Example Models

Here I implement how to create two different example models of influencer sets, how to initiate actor opinions and how to simulate opinion changes/updates throughout a user defined number of iterations or epochs.

- $\mathscr{A}$: finite set of actors
- $\mathscr{O} = \{+1, -1\}$: set of possible opinions
- $X_n(a) \in \mathscr{O}$: last opinion emitted by actor a until instant n
- $A_n \in \mathscr{A}$: actor that emitted an opinion on instant n
- $X_n = (X_n(a) : a \in \mathscr{A})$: list of last opinions emitted by the set of actors until instant n
- $X_0,\ X_1,\ X_2, \ldots$: sequence describing the evolution of actor opinions over time

Definitions

Example 1

Let $\mathscr{A} = \{1, \ldots, N\}$ be the set of actors in the network.

- For each actor $a = \{2, \ldots, 5\}$, $V_{\cdot \to a} = \{a - 1,\ a + 1\}$. Also $V_{\cdot \to 1} = \{6,\ 2\}$ and $V_{\cdot \to 6} = \{5,\ 1\}$.

Example 2

Let $\mathscr{A} = \{(a_1,\ a_2) : a_1 = 1, \ldots, N, a_2 = 1, \ldots, N\}$ be the set of actors in the network.

- For each actor $(a_1, a_2) \in \mathscr{A}$: $V_{\cdot \to (a_1,\ a_2)} = \{(a_1 + 1,\ a_2), (a_1 - 1,\ a_2), (a_1,\ a_2 + 1), (a_1,\ a_2 - 1)\}$ with the convention that $N + 1 = 1$.

Packages Used

I used the `igraph` package to create the graph models.

```
library(igraph)
```

```
##
## Attaching package: 'igraph'
```

```
## The following objects are masked from 'package:stats':
##
##     decompose, spectrum
```

```
## The following object is masked from 'package:base':
##
##     union
```

Implementation

Example 1

```
create_unidimensional_model <- function(N){
  #graph from edgelist
  g <- graph_from_edgelist(cbind(c(1:N), c(2:N, 1)), directed = FALSE)
```

```
  return(g)
}
```

```
g1 <- create_unidimensional_model(N = 10)
```

Example 2

```
# this function creates the bidimensional model with N² actors
create_bidimensional_model <- function(N){

  g = graph.lattice(c(N,N))

  #close lattice
  v1 = seq(from =1, to = N,by = 1)
  v2 = seq(from = 1, to = (N^2) - (N-1), by = N)
  v3 = c(v1,v2)

  v1 = seq(from = (N^2) - (N-1), to = N^2, by = 1)
  v2 = seq(from = N, to = N^2, by = N)
  v4 = c(v1,v2)

  a = rbind(v3, v4)
  a2 = matrix(a,nrow=length(a),ncol = 1)

  g = add.edges(g,a2)

  return(g)
}
```

```
g2 <- create_bidimensional_model(N = 10)
```

We've created both models, but before plotting to take a look at them, let's initiate the actor's opinions:

Initiating actor opinions

First let's define the initiate opinions function:

```
# this function initiates the opinions of the actors of a model and plots the graph
# prob_minus is the probability of an actor starting with a -1 opinion and # prob_plus is the probabili
# the seed allows us to create reproducible examples

initiate_opinions <- function(g, seed, prob_minus, prob_plus){
  set.seed(seed)
  X0 = sample(c(-1,1), length(V(g)), replace = TRUE,
              prob = c(prob_minus, prob_plus))

  V(g)$opinion = X0
  V(g)$color <- ifelse(V(g)$opinion == 1, "blue", "red")

  return(g)
}
```

