R3 Model - Quick Start

Benjamin Ortiz Ulloa

1/28/2020

Getting Started

First make sure you set your **R** working directory is set to the

```
setwd('path/to/r3-model')
```

Then ensure you have all the main package dependencies we'll use

```
install.packages(c(
  'dplyr',
  'ggplot2',
  'igraph',
  'magrittr',
  'purrr',
  'tidyr'
))
```

Then make sure to import the relevant functions to your R environment

```
source('r3-model.R')
```

Initializing elements

We first need to initialize the actors/nodes we'll be using. The different types of nodes will represent **people** actors that can get addicted or recover, **rehabs** where an addicted person can recover from addiction, and **postRehab** where people go when they leave rehab and have a chance of relapsing.

```
set.seed(4321) #setting the seed ensures that we obtain the same results

testNodes <- initializeNodes(nPeople = 1000,
                             nRehab = 20,
                             nPostRehab = 2,
                             ratePostRehab = c(.2, .8))

#purrr::map is a function that allows you to iterate over a list with another fucntion
#head is a function that give the top 6 rows of any data frame
purrr::map(testNodes, head)
```

```
## $people
## # A tibble: 6 x 3
##   id       stable type
##   <chr>    <lgl>  <chr>
## 1 person_1 TRUE   person
## 2 person_2 FALSE  person
## 3 person_3 FALSE  person
## 4 person_4 FALSE  person
## 5 person_5 TRUE   person
```

```
## 6 person_6 TRUE   person
##
## $rehab
## # A tibble: 6 x 3
##   id        rate type
##   <chr>    <dbl> <chr>
## 1 rehab_1 0.335  rehab
## 2 rehab_2 0.909  rehab
## 3 rehab_3 0.412  rehab
## 4 rehab_4 0.0438 rehab
## 5 rehab_5 0.764  rehab
## 6 rehab_6 0.750  rehab
##
## $postRehab
## # A tibble: 2 x 3
##   id           rate type
##   <chr>       <dbl> <chr>
## 1 postRehab_1   0.2 postRehab
## 2 postRehab_2   0.8 postRehab
```

The rate of a **rehab** corresponds to the probability that an addict will recover while in its facilities. The rate of a **postRehab**, however, corresponds to the probability that an addict will relapse and return to a **rehab**.

If the *ratePostRehab* parameter is not set, then the **relapse rate** of the post rehab center will be set to a random number between 0 and 1.

We now need to create the edge list, which connects a *person* to either a *rehab facility* or a **post rehab facility**.

```
set.seed(4321)
testEdges <- initializeEdges(testNodes)

head(testEdges)
```

```
##       from          to rate    toType fromStabilized
## 1 person_1 postRehab_1  0.2 postRehab           TRUE
## 2 person_2 postRehab_2  0.8 postRehab          FALSE
## 3 person_3 postRehab_1  0.2 postRehab          FALSE
## 4 person_4 postRehab_1  0.2 postRehab          FALSE
## 5 person_5 postRehab_2  0.8 postRehab           TRUE
## 6 person_6 postRehab_2  0.8 postRehab           TRUE
```

fromStabilized refers to whether the person (*the source/from node*) is actively using drugs or not. **toType** refers to the the type of facility (*the target/to node*) the person is residing in. The **rate** refers to the probability of recovery if the **toType** is *rehab* or it refers to the probability of relapse if the **toType** is *postRehab*.

Simulations

noIntervention

The null model will be called using the `noIntervention` function. This essentially sees the system behave with no major changes to the facilities.

```
set.seed(4321)
nullModel <- noIntervention(edgeList = testEdges,
```

```
                            nodeList = testNodes,
                            nTime = 100)
head(nullModel)
```

