Econ 21410 - Problem Set II

Schelling's segregation and related models[*]

April 19, 2014

This homework should be done in LaTeX The homework will be graded on correctness, but will also heavily weight clarity and professionalism. Being able to produce clean, clear, well documented write-ups and code is an important skill which will be rewarded. Its better to not do some of the harder parts than to turn in an incomprehensible document. Your R script as well as a log-file should be submitted. Alternatively, use knitr to print your code inline in your latex document.

Make sure to write code which is clear and flexible. Read the whole problem before you begin coding. Some parameters will change and the code should be written in a way to make this easy to implement. We will re-use code in this course. Flexibility and documentation now will save you headaches later in the quarter. Remember to properly indent your code!

SUBMISSION: The homework must be emailed to Oliver and myself by 2p.m. Monday, April the 14th. The email must include a pdf with the filename lastname_pset2.pdf and R code called lastname_pset2_code.R where "lastname" should be replaced with your last name. The subject of your email should be [ECON 21410: pset2 submission]

Remember that asking and answering questions on our github page, coming to office hours to ask questions, and contributing to the class wiki are all worth participation credit, which is 10% of your grade in this class.

1 Control-Flow in R

Complete the following exercise on if/for/while loops in R (use correct indentation!):

1. initialize a variable z to 0
2. write a for-loop that runs from 1 to 1000, call the current iteration i.
3. have the code print out what iteration it is on every 100 iterations
4. in each iteration draw a random value from c(1,2,3)
5. if the drawn random value is equal to 3, set it equal to 1
6. if the drawn random value is equal to 2, set it equal to 1

[*]Please email johneric@uchicago.edu and obrowne@uchicago.edu if you have questions.

7. if the drawn random value is equal to 1, add it to z
8. After the loop ends, print the value of z.

```r
z <- 0  #1
for (i in 1:1000) {
    # 2 3
    if (i%%100 == 0) {
        print(i)
    }
    rv <- sample(1:3, 1)  #4
    if (rv == 3) {
        # 5
        rv <- 1
    } else if (rv == 2) {
        # 6
        rv <- 1
    } else if (rv == 1) {
        # 7
        z <- z + rv
    }
}
```

```
## [1] 100
## [1] 200
## [1] 300
## [1] 400
## [1] 500
## [1] 600
## [1] 700
## [1] 800
## [1] 900
## [1] 1000
```

```r
print(z)  #8
```

```
## [1] 329
```

2 Schelling's Segregation Model

Consider a version of Schelling's Segregation Model implemented in the following way:

1. $\{n_1, ..., n_N\}$ are generated where n^r are red, n^g are green, and n^b are blue.
2. Each individual is initially placed (uniformly) at random on $[0, 1] \times [0, 1]$
3. Start with individual n_1 (who has type $t \in (r, b, g)$) then proceed with the algorithm.

a) Take individual $n_{i,t}$. If *at least* j_t of their m_t closest neighbors are the same color as them, move on to n_{i+1},

b) if fewer than j_t of their m_t closest neighbors are the same color as them, randomly draw a new living location and move on to n_{i+1}.

c) Continue this process until no individuals remain who wish to move.

Machinery for the Schelling Model:

1. write a function that calculates the distances between a coordinate point (x_i, y_i) and a vector of coordinate points $[(x, y)]$. This should return a vector of distances

```r
# Part 2: Schelling's Segregation Model Code 1:
# Distance Function ==============================

Dist <- function(a, b) {
    if (length(b) > 2) {
        rowSums((matrix(a, dim(b)[1], length(a), byrow = T) -
            b)^2)
    } else if (length(b) == 2) {
        sum((a - b)^2)
    } else "broken"
}

CountSimilarNeighbors <- function(coords, types, ind,
    m) {
    distances <- rep(NA, nrow(coords))
    distances[] <- Dist(as.matrix(coords[ind, ]), as.matrix(coords))
    neighbors <- order(distances)[2:m + 1]
    return(sum(types[neighbors] == types[ind]))
}

# ==============================
```

2. Write a function which simulates Schelling's Segregation model. The function should take the size of each population, j_t and m_t for each type. The function should allow for up to three types.

