Contagio en redes y modelos de Markov

En este notebook podrá repasar la generación de diferentes redes aleatorias y practicará la implementación de modelos de Markov que le permitan simular contagios que funcionen con las matrices de adyacencia que generemos

Instalando y cargando librerías

En caso que no tenga instaladas las siguientes librerías, corra el siguiente bloque de código para instalarlas.

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

Cargue las librerías que vamos a utilizar

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

Generando redes aleatorias

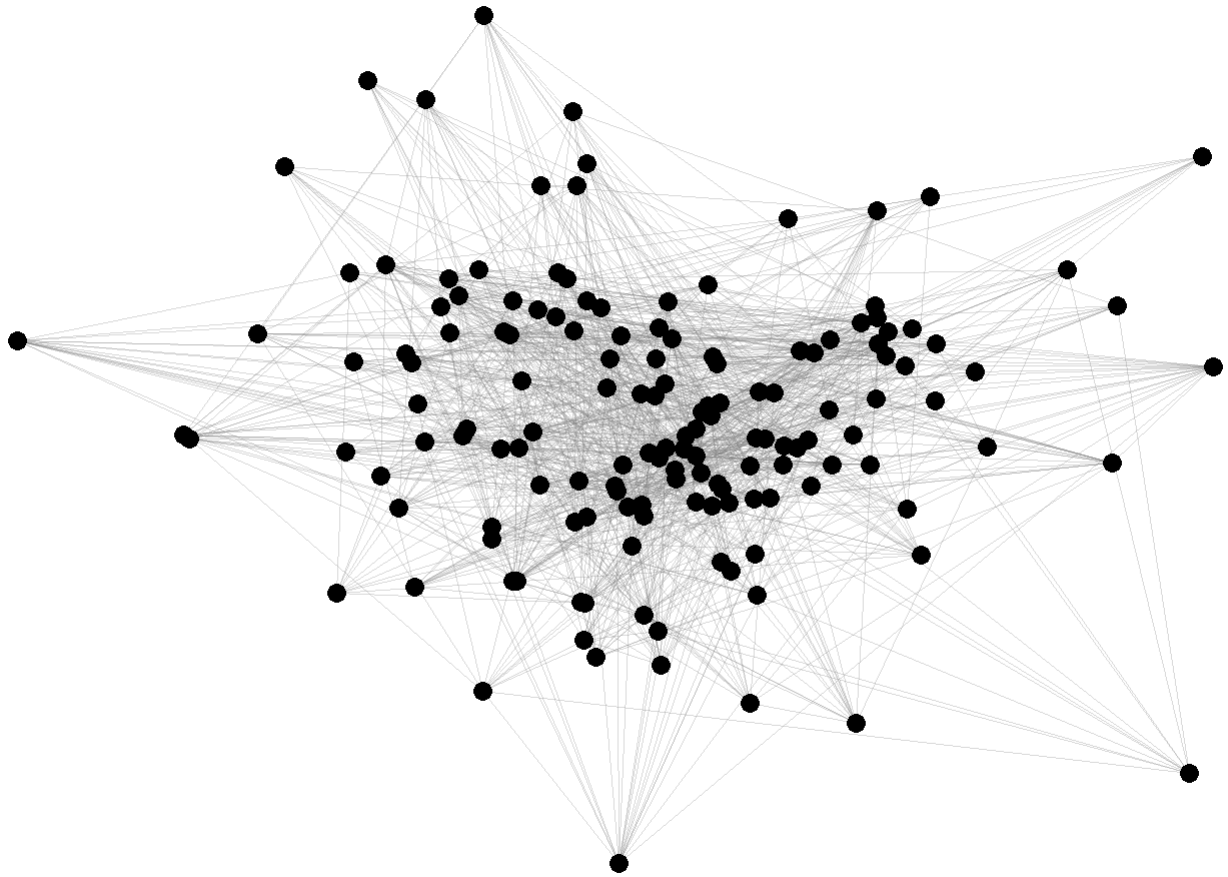
Vamos a empezar con las redes más comunes, empezando por la red de Erdos-Renyi. Para esto vamos a utilizar la librería **igraph** con su función **erdos.renyi.game**. Esta función es la siguiente:

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

Recibe diferentes parámetros, pero para este tutorial nos van a interesar solamente *n* que corresponde el número de nodos en la red, *p* que es la probabilidad de que se cree un arco entre dos nodos, y *directed* para configurar que la red sea dirigida o no dirigida.

Veamos cómo se crea una red y grafiquémosla:

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



Ahora vamos a querer extraer la matriz de adyacencia de esta red que acabamos de crear. Para eso vamos a utilizar la función **as_adjacency_matrix**, la cual recibe como parámetro la red de la que se va a extraer la matriz.