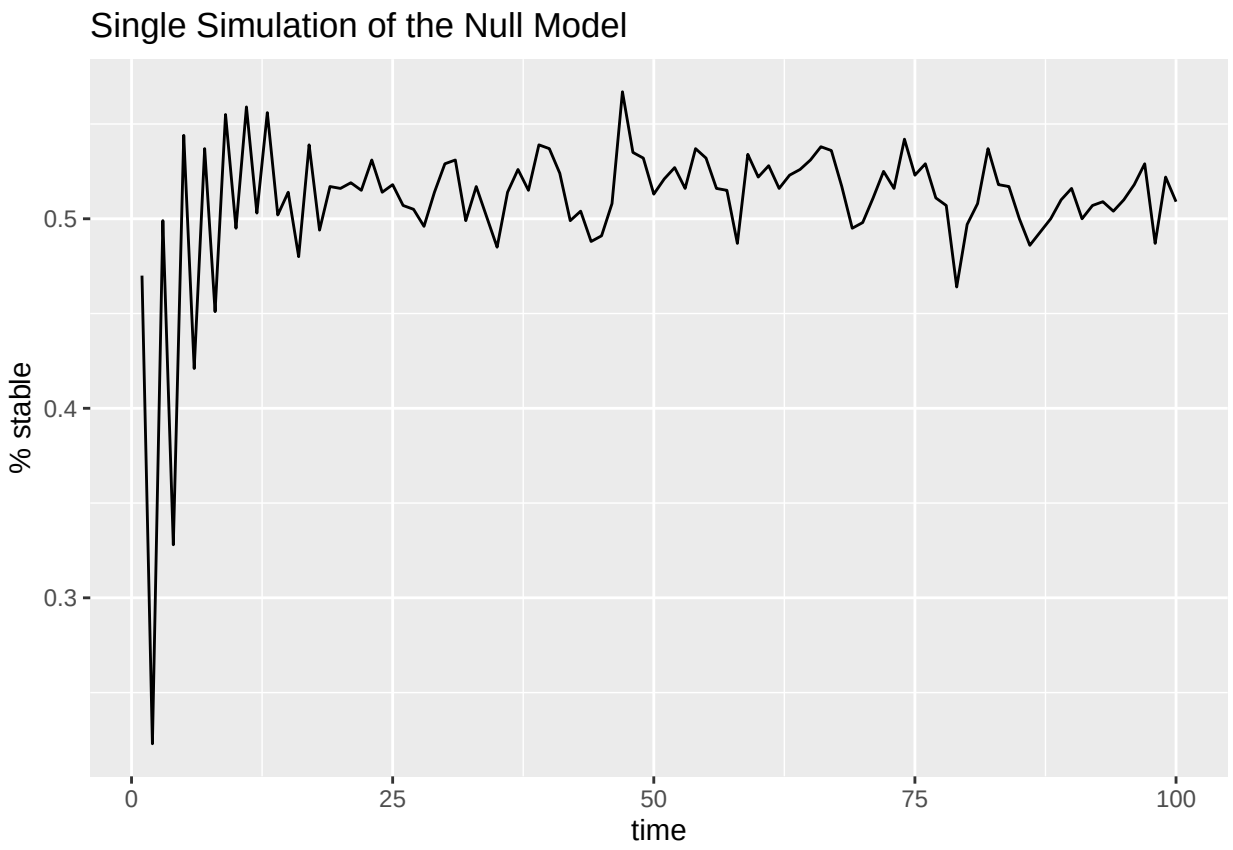
```
## # A tibble: 6 x 7
##    time nStable nTotal pStable pPostRecovery pRehab g
##   <int>   <int>  <int>   <dbl>         <dbl>  <dbl> <list>
## 1     1     470   1000   0.47          1      0     <S3: igraph>
## 2     2     223   1000   0.223         0.223  0.777 <S3: igraph>
## 3     3     499   1000   0.499         0.9    0.1   <S3: igraph>
## 4     4     328   1000   0.328         0.38   0.62  <S3: igraph>
## 5     5     544   1000   0.544         0.843  0.157 <S3: igraph>
## 6     6     421   1000   0.421         0.502  0.498 <S3: igraph>
```

What we get is data frame with each row representing a new turn in the simulation. The parameter **nTime** sets the amount of turns taken in a simulation.

nStable represents the number of stablized actors in the system. **pPostRecovery** and **pRehab** is the percentage of actors in *postRecovery* and *rehab* facilities respectively. *g* is an igraph object that holds a graphical representation of the system during each turn.