   - The function should output a plot (or the data necessary to output a plot) of the initial distribution of agents and the final distribution of agents (even better if it outputs some intermediate plots)
   - The function should return the data for the final allocation of individuals.
   - The function should return the number "cycles" the algorithm takes

```r
# Part 2: Schelling's Segregation Model Code 2:
# Simulation Function ==============================

SchellingSim <- function(num.types = 2, pop = c(250,
    250), j = c(4, 4), m = c(8, 8), show.iter = "false") {
```

```r
# Simulates a version of Schelling's segregation
# model as descrived in Ec21410 PSet2

# Args:

# num.types: number of different populations in
# model

# pop : a (1 x num.types) integer vector specifying
# the populations of each type

# j : a (1 x num.types) vector on the unit interval
# specfiying the fraction of similar individuals
# each type wants to live with

# m : a (1 x num.types) vector on the unit interval
# specfiying the fraction of similar individuals
# each type wants to live with

# show.iter: set to 'iter' to display iteration
# count, countains 'figs' to output figures

# Returns: -1 : if simulation fails

# Otherwise returns a data frame with each
# individuals

# Start timing
fn.start <- proc.time()

# Error Checking
if (length(pop) != num.types | length(j) != num.types |
    length(m) != num.types) {
    print("Check Length of inuput vectors")
    return(-1)
}

# Generate Agents
types <- rep(1, pop[1])
for (t in 2:num.types) {
    types <- c(types, rep(t, pop[t]))
}
types <- as.factor(types)
x <- runif(sum(pop))
y <- runif(sum(pop))
happy <- rep(FALSE, sum(pop))
data <- data.frame(types[], x, y, happy)
plt0 <- qplot(x, y, data = data, color = types,
    size = I(5), alpha = I(0.5), main = "Initial Distribution")

# initialize loop
```

```r
    iter <- 0
    max.num.iter <- 300
    # print(plt0)

    # loop until all agents are happy
    while (!all(data$happy)) {

        iter <- iter + 1

        # exit if does not converge in 300 iterations
        if (iter > 300) {
            print("Convergence Failed")
            return(-1)
        }

        # For all individuals
        for (ind in 1:sum(pop)) {
            type <- data$type[ind]
            # Test if they are happy
            data[ind, 4] <- (CountSimilarNeighbors(data[,
                2:3], data[, 1], ind, m[type]) > j[type])
            # Move them if they are not happy
            if (data[ind, 4] == 0) {
                data[ind, 2:3] <- runif(2)
            }
        }
        # Plot agents
        plt <- qplot(x, y, data = data, color = types,
            size = I(5), alpha = I(0.5), main = paste("Distribution, Iteration:",
                iter))

        # Display plot or iteration count
        if (show.iter == "plots") {
            print(plt)
            print(paste("Iteration:", iter, "Happy:",
                mean(data$happy) * 100, "%"))
            Sys.sleep(0.1)
        } else if (show.iter == "iter") {
            print(paste("Iteration:", iter, "Happy:",
                mean(data$happy) * 100, "%"))
        }
    }
    plt <- qplot(x, y, data = data, color = types,
        size = I(5), alpha = I(0.5), main = "Final Distribution")
    # Generate outputs
    output <- list(data = data, final.plot = plt, initial.plot = plt0,
        iter = iter, time = proc.time() - fn.start)
    return(output)
}

# ============================
```

3. We would like to study the "amount" of segregation in an our "city". To do this, we will write a function which will compute three different segregation metrics. The function should take the simulated final data from your Schelling function and return the three following metrics (focus on the first metric, questions involving the second two indexes will in total be not be worth more than 5% of the grade and are more challenging):

   - Similar neighbors index: For each individual, calculate the proportion of their m_t nearest neighbors that are the same type as them. Take the average of this number across individuals.

   - Dissimilarity index: grid up the $[0, 1] \times [0, 1]$ city into "blocks" of size 0.2×0.2. Using these blocks have the function return the dissimilarity index.[1]

   - Gini index: Using the same grid as above, have the function return the gini index.

```r
# Part 2: Schelling's Segregation Model Code 3:
# Segregation Indicies
# =============================

SimilarNeighborIndex <- function(data, m = 8) {
    # This function calulates the average number of
    # similar neighbors each individual has

    # Input: The data structure used in our
    # simulations: a data frame where the first column
    # are the

    # types of each individuals, the second and third
    # column are the x and y coordinates of each
    # individual

    # Output: The average fraction of similar neighbors
    # each individual has

    sum <- 0
    # For each individual how many of the m nearest
    # neighbors are similar
    for (ind in 1:nrow(data)) {
        sum <- sum + CountSimilarNeighbors(data[, 2:3],
            data[, 1], ind, m)
    }
    # Divide by the number of individuals
    mean <- sum/nrow(data)
    # Divide by number of neighbors considered to get
    # ratio
    proportion <- mean/m
    return(proportion)
}
```

[1] Calculating the dissimilarity index and the gini index are a fair amount of additional work. These will be worth substantially fewer points than the similar neighbor index. Make sure you finish the rest of the problem set first.

