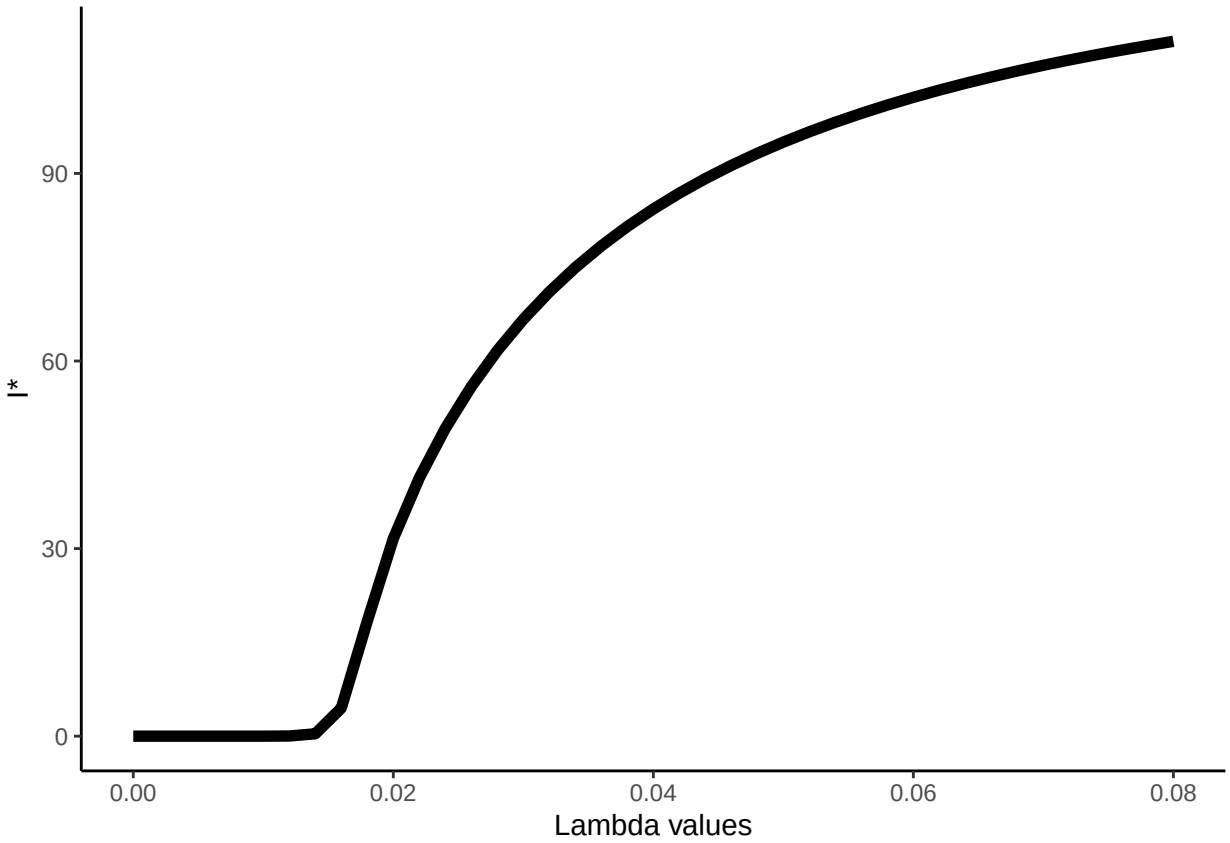


Just as we did above, we are going to plot how the number of infected individuals changes when the system stabilizes under different variations of lambda.