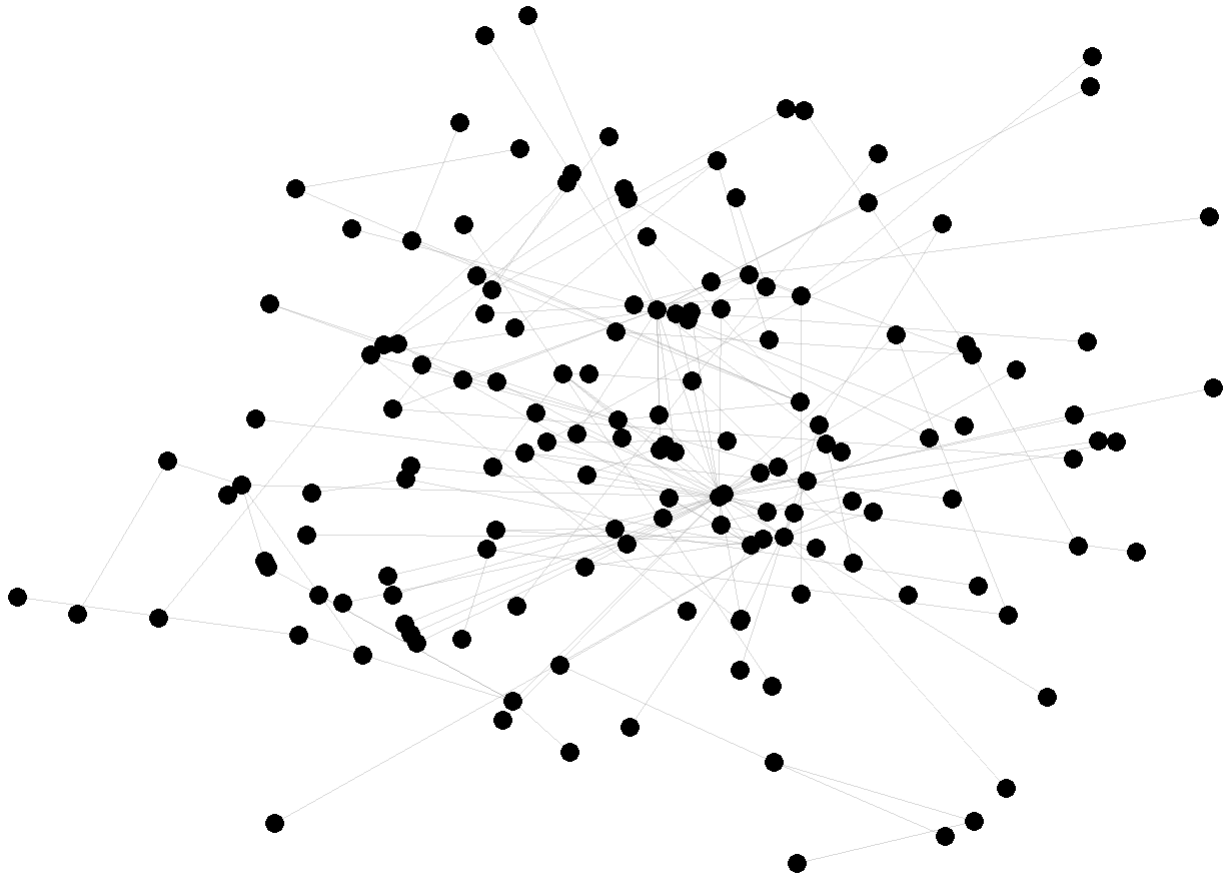
```
A2 <- as.matrix(as_adjacency_matrix(g1))
```

```
## as(<dsCMatrix>, "dgCMatrix") is deprecated since Matrix 1.5-0; do as(., "generalMatrix") inst
ead
```

Ahora, creemos una red libre de escala usando el modelo de Barabasi-Albert. La función que vamos a utilizar es **sample_pa** que tiene la siguiente definición: 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) Esta función recibe diferentes parámetros, pero para este tutorial nos van a interesar solamente *n* que corresponde el número de nodos en la red, *power* que es el poder del *preferential attachment* que por defecto está en 1 (*linear preferential attachment*), y *directed* para configurar que la red sea dirigida o no dirigida.

Veamos cómo se crea una red y grafiquémosla:

```
g2 <- sample_pa(n = 150)
ggnet2(g2,mode = "spring", vjust = -1, size = 3, color = 1,edge.alpha = 0.3,edge.size = 0.1)
```