This single simulation can be quickly viewed using **ggplot2**

```
library(ggplot2)
ggplot(data = nullModel) +
  geom_line(aes(x = time, y = pStable)) +
  labs(title = 'Single Simulation of the Null Model',
       y = '% stable')
```

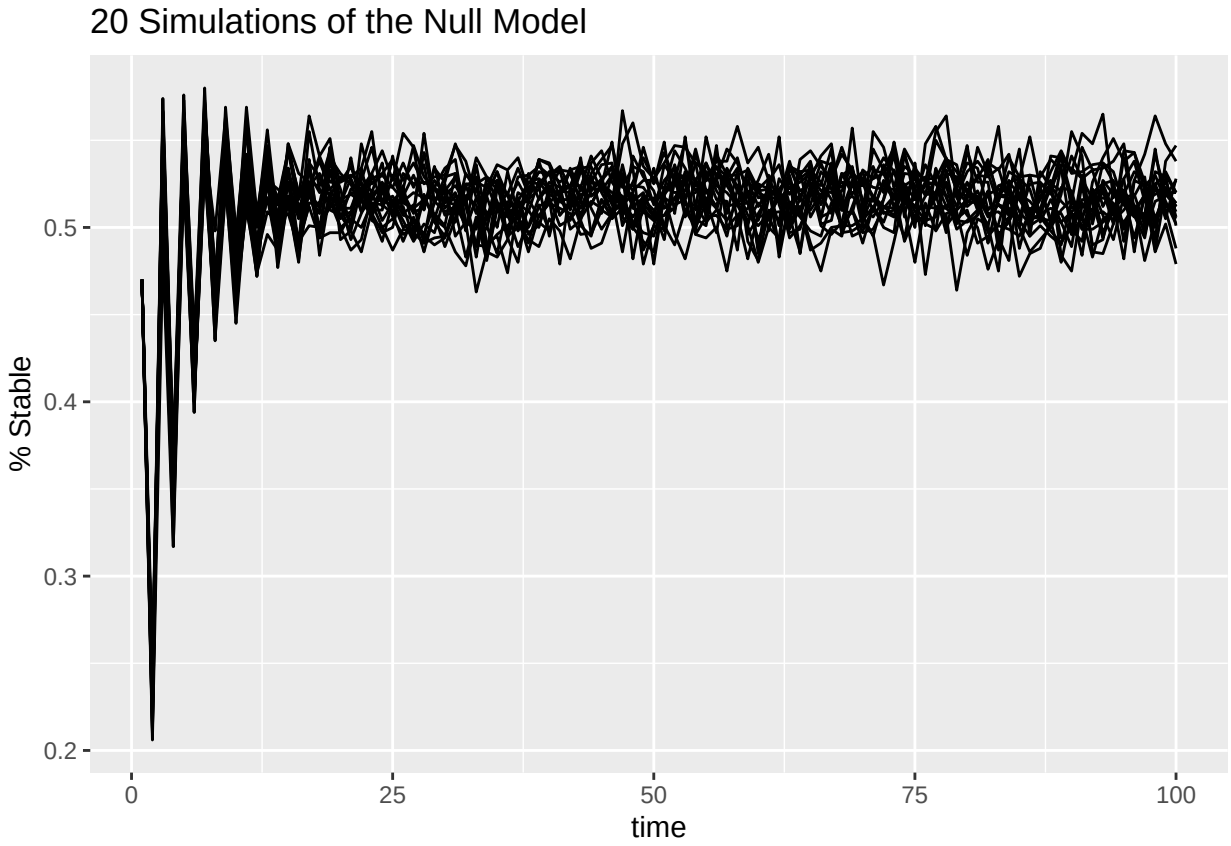


We can also run multiple iterations of the model.

```
#must wrap the model in a function
nullModelFunc <- function(){
  noIntervention(edgeList = testEdges,
                 nodeList = testNodes,
                 nTime = 100)
}

set.seed(4321)
nullModelIterations <- iterateSim(nIter = 20,
                                  fun = nullModelFunc)

head(nullModelIterations)
```

```
## # A tibble: 6 x 8
##   iteration  time nStable nTotal pStable pPostRecovery pRehab g
##       <int> <int>   <int>  <int>   <dbl>         <dbl>  <dbl> <list>
## 1         1     1     470   1000   0.47          1      0     <S3: igraph>
## 2         1     2     223   1000   0.223         0.223  0.777 <S3: igraph>
## 3         1     3     499   1000   0.499         0.9    0.1   <S3: igraph>
## 4         1     4     328   1000   0.328         0.38   0.62  <S3: igraph>
## 5         1     5     544   1000   0.544         0.843  0.157 <S3: igraph>
## 6         1     6     421   1000   0.421         0.502  0.498 <S3: igraph>
```

```
ggplot(nullModelIterations) +
  geom_line(aes(x = time, y = pStable, group = iteration)) +
  labs(title = '20 Simulations of the Null Model',
      y = '% Stable')
```

