Schelling Segregation Model

Mikah Nelson

2023-09-08

Contents

Model Overview

Purpose

Determine whether segregation can be explained by individual motivations to avoid being a local minority. Agents will move around a two-dimensional space depending on whether they are a racial minority within their local neighborhood.

Agents

A population of agents will be generated; each agent will have a race (black or white) and a position in a 2-dimensional space.

Life Cycle

1. Assess
2. Move

Assess Each agent will first calculate how far they are from each other agent in the 2D space. The agent will then identify all agents who reside within a set radius around them. These agents constitute the focal agent's neighborhood.

The focal agent will then determine the race of all agents in their neighborhood. Finally, the agent will determine whether they are in the minority race in their neighborhood.

Move In this stage of the life cycle, the agent makes one decision: if the agent is not in the minority then they will stay where they are. If the agent is in the minority, they will move to a new random location.

Analysis

The model will plot the position of agents before and after the movement phases

Model

Packages

```r
library(ggplot2)
```

Parameters

```r
#Population Size#
#The number of agents to generate
popSize<-1000



#Neighborhood Radius#
#How close an agent must be to be a neighbor
#smaller radius, fewer patches need to be homogenous for people to be happy
radius<-20



#Minority Threshold#
#The proportion of same-race neighbors below which an agent will decide to move
threshold<-.5

#maximum number of movement cycles that agents will go through
maxSteps <- 100

#Current Steps#
#create a variable to track completed movement cycles
steps <- 0
```

Functions

```r
agentGenerate<-function(n){

  #Assign each agent a random race
  race<-sample(c("black","white"),n,replace=T)

  #Assign each agent a random x-position
  xPos<-runif(n,0,100)

  #Assign each agent a random y-position
  yPos<-runif(n,0,100)

  #Generate the agents
  agents<-data.frame(race,xPos,yPos)

  #Output the agents
  return(agents)
}



#Assess#
#Have the agents assess their neighborhood and decide whether they want to move
assess<-function(agents,radius,threshold){

  #Computes the straight-line distance between each agent and all other agents
  distances<-as.matrix(dist(agents[,2:3]))

 moveDecision <-rep(NA, nrow(agents))

  for(a in 1:nrow(agents)){

    #Determine which agents are the focal agent's neighbors
    neighbors<-distances[a,]<=radius

    #Determine which agents are of the same race as the focal agent
    sameRace<-agents$race[a] == agents$race

    #have the agent determine the proportion of neghbors who are of the same race
    assessment <- sum(sameRace & neighbors)/sum(neighbors)

  #have the agent decide whether or not to move
     moveDecision[a] <- assessment<threshold

  }
  #output the agent's movement decisions!
 return(moveDecision)
}


#Move#
# a function to have agents move

move <- function(agents,moveDecision){

  #loop one-by-one through agents
  for(a in 1:nrow (agents)){

    #if this particular agent bracket a decides to move, then...
    if(moveDecision[a]==T){

      #assign them a new random x and y position
      #silly way to move, since its a random place in neighborhood
      agents$xPos[a] <- runif(1,0,100)
      agents$yPos[a] <- runif(1,0,100)
    }

  }
  #output the agents with their (potential) new positions
  return(agents)
}
```

Life Cycle

```r
#Generate the agents
agents<-agentGenerate(popSize)

#save the starting state of the agents dataframe
startAgents <- agents

while(maxSteps>steps){
  #increment steps by 1
  steps <- steps+1

  #Assess#

#have the agents assess their neighborhood and decide whether to move
moveDecision <- assess(agents,radius,threshold)

#Move#

#have the agents move if they are dissatisfied with their current
#neighborhood

agents <- move(agents,moveDecision)

#if no one wants to move...
#break automatically stops any loop, end the loop
  if(sum(moveDecision)==0){
    break
  }

}

#save the state of the agents data frame after they move
endAgents <- agents
```