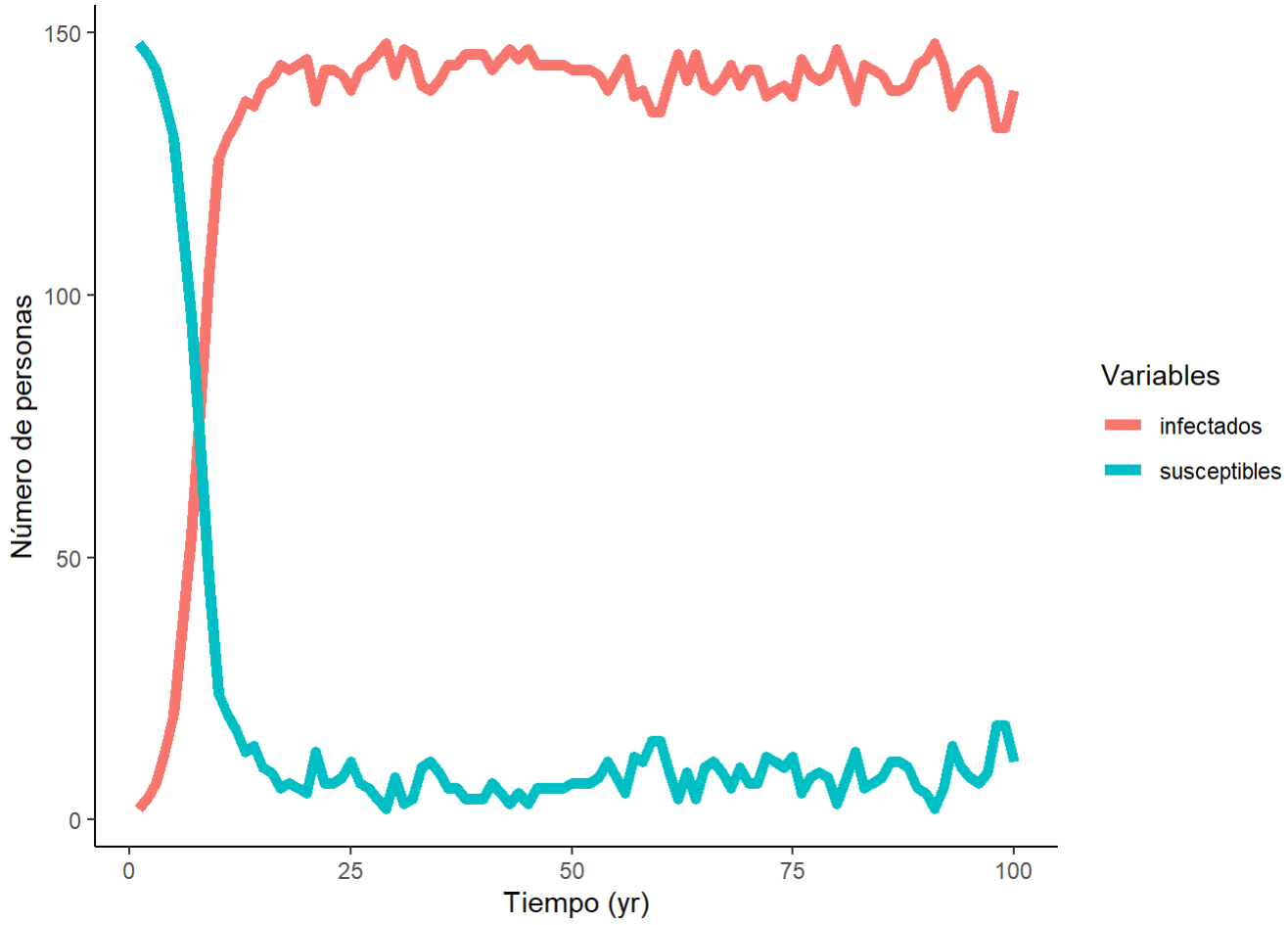
Contagio tipo cascada

Para el contagio lo que haremos es hacer un sorteo del contagio con todos los vecinos de cada nodo, usando como referencia la matriz de adyacencia. Comencemos implementando el contagio para un modelo SIS:

```
Tsim <- 100
lambda <- 0.075 #tasa de contagio
gamma <- 0.035 #tasa de recuperación

#Listas para guardar los valores para cada t de los infectados y susceptibles
susceptibles <- rep(NA,Tsim)
infectados <- rep(NA,Tsim)
estado <- rbinom(150,1,5/150)
ids_nodos<-1:150

for(i in 1:Tsim){
  infectados[i]<-sum(estado==1) #Registramos los infectados
  susceptibles[i]<-sum(estado==0) #Registramos los susceptibles

  #Dinámica de infección
  nodos_infectados<-ids_nodos[estado==1] #Extraer los nodos infectados
  for(j in 1:length(nodos_infectados)){
    vecinos <- ids_nodos[A2[nodos_infectados[j],]==1] #Extraer los vecinos de cada nodo infectad
o
    if(length(vecinos)>0){
      aleatorio <- runif(length(vecinos),0,1)
      chance <- as.integer(aleatorio<=lambda)
      estado[vecinos]<-ifelse(estado[vecinos]==0,estado[vecinos]+chance,estado[vecinos])
    }
  }
  #Dinámica de recuperación
  aleatorio <- runif(length(nodos_infectados),0,1)
  chance <- as.integer(aleatorio<=gamma)
  estado[nodos_infectados]<-estado[nodos_infectados]-chance
}
```