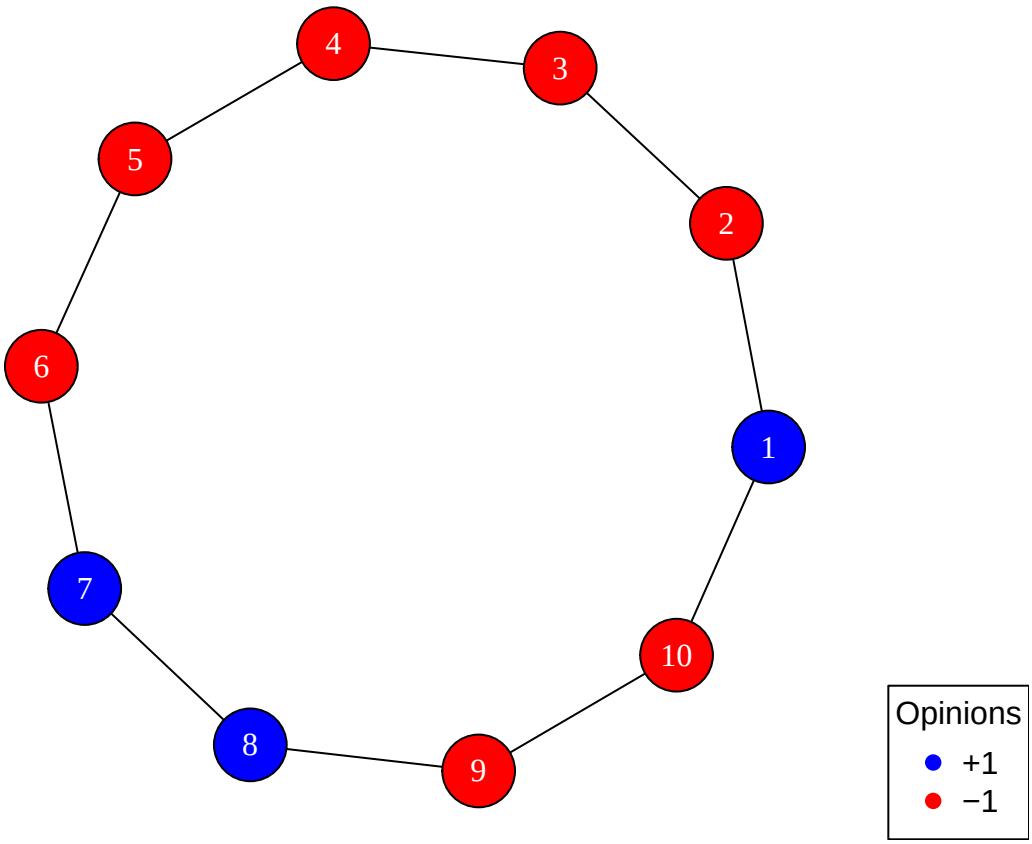
Now let's define the plot opinions function, since we will use it various times.

```r
plot_opinions <- function(g){

  par(mar = c(0, 0, 0, 0)) # Set the margin on all sides to 0
  plot(g, vertex.label.color = "white", edge.color = "black",
       vertex.size = 20)
  legend("bottomright", bty = "c", pch = 19,
       legend=c("+1", "-1"),
       col=c("blue", "red"), title = "Opinions")

}
```

Example 1

```r
g1 <- initiate_opinions(g1, seed = 1234,
                        prob_minus = 0.5, prob_plus = 0.5)
```