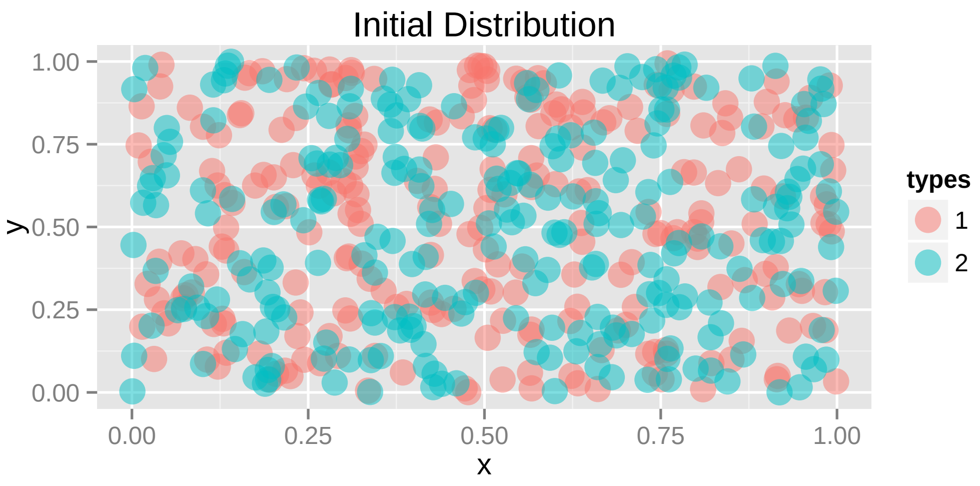
```r
DissimilarityIndex <- function(data, grid.size = 5,
    min = 1) {
    # This function calculates a dissimilarity index
    # from the output of our Schelling function Inputs:

    # data = The data structure used in our
    # simulations: a data frame with rows names x and y
    # describing the location of each individual, and
    # another row named type describing the location of
    # each individual

    # grid.size = The number of points used on each
    # axis to create the grid defining neighborhoods

    # type = the index of the minority types

    # Output: The Dissimilarity index

    # Index = (sum_i t_i * |p_i - P|)/ (2*T*P*(1-P))

    # where t_i is the population of a neighborhood

    # p_i is the proportion of minorities in that
    # neighborhood

    # T is the total population

    # P is the total proportion of minorities

    tot.popn <- nrow(data)
    popn.rate <- sum(data$type == min)/tot.popn
    index <- 0

    # We partition the grid into a grid.size x
    # grid.size neighborhoods

    # Then we loop over all of these neighborhoods
    for (i in 1:grid.size) {
        for (j in 1:grid.size) {

            # Find the sub population of that neighborhood
            sub.pop <- data[data$x > (i - 1)/grid.size &
                data$x < (i)/grid.size & data$y > (j -
                1)/grid.size & data$y < (j)/grid.size,
                ]
            sub.pop.size <- nrow(sub.pop)
            if (sub.pop.size > 0) {
                # Calculate the rate of minorities in that
                # neighborhood
                sub.popn.rate <- sum(sub.pop$type ==
                  min)/sub.pop.size
                # Calculate that neighborhoods contribution to the
                # Dissimilarity Index
                index <- index + sub.pop.size * abs(sub.popn.rate -
                  popn.rate)
            }


        }
    }
    # Divide by the demoninator
    index <- index/(2 * tot.popn * popn.rate * (1 -
        popn.rate))
    return(index)
}

GINI <- function(data, grid.size = 5, min = 1) {
    # This function calculates a segregation GINI index
    # from the output of our Schelling function

    # Inputs:

    # data = The data structure used in our
    # simulations: a data frame with rows names x and y
    # describing the location of each individual, and
    # another row named type describing the location of
    # each individual

    # grid.size = The number of points used on each
    # axis to create the grid defining neighborhoods

    # type = the index of the minority types

    # Output: The Segregation GINI index

    # Index = (sum_i sum_j t_i * t_j * |p_i - p_j|)/
    # (2*T^2*P*(1-P))

    # where t_i,t_j is the population of a neighborhood

    # p_i,p_j is the proportion of minorities in that
    # neighborhood

    # T is the total population

    # P is the total proportion of minorities

    tot.popn <- nrow(data)
    popn.rate <- sum(data$type == min)/tot.popn
    index <- 0
    # We partition the grid into a grid.size x
    # grid.size neighborhoods
    # Then we loop over all of these neighborhoods once
    for (i1 in 1:grid.size) {
        for (j1 in 1:grid.size) {
            # Then we loop over all of these neighborhood again
            # in a nested loop
            for (i2 in 1:grid.size) {
                for (j2 in 1:grid.size) {

                  # Caclulation the sub populations for each of these
                  # neighborhoods
                  sub.pop1 <- data[data$x > (i1 - 1)/grid.size &
                    data$x < (i1)/grid.size & data$y >
                    (j1 - 1)/grid.size & data$y < (j1)/grid.size,
                    ]
                  sub.pop2 <- data[data$x > (i2 - 1)/grid.size &
                    data$x < (i2)/grid.size & data$y >
                    (j2 - 1)/grid.size & data$y < (j2)/grid.size,
                    ]

                  # Calculate the minority population for each of
                  # these neighborhoods
                  sub.pop.size1 <- nrow(sub.pop1)
                  sub.pop.size2 <- nrow(sub.pop2)

                  if (sub.pop.size1 > 0 & sub.pop.size2 >
                    0) {
                    # Calculate the contribution of this pair of
                    # neighborhoods to the GINI index
                    sub.pop.rate1 <- sum(sub.pop1$type ==
                      min)/sub.pop.size1
                    sub.pop.rate2 <- sum(sub.pop2$type ==
                      min)/sub.pop.size2
                    index <- index + sub.pop.size1 *
                      sub.pop.size2 * abs(sub.pop.rate1 -
                      sub.pop.rate2)
                  }
                }
            }
        }
    }
    # Divide by the denominator
    index <- index/(2 * tot.popn^2 * popn.rate * (1 -
        popn.rate))

    return(index)

}

# ==============================
```