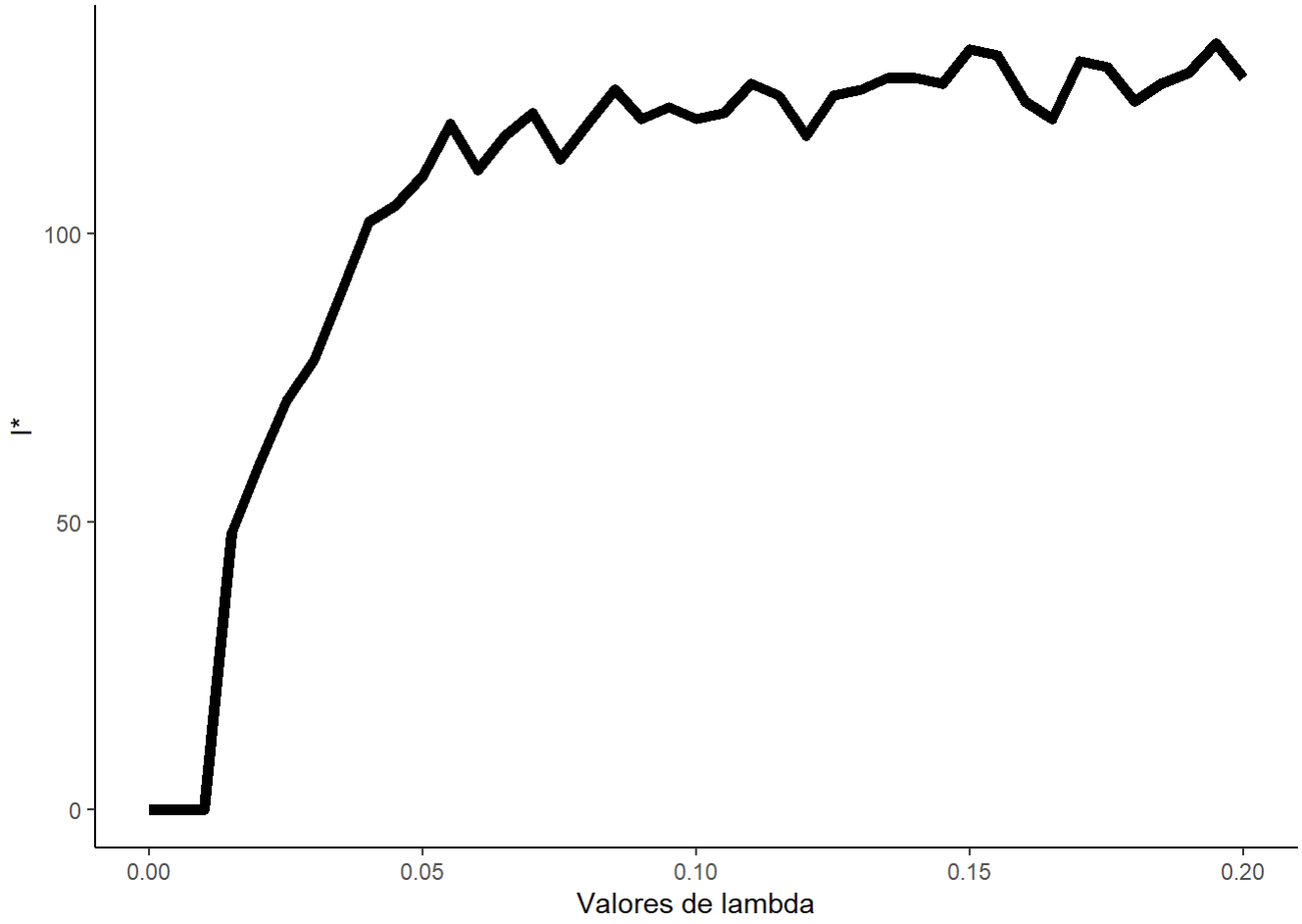
Veamos gráficamente el comportamiento de nuestras curvas:

```
salida <-data.frame(time = 1:length(susceptibles),susceptibles,infectados)

q<-salida %>%
  gather(variable,value,-time) %>%
  ggplot(aes(x=time,y=value,color=variable))+
  geom_line(size=2)+
  theme_classic()+
  labs(x='Tiempo (yr)',y='Número de personas',color = "Variables")
q
```



Al igual que como hicimos en el primer notebook, vamos a graficar cómo cambia el número de infectados cuando el sistema se estabiliza frente a diferentes variaciones de lambda.

```
lambdas<-seq(from=0,to=0.2,by =0.005)
gammas<-rep(0.15,length(lambdas))
Iestable <- rep(NA,length(lambdas))

for(t in 1:length(lambdas)){
  Tsim <- 150
  lambda <- lambdas[t] #tasa de contagio
  gamma <- gammas[t] #tasa de recuperación

  #Listas para guardar los valores para cada t de los infectados y susceptibles
  susceptibles <- rep(NA,Tsim)
  infectados <- rep(NA,Tsim)
  estado <- rbinom(150,1,4/150)
  ids_nodos<-1:150

  for(i in 1:Tsim){
    infectados[i]<-sum(estado==1) #Registramos los infectados
    susceptibles[i]<-sum(estado==0) #Registramos los susceptibles

    #Dinámica de infección
    nodos_infectados<-ids_nodos[estado==1] #Extraer los nodos infectados
    if(length(nodos_infectados)>0){
      for(j in 1:length(nodos_infectados)){
      vecinos <- ids_nodos[A2[nodos_infectados[j],]==1] #Extraer los vecinos de cada nodo infect
ado
        if(length(vecinos)>0){
          aleatorio <- runif(length(vecinos),0,1)
          chance <- as.integer(aleatorio<=lambda)
          estado[vecinos]<-ifelse(estado[vecinos]==0,estado[vecinos]+chance,estado[vecinos])
        }
      }
    }

    #Dinámica de recuperación
    aleatorio <- runif(length(nodos_infectados),0,1)
    chance <- as.integer(aleatorio<=gamma)
    estado[nodos_infectados]<-estado[nodos_infectados]-chance
  }
  Iestable[t]<-infectados[length(infectados)]
}

I_grafica<-Iestable
salida<-data.frame(lambdas,Iestable)

ggplot(data = salida,aes(x=lambdas,y=Iestable))+
  geom_line(size=2)+
  theme_classic()+
  labs(x='Valores de lambda',y='I*')
```