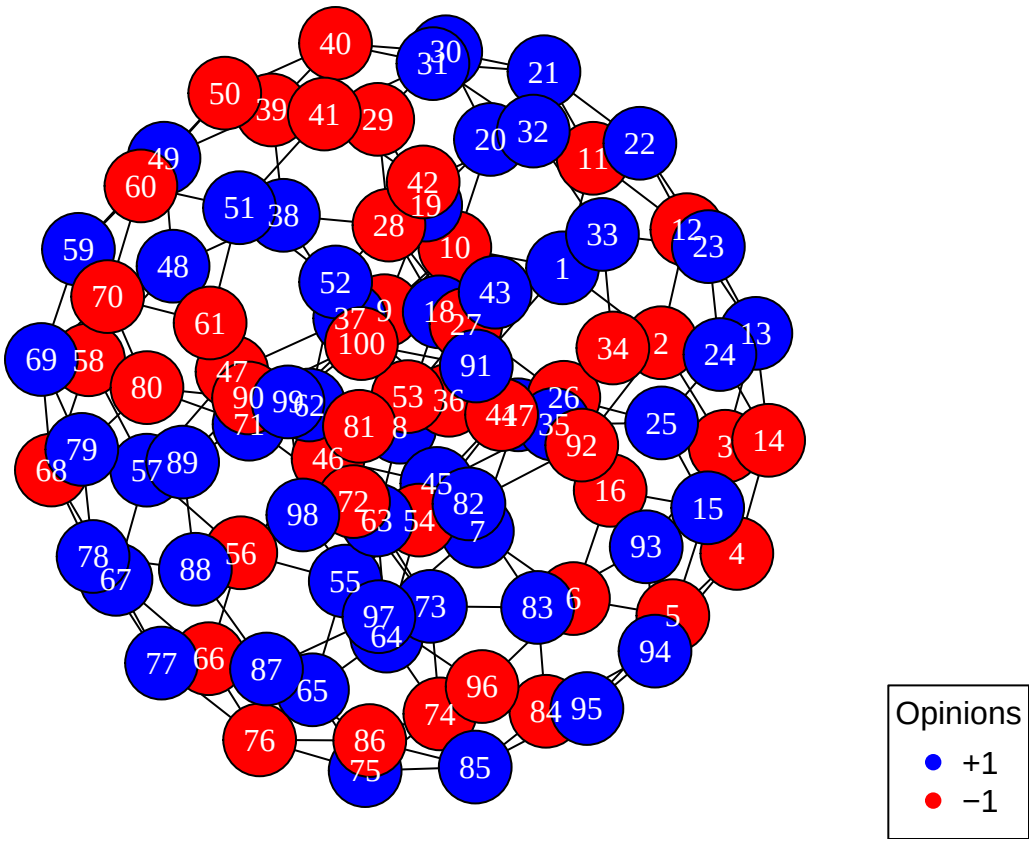
```r
plot_opinions(g1)
```



Example 2

```r
g2 <- initiate_opinions(g2, seed = 1234,
                        prob_minus = 0.5, prob_plus = 0.5)
```

```r
plot_opinions(g2)
```