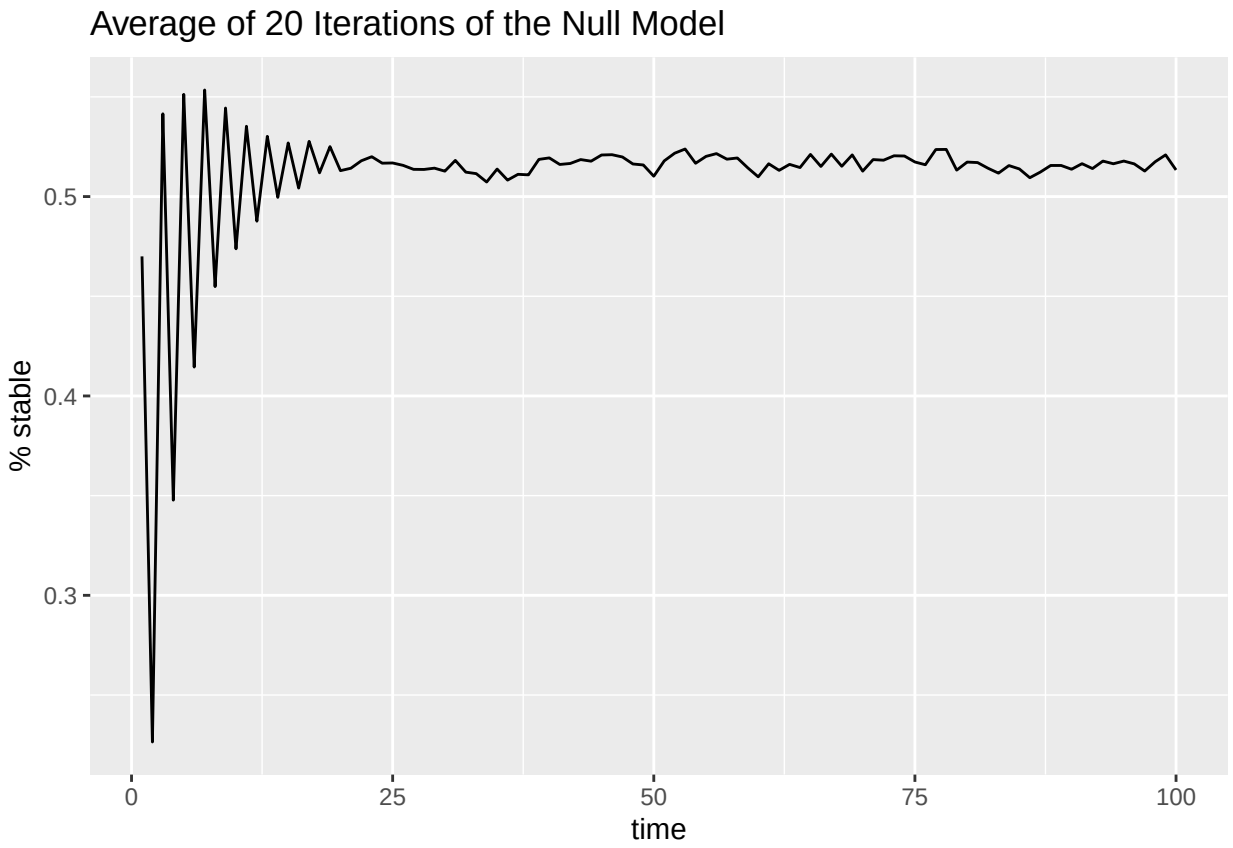


We can clean up the visualization by getting the mean *% stable population* of all models at any give time **t**

```
library(magrittr)

nullModelIterationsAve <- nullModelIterations %>%
  dplyr::group_by(time) %>%
  dplyr::summarise(pStable = mean(pStable))

head(nullModelIterationsAve)
```

```
## # A tibble: 6 x 2
##    time pStable
##   <int>   <dbl>
## 1     1   0.47
## 2     2   0.226
## 3     3   0.542
## 4     4   0.348
## 5     5   0.551
## 6     6   0.414
```

```
ggplot(data = nullModelIterationsAve) +
  geom_line(aes(x = time, y = pStable)) +
  labs(title = 'Average of 20 Iterations of the Null Model',
       y = '% stable')
```