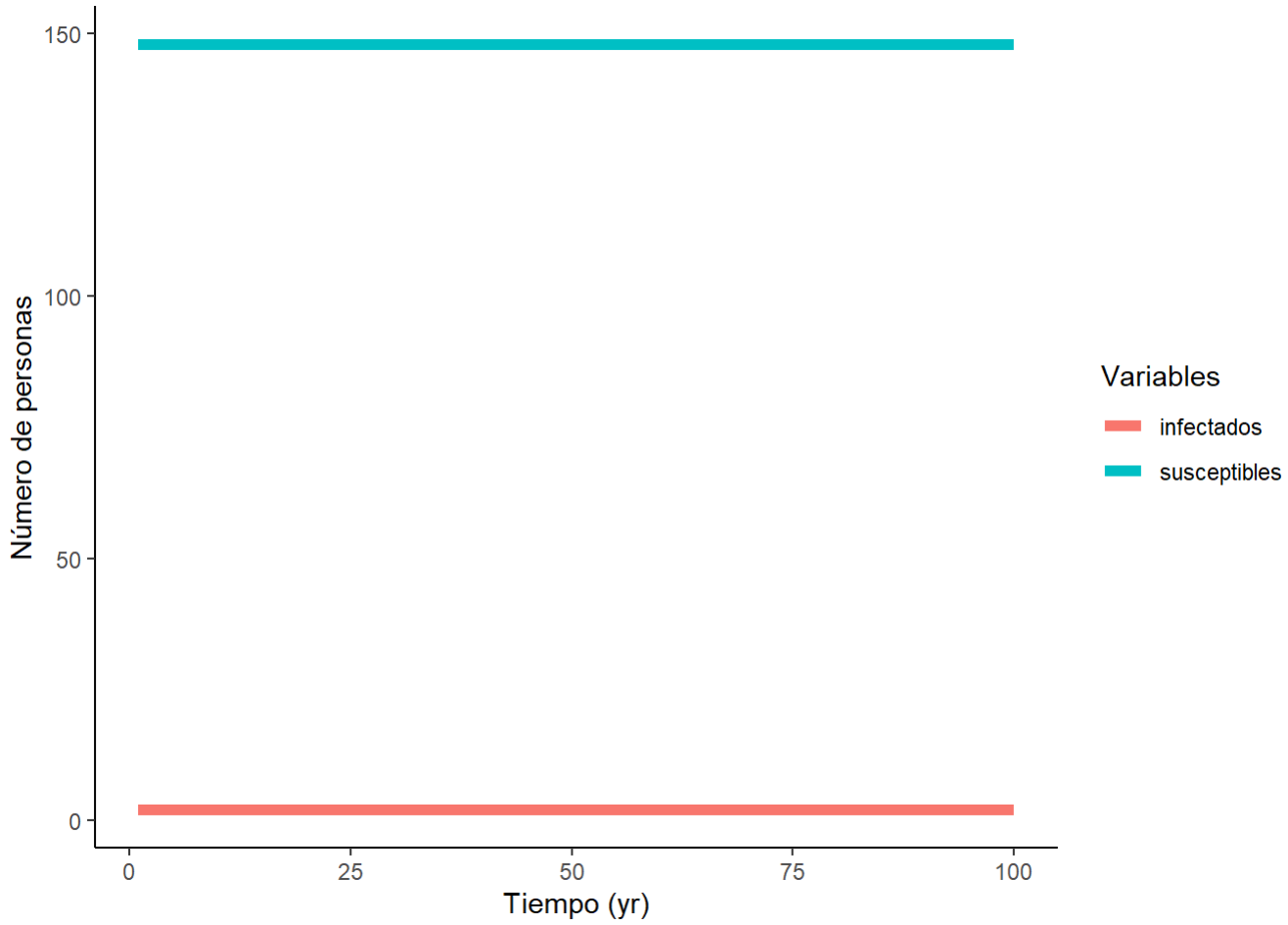
Contagio tipo umbral

Para el contagio lo que haremos es definir un umbral aleatorio, usando como referencia la matriz de adyacencia revisaremos el porcentaje de vecinos que se encuentran infectados y si supera el umbral, entonces el nodo se va a infectar. Comencemos implementando el contagio para un modelo SIS:

```
Tsim <- 100
lambda <- 0.075 #tasa de contagio
gamma <- 0.035 #tasa de recuperación

#Listas para guardar los valores para cada t de los infectados y susceptibles
susceptibles <- rep(NA,Tsim)
infectados <- rep(NA,Tsim)
estado <- rbinom(150,1,5/150)
umbral<- runif(150,0,1)
ids_nodos<-1:150

for(i in 1:Tsim){
  infectados[i]<-sum(estado==1) #Registramos los infectados
  susceptibles[i]<-sum(estado==0) #Registramos los susceptibles

  #Dinámica de infección
  nodos_infectados<-ids_nodos[estado==1] #Extraer los nodos infectados desde el inicio
  nodos_susceptibles<-ids_nodos[estado==0] #Extraer los nodos susceptibles
  for(j in 1:length(nodos_susceptibles)){
    vecinos <- ids_nodos[A2[nodos_susceptibles[j],]==1] #Extraer los vecinos de cada nodo infect
ado
    pinfectados <- sum(estado[vecinos])/length(vecinos) #Calcular el porcentaje de infectados
    if(pinfectados>umbral[nodos_susceptibles[j]]){
      estado[nodos_susceptibles[j]]<-1
    }

  }
  #Dinámica de recuperación
  if(length(nodos_infectados)>0){
    for(j in 1:length(nodos_infectados)){
      vecinos <- ids_nodos[A2[nodos_infectados[j],]==1] #Extraer los vecinos de cada nodo infect
ado
      pinfectados <- sum(estado[vecinos])/length(vecinos) #Calcular el porcentaje de infectados
      if(pinfectados<=umbral[nodos_infectados[j]]){
        estado[nodos_infectados[j]]<-0
      }
    }
  }

}
```

Veamos gráficamente el comportamiento de nuestras curvas:

```
salida <-data.frame(time = 1:length(susceptibles),susceptibles,infectados)

q<-salida %>%
  gather(variable,value,-time) %>%
  ggplot(aes(x=time,y=value,color=variable))+
  geom_line(size=2)+
  theme_classic()+
  labs(x='Tiempo (yr)',y='Número de personas',color = "Variables")
q
```



Prueba con diferentes valores para el umbral y se dará cuenta que en función de eso las curvas crecerán más o menos rápido.