Simulating opinion changes over time

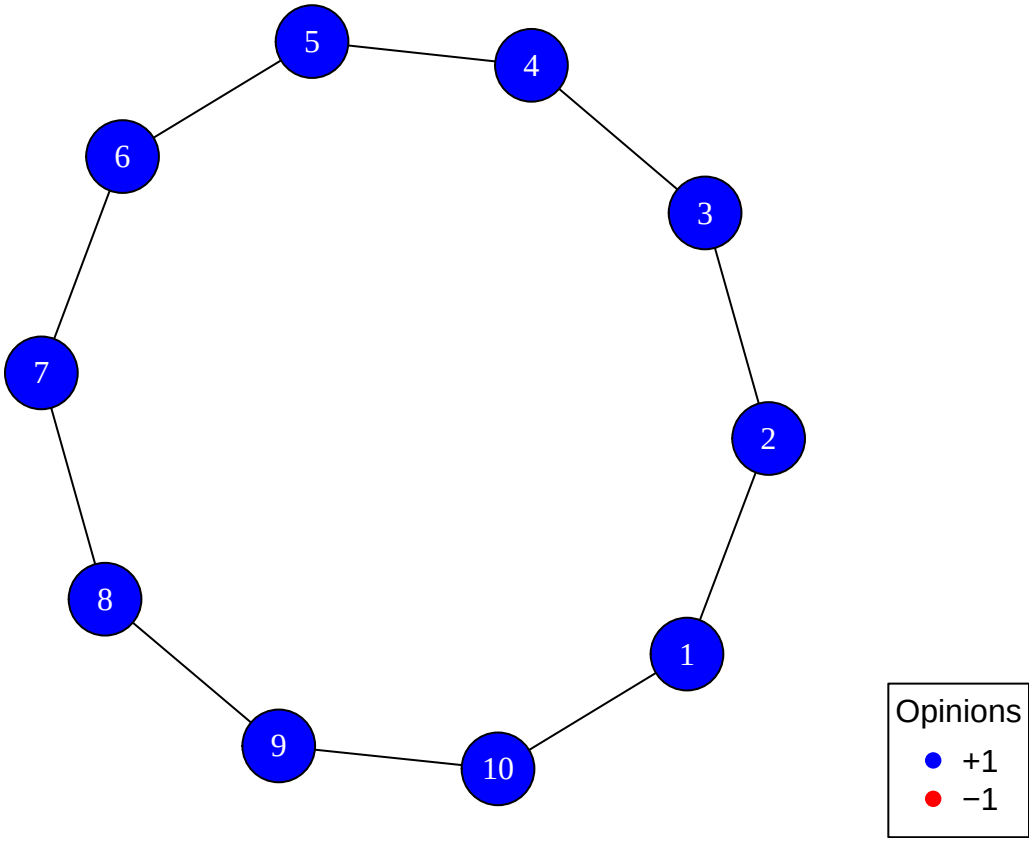
```
# this functions simulates the change of opinion during during a user given number of epochs
# this function returns the graph with final opinions and a dataframe with the sampled
# actor in the first column and its influencer in the second column
model_opinions <- function(g, seed, epochs){
  df <- data.frame(Takes=vector(length = epochs),
                   Gives=vector(length = epochs))
  set.seed(seed)
  for (n in seq(1:epochs)) {

    b = sample(V(g), 1) #sample an actor

    #This IF avoids problems in case the vertex is disconnected from the graph
    if (degree(g)[b] != 0) {
      influencer = sample(neighbors(g, b), size=1) #sample an influencer
      V(g)[b]$opinion = V(g)[influencer]$opinion #get the influencer's opinion

      df[n,] = c(as_ids(V(g)[b]), as_ids(V(g)[influencer]))

    } else {df[n,] = c(b, b)}

    V(g)$color <- ifelse(V(g)$opinion == 1, "blue", "red")

  }
```

```
  return(list(g, df))

}
```

Example 1

```
l1 <- model_opinions(g1, seed = 1234, epochs = 100)

g1_final <- l1[[1]]
df1 <- l1[[2]]
```