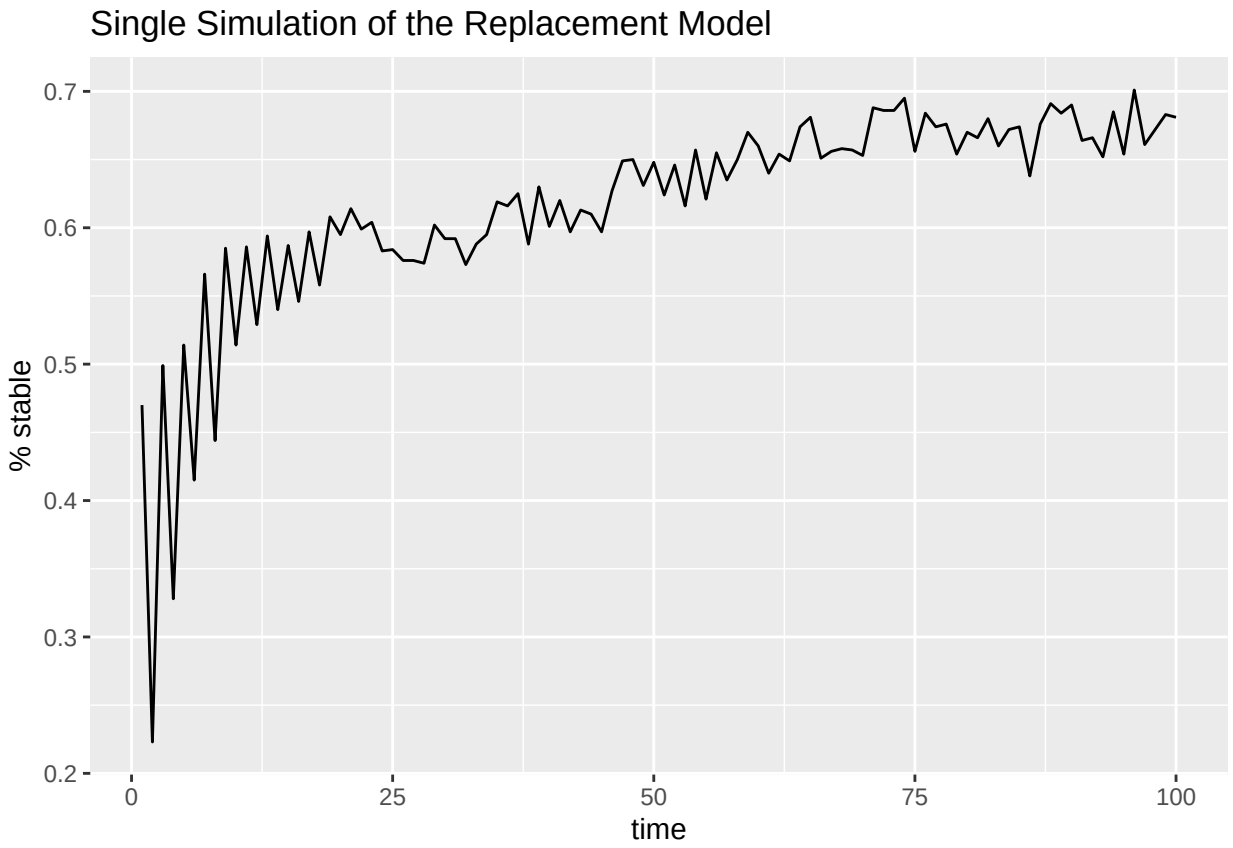


replacementIntervention

```
set.seed(4321)
replacementModel <- replacementIntervention(edgeList = testEdges,
                                            nodeList = testNodes,
                                            nTime = 100,
                                            interventionFreq = 5)
```

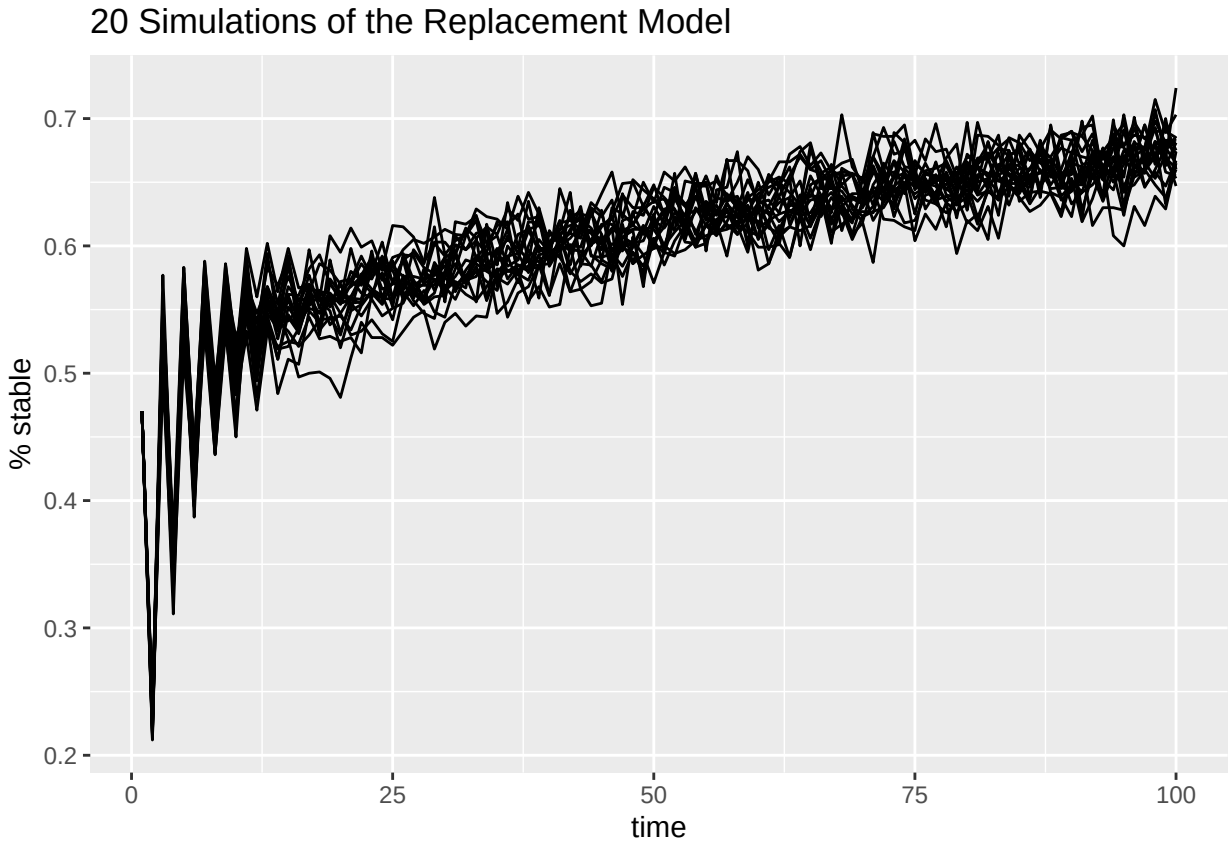
The parameter **interventionFreq** replaces a rehab facility ever n turns. The rehab facility that is replaced is the one with the worst recovery rate. We replace it with one that has a recovery rate between the worst and best rates of the rehabs.

```
ggplot(data = replacementModel) +
  geom_line(aes(x = time, y = pStable)) +
  labs(title = 'Single Simulation of the Replacement Model',
       y = '% stable')
```