```
lambdas<-seq(from=0,to=0.08,by =0.002)
mus<-rep(0.15,length(lambdas))
Istable <- rep(NA,length(lambdas))

for(t in 1:length(lambdas)){
  Tsim <- 150
  lambda <- lambdas[t] #contagion rate
  mu <- mus[t] #recovery rate

  #Lists to store the values for each t from the infected and susceptible
  susceptible <- rep(NA,Tsim)
  infected <- rep(NA,Tsim)
  ids_nodes <-1:150

  #P_i(t) and q_i(t)
  q_i <- rep(0,150)

  #Initial conditions for P_i(0)
  P_i <- rep(3/150,150)

  for(j in 1:Tsim){
  #Storing the infection percentages
  infected[j] <- sum(P_i)/150
```

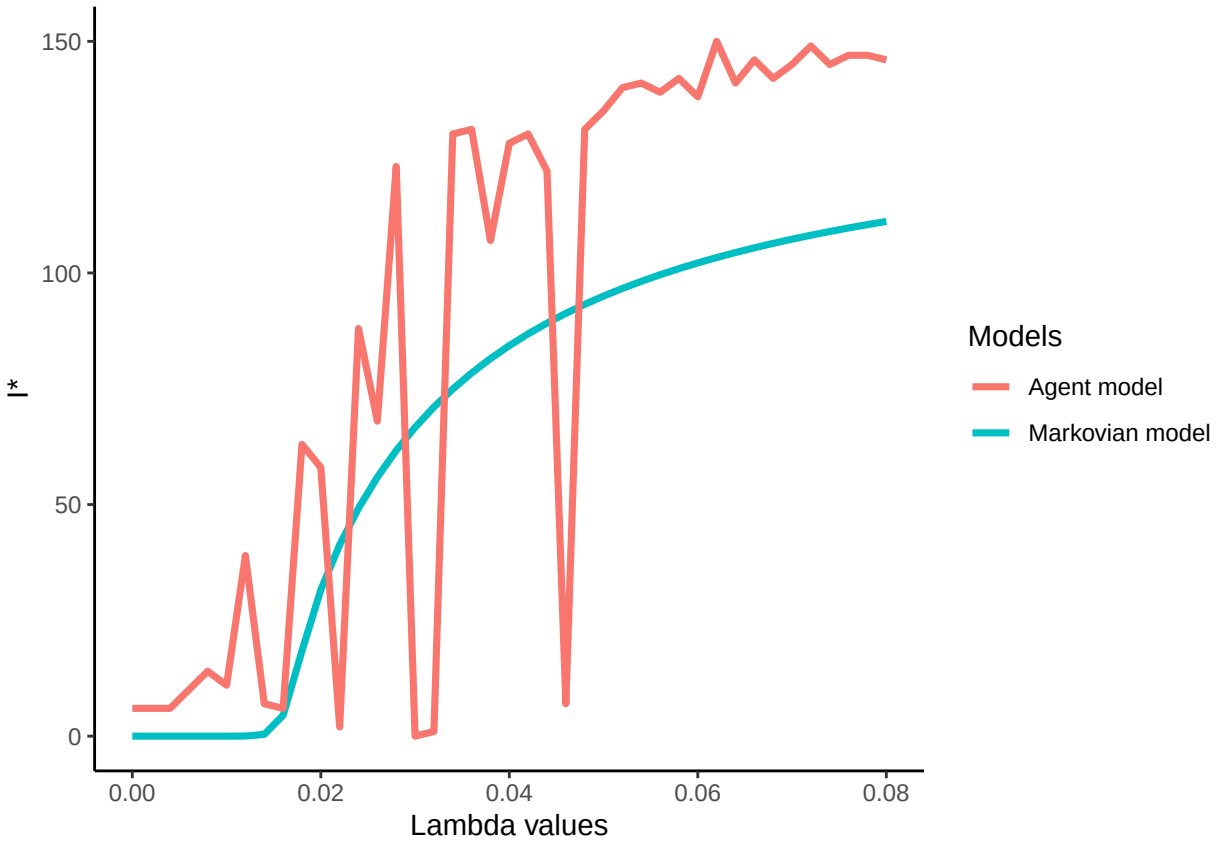
```
  susceptible[j] <- 1-infected[j]

  for(i in 1:150){
    #Updating q_i(t)
    neighbours <- ids_nodes[A1[i,]==1]
    if(length(neighbours)>0){
      q_i[i]<-1
      for(j in 1:length(neighbours)){
        q_i[i]<-q_i[i]*(1-lambda*P_i[j])
      }
      q_i[i]<-1-q_i[i]
    }
  }
  #Updating P_i(t)
  P_i <- P_i*(1-mu)+(1-P_i)*q_i
  }
  Istable[t]<-infected[length(infected)]*150
}

output<-data.frame(lambdas,Istable)

ggplot(data = output,aes(x=lambdas,y=Istable))+
geom_line(size=2)+
theme_classic()+
labs(x='Lambda values',y='I*')
```