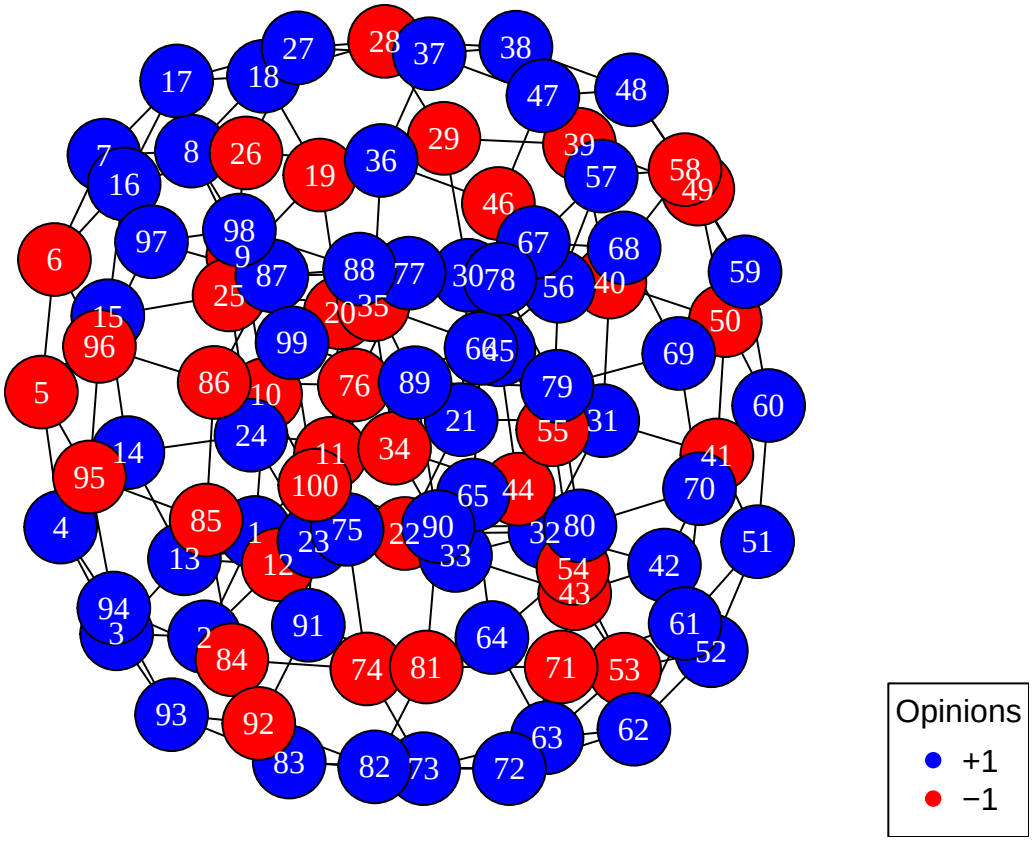
```
plot_opinions(g1_final)
```



Example 2

```
l2 <- model_opinions(g2, seed = 1234, epochs = 100)

g2_final <- l2[[1]]
df2 <- l2[[2]]
```

```
plot_opinions(g2_final)
```



As we can see, after the iterations, that our unidimensional model comes to a consensus in opinions but out bidimensional model does not.

Searching for original opinions

Each final actor opinion can be mapped back to an original influencer. dataframe containing all influences throughtout the simulation, and we will use this dataframe to find the original influencers for each actor.

```
#this function searches for the origin (influencers) of opinions
get_origin <- function(g, df, epochs){

  for (i in seq(1:length(V(g)))) {

    size = epochs
    origin = i
    index = match(origin, df$Takes[1:size])

    while (is.na(index) == FALSE  && size != 0) {
      size = index-1
      origin = df$Gives[index]
      index = match(origin, df$Takes[1:size])
    }
    V(g)[i]$origin = origin
  }

  return(g)
```

```
}
```

Example 1

```
g1 <- get_origin(g1, df1, epochs = 100)

V(g1)$origin
```

```
##  [1] 6 6 6 6 6 7 8 8 8 9
```

Example 2

```
g2 <- get_origin(g2, df2, epochs = 100)

V(g2)$origin
```

```
##   [1]  1  1 13  5  6  6  7  8  8 10 11 12 13 13 15 15 18 18 10 10 31 12 33 24 26
##  [26] 26 37 38 29 29 31 31 33 34 34 37 37 48 39 29 50 50 42 44 45 46 57 48 48 50
##  [51] 52 52 53 54 54 55 55 58 59 50 71 62 64 64 65 65 77 88 69 69 71 82 73 74 75
##  [76] 76 77 88 88 88 81 82 82 94  5 85 97 87 89 89 91 92 94 94  5  5 97  8 89 99
```

Getting the proportion of final opinions of an actor by simulating the model

```
# this function simulates the model 100 times and returns a given actors
# last opinion from each simulation
last_opinion <- function(g, actor, prob_minus, prob_plus, n_sim, epochs){
  opinion <- vector(length = n_sim)

  for (i in seq(1:n_sim)) {
    g <- initiate_opinions(g, seed = i,
                           prob_minus = prob_minus, prob_plus = prob_plus)
    l <- model_opinions(g, seed = i, epochs = epochs)
    g <- l[[1]]

    opinion[i] <- V(g)[actor]$opinion
  }
  return(opinion)
}
```

Example 1

First let's get the last opinions 100 times with the same probability of having an inital opinion +1 or -1.

```
opinion_10 <- last_opinion(g1, actor = 10, prob_minus = 0.5, prob_plus = 0.5,
                           n_sim = 100, epochs = 100)
opinion_5 <- last_opinion(g1, actor = 5, prob_minus = 0.5, prob_plus = 0.5,
                           n_sim = 100, epochs = 100)

#proportion of +1
length(opinion_10[opinion_10 == 1])/length(opinion_10)
```

```
## [1] 0.53
```

```
length(opinion_5[opinion_5 == 1])/length(opinion_5)
```

```
## [1] 0.5
```