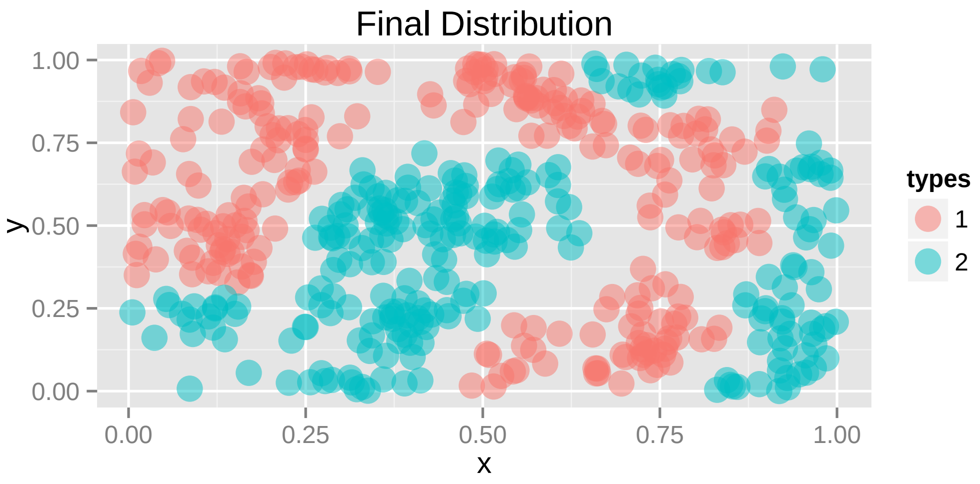
Output for Schelling

1. Run a "baseline" model with two populations of 250. Let each group care about their 8 nearest neighbors and lets assume members from both groups are "happy" if half of their nearest 8 neighbors are the same color as them.