```
## Warning: Using ‘size‘ aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use ‘linewidth‘ instead.
## This warning is displayed once every 8 hours.
## Call ‘lifecycle::last_lifecycle_warnings()‘ to see where this warning was
## generated.
```



What differences and similarities could be observed in the behaviour of both approaches? Let's compare the graphs of both stationary infected values to see their similarities (keeping in mind that they start from the same initial conditions).

```
output$Istable2<-I_graph
ggplot(data = output)+
geom_line(aes(x=lambdas,y=Istable,color = "Markovian model"),size=1.25)+
geom_line(aes(x=lambdas,y=Istable2,color = "Agent model"),size=1.25)+
theme_classic()+
labs(x='Lambda values',y='I*',color = "Models")
```



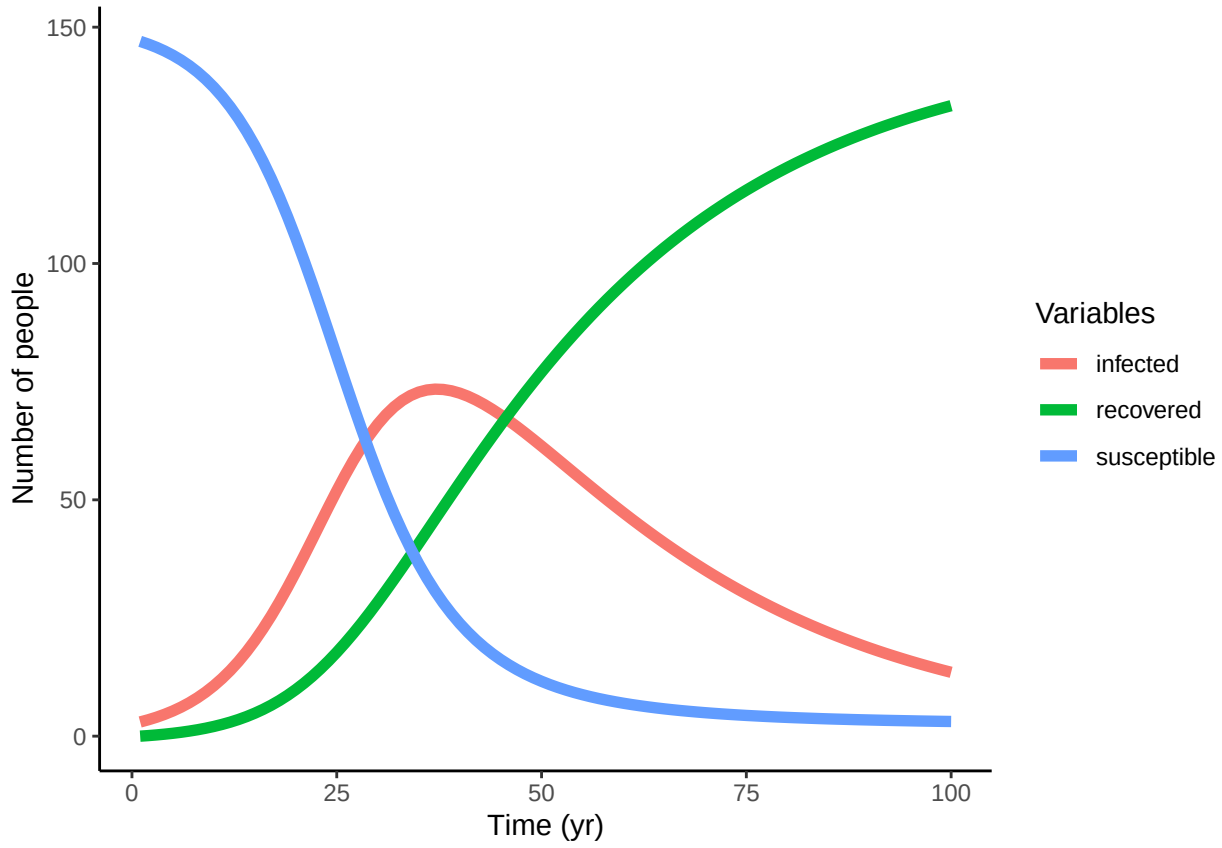
Exercise: just as in the agent based approach, modify the previous Markov algorithms to obtain the plots for an SIR model

```
Tsim <- 100
lambda <- 0.02 #Contagion rate
mu <- 0.035 #Recovery rate

#Lists to store the values for each t from the infected and susceptible
susceptible <- rep(NA,Tsim)
infected <- rep(NA,Tsim)
recovered <- rep(NA,Tsim)

ids_nodes <-1:150