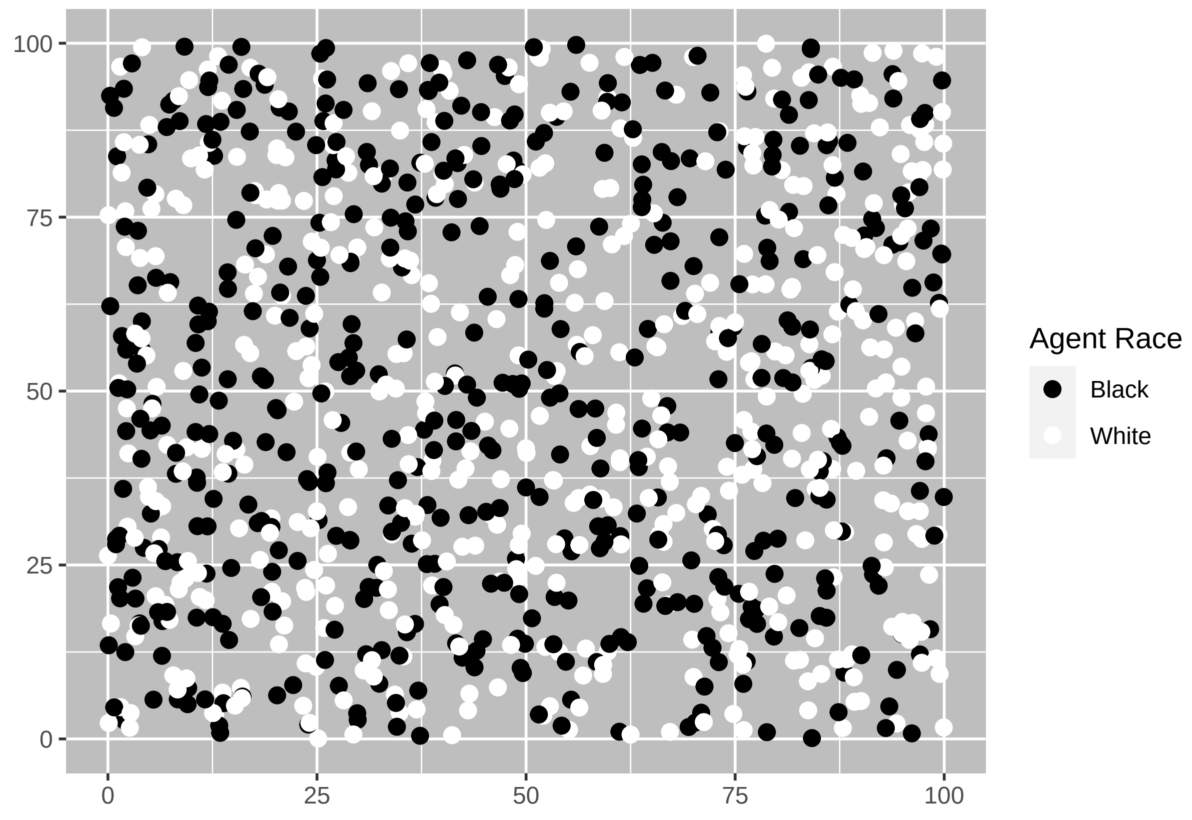
Analysis

```r
t1plot <- ggplot(data=startAgents,
                 aes(x=xPos,
                     y=yPos,
                     color=race)) +
  labs(x="", y="", color="Agent Race")+
  geom_point(size=2.5)+
  scale_color_manual(values=c("black", "white"),
                     labels=c("Black", "White"))+
  theme(panel.background = element_rect(fill="grey"))
```

```
t2plot <- ggplot(data=endAgents,
                 aes(x=xPos,
                     y=yPos,
                     color=race)) +
  labs(x="", y="", color="Agent Race")+
  geom_point(size=2.5)+
  scale_color_manual(values=c("black", "white"),
                     labels=c("Black", "White"))+
  theme(panel.background = element_rect(fill="grey"))
```

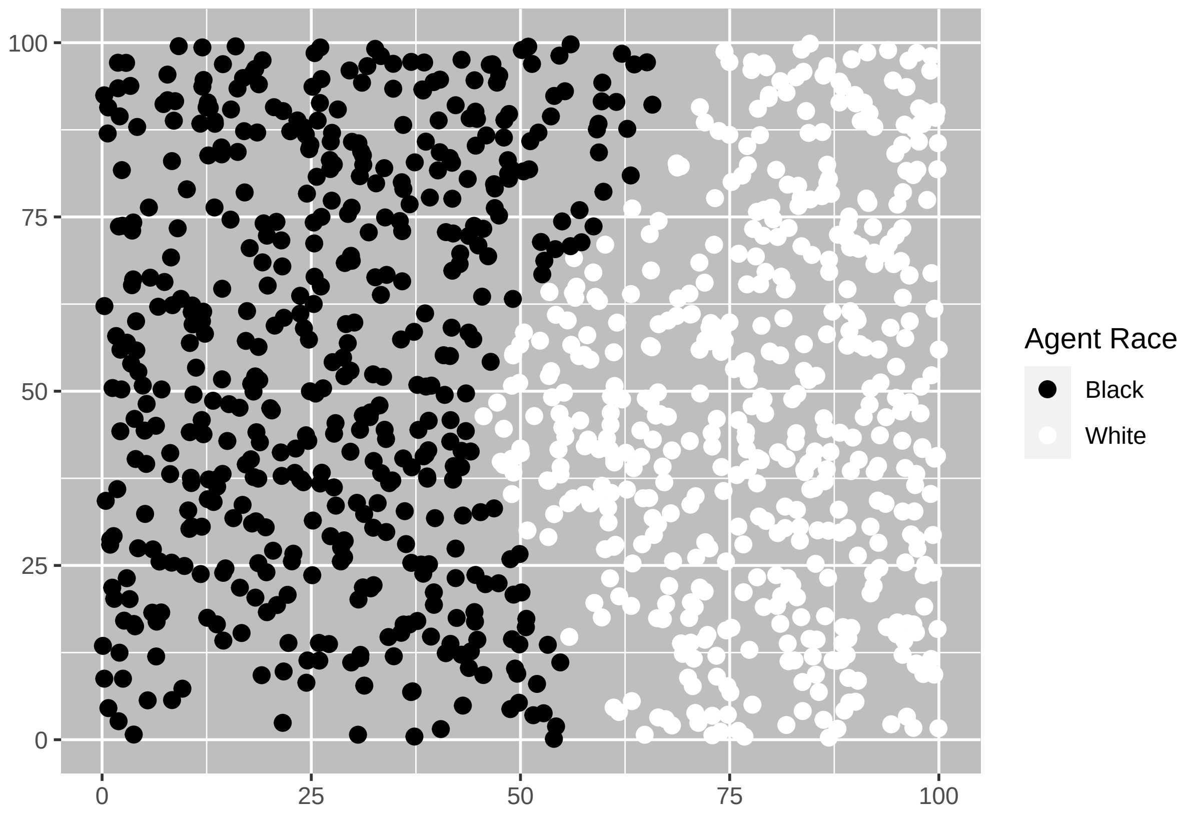
Consistent with expectations, agents are initially not highly segregated:

```
t1plot
```



As long as individuals do not want to be the minority in their local neighborhood, the agents will have the desire to be in different local environments. So, after agents are allowed to move in accordance with their motivations, highly segregated neighborhoods do appear:

```
t2plot
```


Agent Race
Black
White