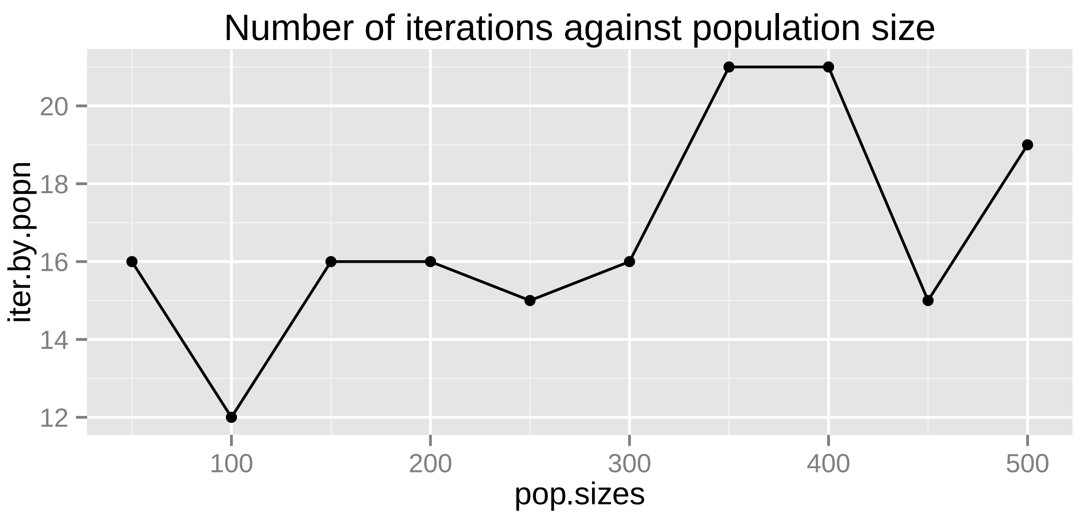
   - Make a plot of the initial distribution of individuals and the final distribution of individuals.

   - Run your model from a few different starting seeds and see how stable your results are above and discuss (2-5 sentences)





2. Make a plot showing how the number of iterations changes as you increase the populations of the two groups (symmetrically).



If you wanted to make this plot smoother you could consider plotting averages over several simulations

3. Make a table showing how run-time increases as you increase the populations of the two groups (symmetrically) (hint, see the command "system.time()").

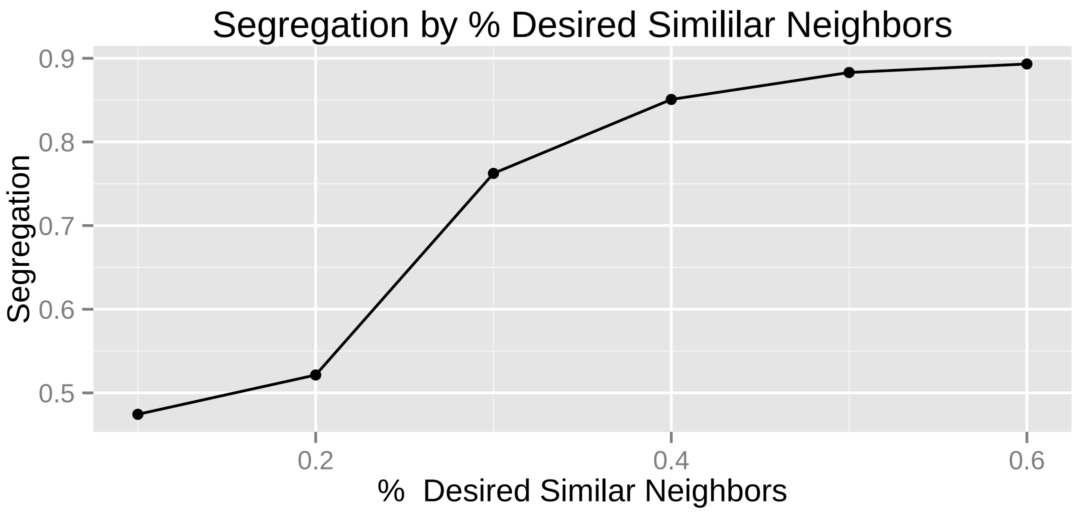
	pop.sizes	iter.by.popn	time.by.popn
1	50.00	16.00	1.46
2	100.00	12.00	1.79
3	150.00	16.00	3.92
4	200.00	16.00	5.18
5	250.00	15.00	5.82
6	300.00	16.00	8.03
7	350.00	21.00	12.50
8	400.00	21.00	14.59
9	450.00	15.00	12.14
10	500.00	19.00	17.63