#P_i(t) and q_i(t)
q_i <- rep(0,150)

r_i  <- rep(0,150)

#Initial conditions for P_i(0)
P_i <- rep(3/150,150)

for(t in 1:Tsim){
  #Storing the infection percentages
  infected[t] <- sum(P_i)/150
  recovered[t] <- sum(r_i)/150
```

```
  susceptible[t] <- 1-infected[t]-recovered[t]

  for(i in 1:150){
    #Updating q_i(t)
    neighbours <- ids_nodes[A1[i,]==1]
    if(length(neighbours)>0){
      q_i[i]<-1
      for(j in 1:length(neighbours)){
        q_i[i]<-q_i[i]*(1-lambda*P_i[j])
      }
      q_i[i]<-1-q_i[i]
    }
  }
  #Updating P_i(t)
  P_i <- P_i*(1-mu)+(1-P_i-r_i)*q_i
  r_i <- r_i + P_i*mu
}

rm(P_i,q_i,mu,lambda,i,ids_nodes,t,Tsim,neighbours,j)
```

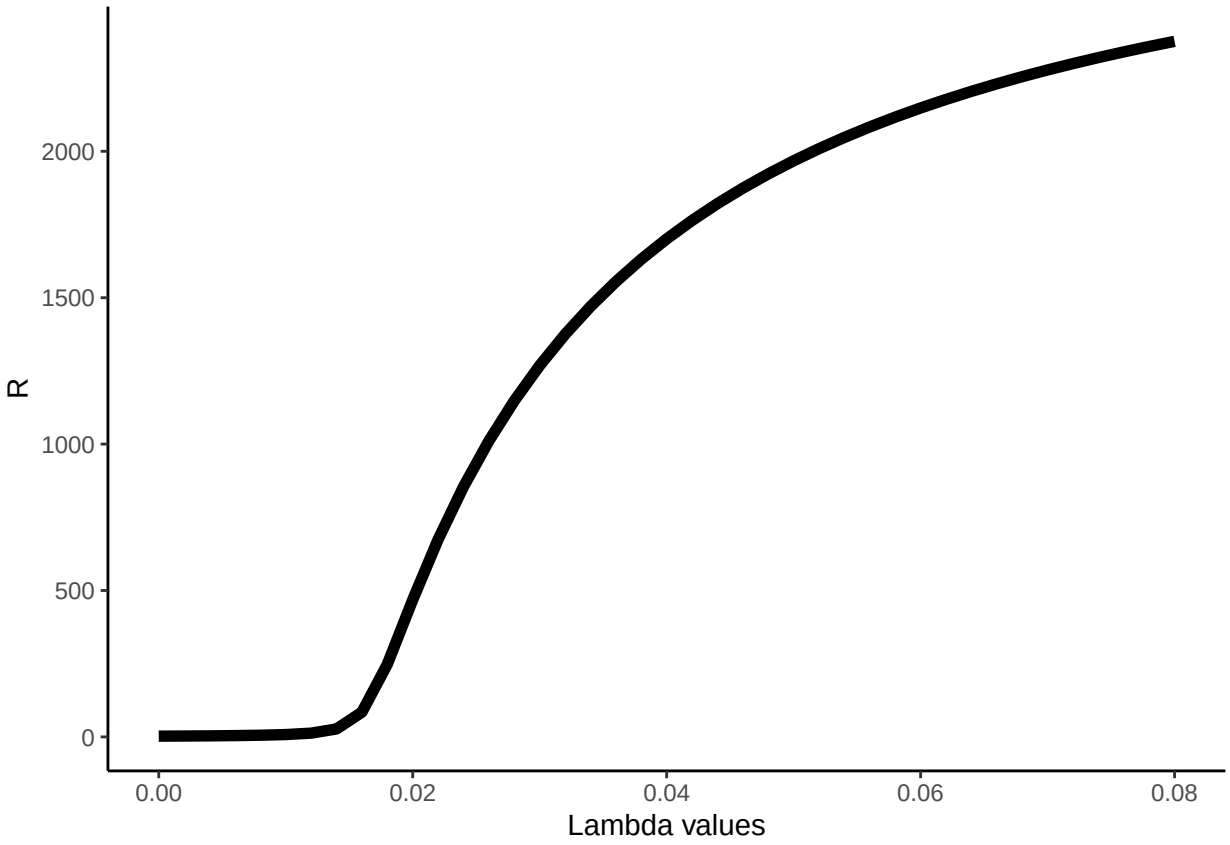
Let's see the behaviour of the curves:

```
output <-data.frame(time = 1:length(susceptible),susceptible = susceptible*150,infected = infected*150,
q<-output %>%
  gather(variable,value,-time) %>%
  ggplot(aes(x=time,y=value,color=variable))+
  geom_line(linewidth=2)+
  theme_classic()+
  labs(x='Time (yr)',y='Number of people',color = "Variables")
q
```




```
lambdas<-seq(from=0,to=0.08,by =0.002)
mus<-rep(0.15,length(lambdas))
Istable <- rep(NA,length(lambdas))