Now let's get the last opinions 100 times with the probabilities of having an initial opinion +1 as 0.3 and of -1 as 0.7.

```
opinion_10 <- last_opinion(g1, actor = 10, prob_minus = 0.7, prob_plus = 0.3,
                           n_sim = 100, epochs = 100)
opinion_5 <- last_opinion(g1, actor = 5, prob_minus = 0.7, prob_plus = 0.3,
                           n_sim = 100, epochs = 100)

#proportion of +1
length(opinion_10[opinion_10 == 1])/length(opinion_10)
```

```
## [1] 0.27
```

```
length(opinion_5[opinion_5 == 1])/length(opinion_5)
```

```
## [1] 0.29
```

We can see from these results that the proportion of times that a given opinion is final is approximately the initial proportion (probability) of the same opinion.

Robots in a social network

We consider a robot to be an actor whose opinion is always the same as its initial opinion, being uninfluenced by other actors, while still influencing those actors connected to it.

Let's define the function that will simulate opinion changes considering there is a user defined robot in the network.

```
# this functions simulates the change of opinion during a user defined number of
# epochs with a given actor being a robot (a robot is an actor that does not
# change its original opinion)
model_opinions_robot <- function(g, seed, epochs, robot, robot_opinion){

  V(g)[robot]$opinion <- robot_opinion

  df <- data.frame(Takes=vector(length = epochs),
                   Gives=vector(length = epochs))
  set.seed(seed)
  for (n in seq(1:epochs)) {
    b = sample(V(g), 1) #sample an actor
    if (b != robot) {
      influencer = sample(neighbors(g, b), size=1) #sample an influencer
      V(g)[b]$opinion = V(g)[influencer]$opinion #get the influencer's opinion
      df[n,] = c(b, influencer)
    } else {
      df[n,] = c(b, b)
    }

  }

  V(g)$color <- ifelse(V(g)$opinion == 1, "blue", "red")

  return(list(g, df))
```

```
}
```

```
l1_robot <- model_opinions_robot(g1, seed = 3, epochs = 100,
                            robot = 5, robot_opinion = -1)

g1_final_robot <- l1_robot[[1]]
df1_robot <- l1_robot[[2]]

g1_final_robot <- get_origin(g1_final_robot, df1_robot, epochs = 100)

V(g1_final_robot)$origin
```

```
##  [1] 1 1 4 4 5 5 8 9 9 9
```

Getting the proportion of final opinions of an actor by simulating the model with a robot

```
# this function simulates the model 100 times and returns a given actors
# last opinion from each simulation (a given actor is a robot)
last_opinion_robot <- function(g, actor, robot, robot_opinion, prob_minus, prob_plus, n_sim, epochs){

  opinion <- vector(length = n_sim)

  for (i in seq(1:n_sim)) {

    g <- initiate_opinions(g, seed = i,
                           prob_minus = prob_minus, prob_plus = prob_plus)
    l <- model_opinions_robot(g, seed = i, epochs = epochs,
                              robot = robot, robot_opinion = robot_opinion)
    g <- l[[1]]

    opinion[i] <- V(g)[actor]$opinion
  }
  return(opinion)
}
```

```
opinion_10_robot <- last_opinion_robot(g1, actor = 10, robot = 5, robot_opinion = -1,

opinion_2_robot <- last_opinion_robot(g1, actor = 2, robot = 5, robot_opinion = -1,

#proportion of +1
length(opinion_10_robot[opinion_10_robot == 1])/length(opinion_10_robot)
```

```
## [1] 0.23
```

```
length(opinion_2_robot[opinion_2_robot == 1])/length(opinion_2_robot)
```

```
## [1] 0.2
```

Here we notice that the previous pattern we observed isn't maintained. Since the robots opinion isn't changed and kept at -1, then the proportion of +1 is now lower than the initial proportion/probability of +1 ($0.2 < 0.3$).