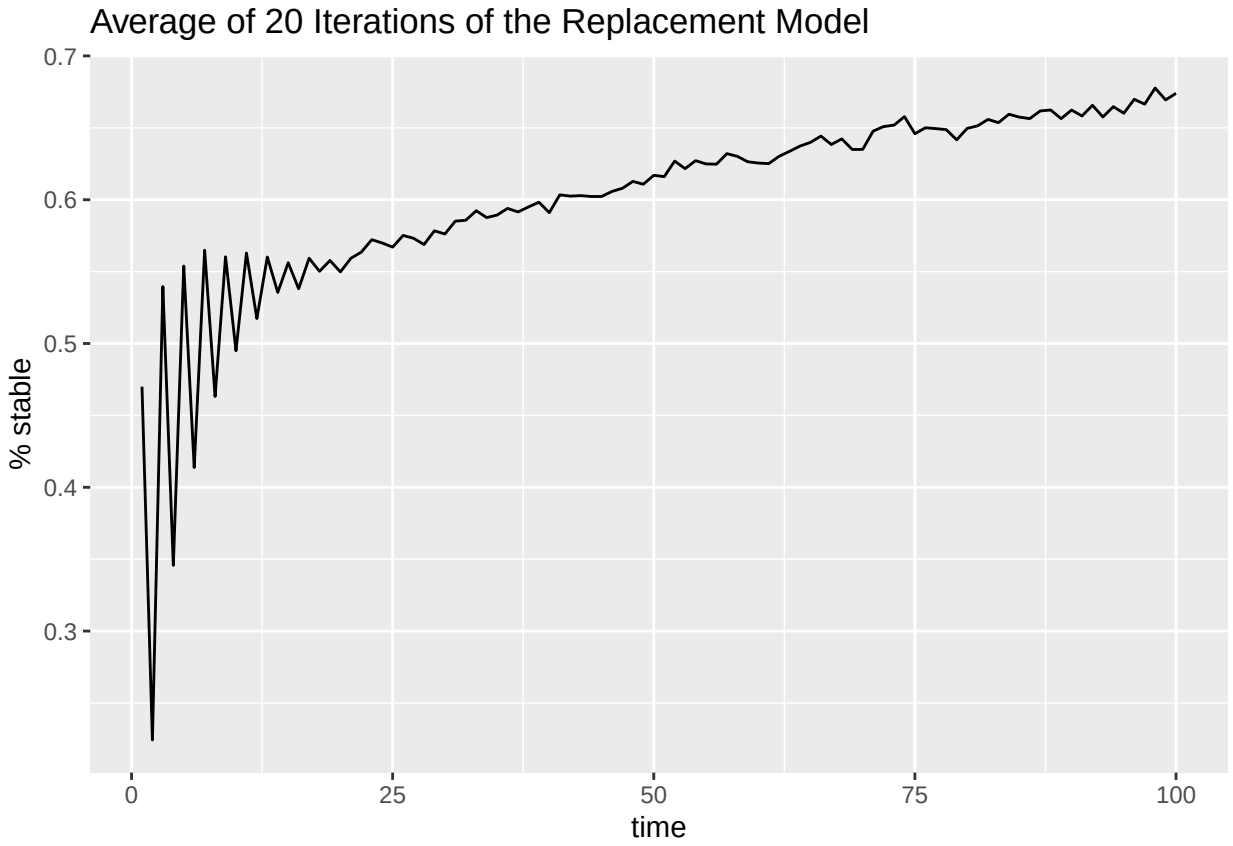



```
replacementModelFunc <- function(){
  replacementModel <- replacementIntervention(edgeList = testEdges,
                                              nodeList = testNodes,
                                              nTime = 100,
                                              interventionFreq = 5)
}

set.seed(4321)
replacementModelIterations <- iterateSim(nIter = 20,
                                         fun = replacementModelFunc)
```

```
ggplot(replacementModelIterations) +
  geom_line(aes(x = time, y = pStable, group = iteration)) +
  labs(title = '20 Simulations of the Replacement Model',
       y = '% stable')
```




```
replacementModelIterationsAve <- replacementModelIterations %>%
  dplyr::group_by(time) %>%
  dplyr::summarise(pStable = mean(pStable))

ggplot(data = replacementModelIterationsAve) +
  geom_line(aes(x = time, y = pStable)) +
  labs(title = 'Average of 20 Iterations of the Replacement Model',
       y = '% stable')
```