for(t in 1:length(lambdas)){
  Tsim <- 150
  lambda <- lambdas[t] #contagion rate
  mu <- mus[t] #recovery rate

  #Lists to store the values for each t from the infected and susceptible
  susceptible <- rep(NA,Tsim)
  infected <- rep(NA,Tsim)
  recovered <- rep(NA,Tsim)
  ids_nodes <-1:150

  #P_i(t) and q_i(t)
  q_i <- rep(0,150)
  r_i  <- rep(0,150)

  #Initial conditions for P_i(0)
  P_i <- rep(3/150,150)

  for(j in 1:Tsim){
  #Storing the infection percentages
  infected[j] <- sum(P_i)/150
  recovered[j] <- sum(r_i)/150
```

```
  susceptible[j] <- 1-infected[j]-recovered[j]

  for(i in 1:150){
    #Updating q_i(t)
    neighbours <- ids_nodes[A1[i,]==1]
    if(length(neighbours)>0){
      q_i[i]<-1
      for(j in 1:length(neighbours)){
        q_i[i]<-q_i[i]*(1-lambda*P_i[j])
      }
      q_i[i]<-1-q_i[i]
    }
  }
  #Updating P_i(t)
  P_i <- P_i*(1-mu)+(1-P_i)*q_i
  r_i <- r_i+P_i*mu
  }
  Istable[t]<-recovered[length(recovered)]*150
}

output<-data.frame(lambdas,Istable)

ggplot(data = output,aes(x=lambdas,y=Istable))+
geom_line(size=2)+
theme_classic()+
labs(x='Lambda values',y='R')
```


2000
1500
1000
500
0
R
0.00
0.02
0.04
0.06
0.08
Lambda values