Aproximación Markoviana

Ahora intentemos modelar el mismo SIS desde las ecuaciones teóricas y comparemos. Para esto vamos a definir dos funciones: *P_i(t)* y *q_i(t)*. Tenemos entonces que *P_i* se define como:

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

definido como la probabilidad de infección del nodo *i* para cada momento *t* de tiempo. Por otra parte, la función *q_i(t)* se define como:

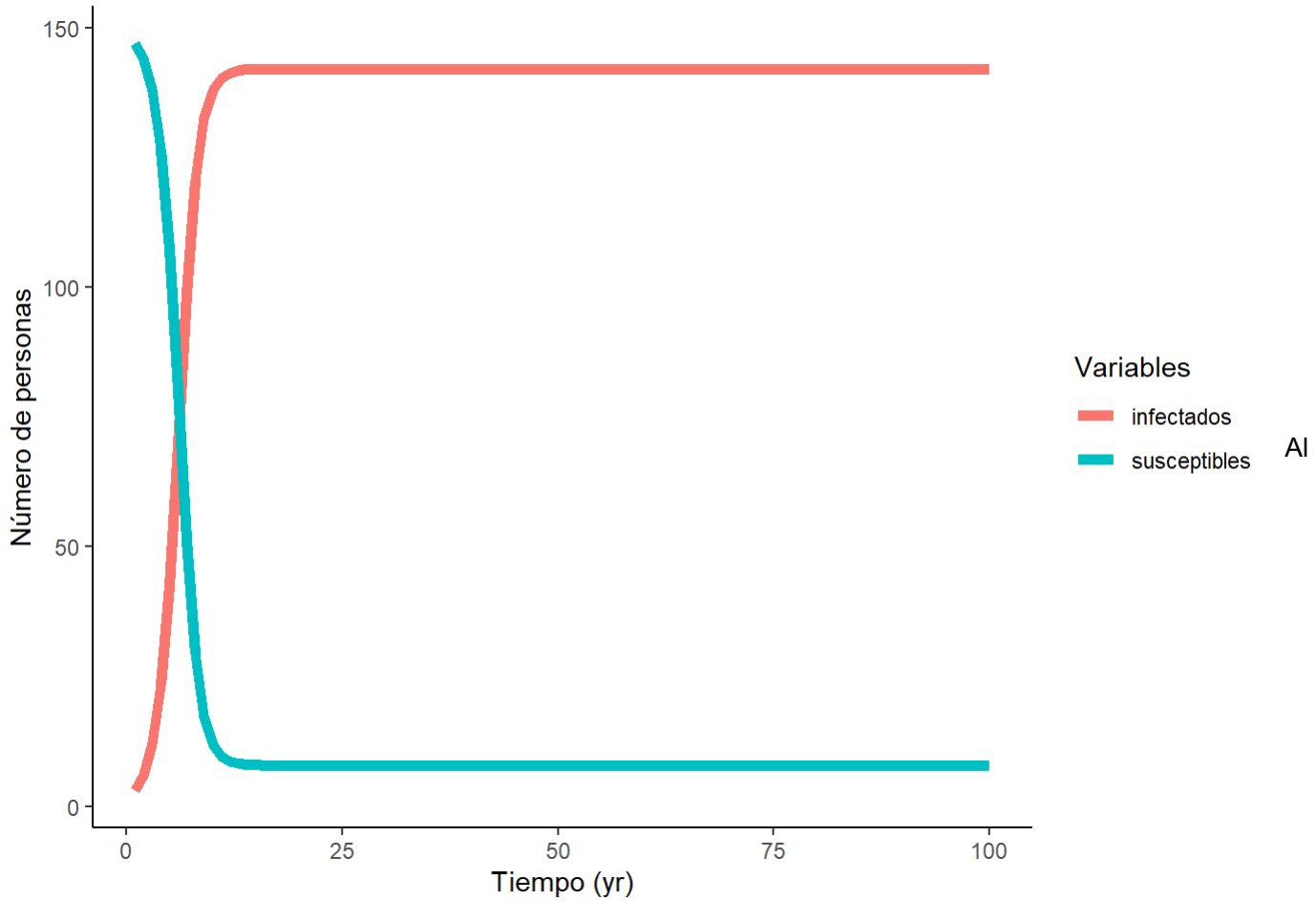
$$q_i(t) = 1 - \prod_{j=1}^{n} [1 - \lambda A_{ij} P_j(t)]$$

Para las condiciones iniciales de *P_i(t)* igualamos todos los valores del vector a a número de nodos infectados iniciales dividido el total de nodos. Comencemos implementando estas fórmulas para el modelo SIS:

```
Tsim <- 100
lambda <- 0.075 #tasa de contagio
mu <- 0.035 #tasa de recuperación

#Listas para guardar los valores para cada t de los infectados y susceptibles
susceptibles <- rep(NA,Tsim)
infectados <- rep(NA,Tsim)
ids_nodos <-1:150

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

#Condiciones iniciales para P_i(0)

P_i <- rep(3/150,150)

for(t in 1:Tsim){
  #Guardando los porcentajes de infección
  infectados[t] <- sum(P_i)/150
  susceptibles[t] <- 1-infectados[t]

  for(i in 1:150){
    #Actualizando q_i(t)
    vecinos <- ids_nodos[A2[i,]==1]
    if(length(vecinos)>0){
      q_i[i]<-1
      for(j in 1:length(vecinos)){
        q_i[i]<-q_i[i]*(1-lambda*P_i[j])
      }
      q_i[i]<-1-q_i[i]
    }

  }
  #Actualizando P_i(t)
  P_i <- P_i*(1-mu)+(1-P_i)*q_i
}
#rm(P_i,q_i,mu,lambda,i,ids_nodos,t,Tsim,vecinos,j)
```