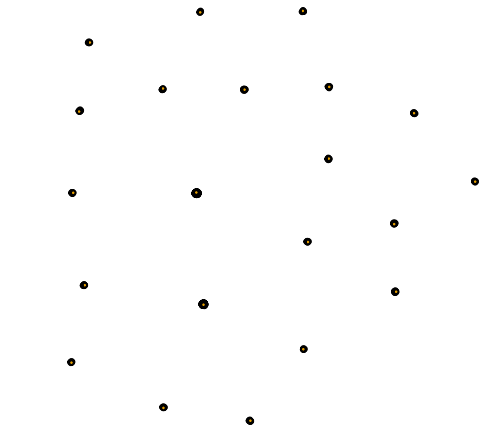


Utilizing networks - a possible extension

The model takes advantage of a bipartite structure. There are people and they are connected to facilities (either rehabs or postRehabs). These bipartite graphs also have start structures since a person can only be in one facility, while a facility can hold many people. If we project the bipartite graph into one that only observes indirect **people** − **people** connections, then we would get n fully connected subgraphs - where n is the combined number of rehab and postRehab facilities.

Let's take a look at one of the graphs we preserved during our simulation

```
exampleGraph <- replacementModel$g[[5]] #let's pull the graph from the 5th turn

p2p <- people2people(g = exampleGraph) #let's get rid of the star structure and make direct connections

plot(p2p, vertex.size = 2, vertex.label = '', edge.arrow.mode = 0)
```



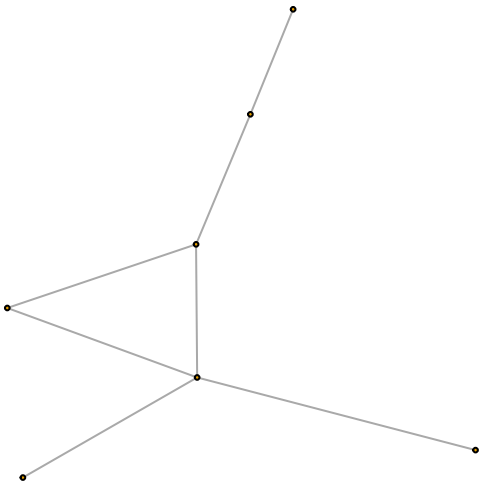
.

What could be interesting, is if we restructure these fully connected subgraphs into something more realistic. We can do this by removing random edges from the graph.

```
set.seed(4321)
stochasticStructure <- removeRandomEdges(g = p2p,
                                         toRemove = .75) #remove 75% of edges

stochasticCommunities <- observeCommunities(stochasticStructure)
```

Removing random edges in these subcommunities makes assumes that people in a facility are not actually connected to everyone else int he facility. But instead, interact with only a few people within the facility. The `observeCommunities` function provides a list of subgraphs. There are 43 at this turn of the simulation. Let's take a look at one of them

```
#change the number in the [[]] to look at a different subgraph
plot(stochasticCommunities[[5]],
     vertex.size = 2,
     vertex.label = '',
     edge.arrow.mode = 0)
```



These subgraph structures could help us with the simulation. As of now, a person's probabilty of relapsing or recoverying is wholey dependent on outside sources (the rates of the facilities). Maybe we can provide an attribute to the individual person which can buffer the rates of the facilities. This attribute could also be influenced by one's neighbors - maybe a form of social contagion.