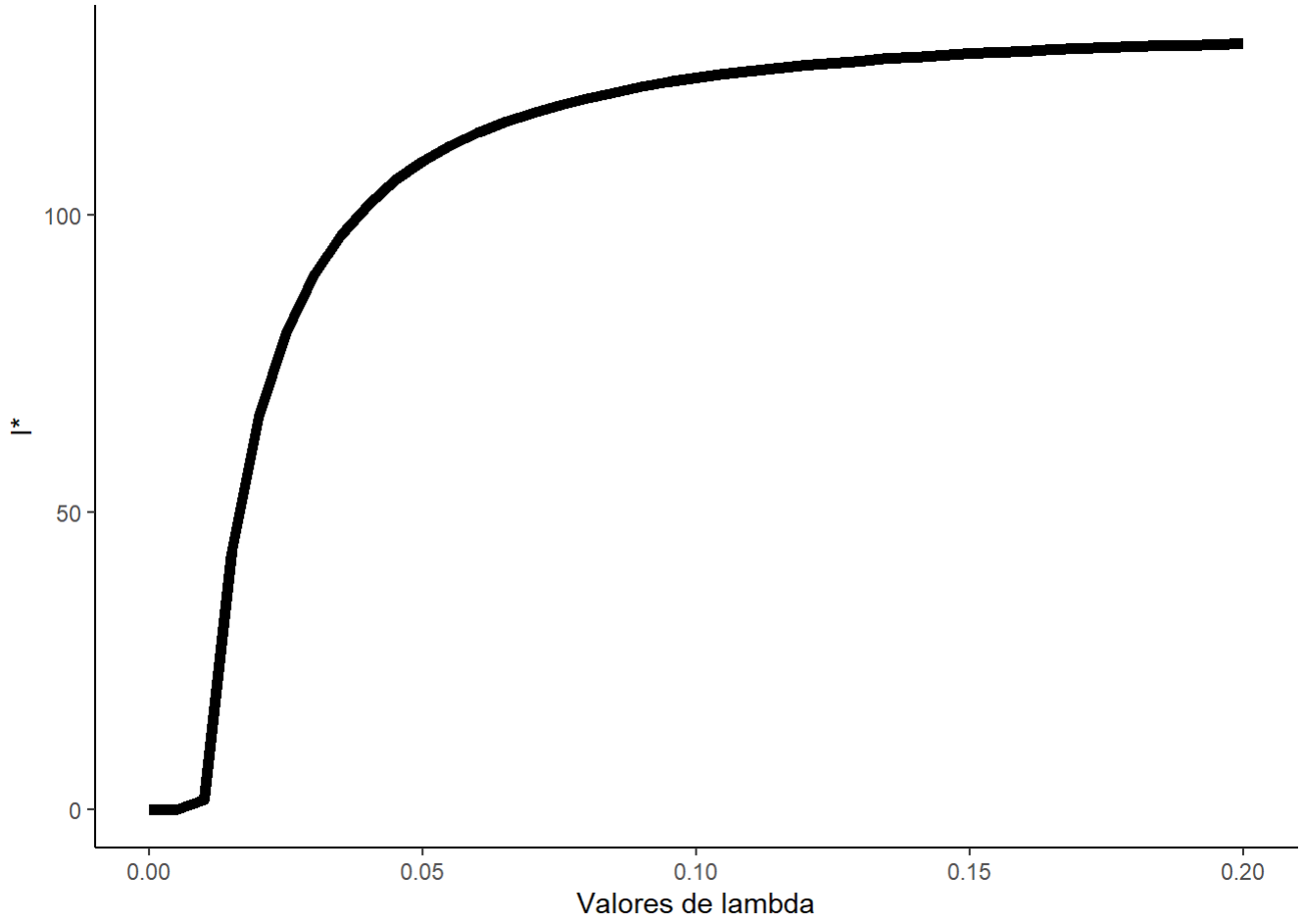
Veamos gráficamente el comportamiento de nuestras curvas:

```
salida <-data.frame(time = 1:length(susceptibles),susceptibles = susceptibles*150,infectados = i
nfectados*150)

q<-salida %>%
  gather(variable,value,-time) %>%
  ggplot(aes(x=time,y=value,color=variable))+
  geom_line(size=2)+
  theme_classic()+
  labs(x='Tiempo (yr)',y='Número de personas',color = "Variables")
q
```



igual que como hicimos arriba, vamos a graficar cómo cambia el número de infectados cuando el sistema se estabiliza frente a diferentes variaciones de lambda.

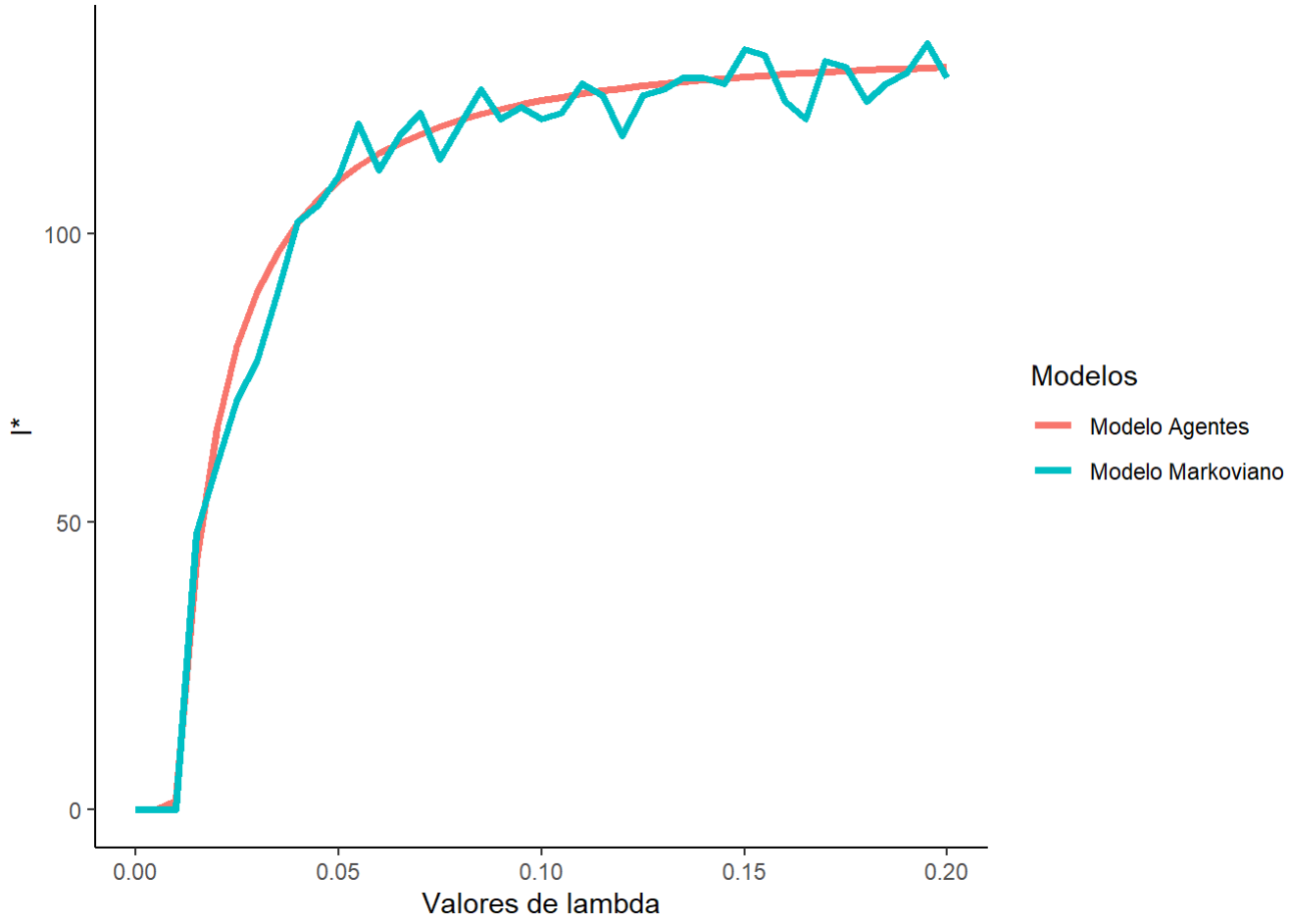
```
lambdas<-seq(from=0,to=0.2,by =0.005)
mus<-rep(0.15,length(lambdas))
Iestable <- rep(NA,length(lambdas))

for(t in 1:length(lambdas)){
  Tsim <- 150
  lambda <- lambdas[t] #tasa de contagio
  mu <- mus[t] #tasa de recuperación

  #Listas para guardar los valores para cada t de los infectados y susceptibles
  susceptibles <- rep(NA,Tsim)
  infectados <- rep(NA,Tsim)
  ids_nodos <-1:150

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

  #Condiciones iniciales para P_i(0)

  P_i <- rep(3/150,150)

  for(j in 1:Tsim){
    #Guardando los porcentajes de infección
    infectados[j] <- sum(P_i)/150
    susceptibles[j] <- 1-infectados[j]

    for(i in 1:150){
      #Actualizando q_i(t)
      vecinos <- ids_nodos[A2[i,]==1]
      if(length(vecinos)>0){
        q_i[i]<-1
        for(j in 1:length(vecinos)){
          q_i[i]<-q_i[i]*(1-lambda*P_i[j])
        }
        q_i[i]<-1-q_i[i]
      }

    }
    #Actualizando P_i(t)
    P_i <- P_i*(1-mu)+(1-P_i)*q_i
  }
  Iestable[t]<-infectados[length(infectados)]*150
}

salida<-data.frame(lambdas,Iestable)
ggplot(data = salida,aes(x=lambdas,y=Iestable))+
  geom_line(size=2)+
  theme_classic()+
  labs(x='Valores de lambda',y='I*')
```



Qué diferencias y semejanzas puede observar en los comportamientos de ambas aproximaciones?

Comparemos las gráficas de ambos valores de infectados estacionarios para ver sus semejanzas (teniendo presente que parten de las mismas condiciones iniciales).

```
salida$Iestable2<-I_grafica
ggplot(data = salida)+
  geom_line(aes(x=lambdas,y=Iestable,color = "Modelo Agentes"),size=1.25)+
  geom_line(aes(x=lambdas,y=Iestable2,color = "Modelo Markoviano"),size=1.25)+
  theme_classic()+
  labs(x='Valores de lambda',y='I*',color = "Modelos")
```


100
50
0
I*
0.00
0.05
0.10
0.15
0.20
Valores de lambda
Modelos
Modelo Agentes
Modelo Markoviano