Table 1: Convergence Time by Population

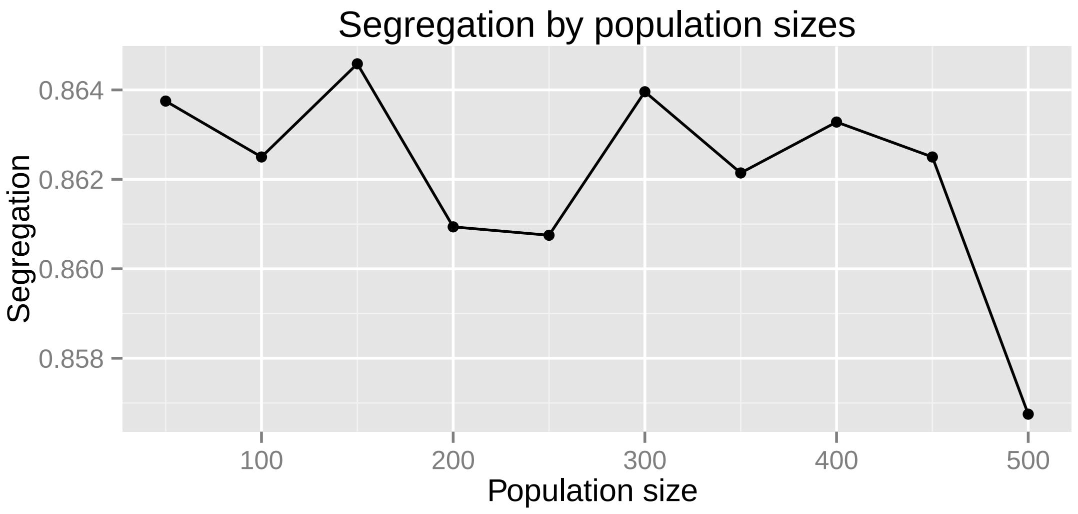
4. Calculate the similar-neighbor index, discuss (1-4 sentences).

```
## [1] "Over 5 simulations:"
## [1] "Similar Neighbor Index, Mean: 0.863 Standard Deviation: 9e-04"
## [1] "GINI Index            , Mean: 0.934 Standard Deviation: 0.03"
## [1] "Dissimilarity Index   , Mean: 0.795 Standard Deviation: 0.05"
```

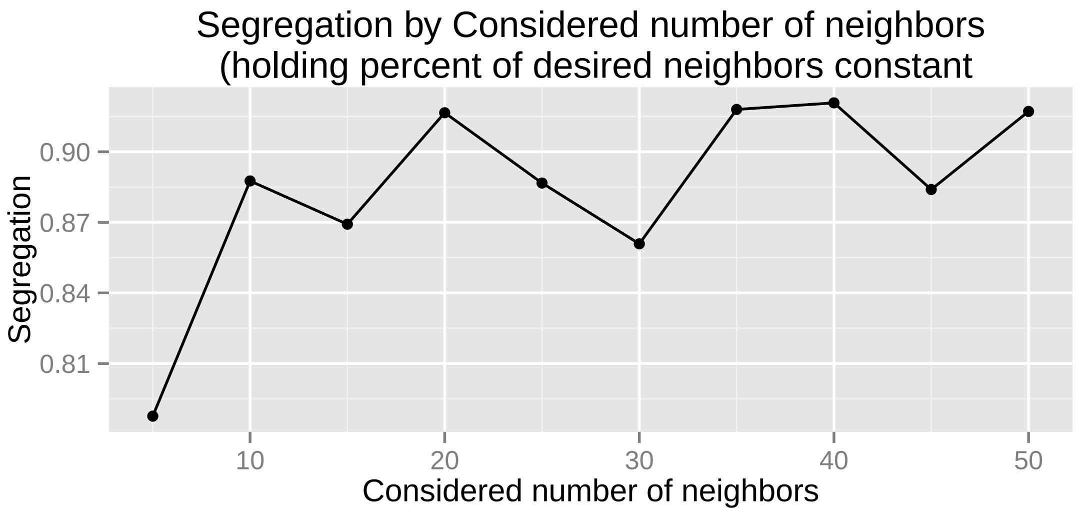
5. Make a plot of how the similar-neighbor index changes as you increase the ratio of nearest neighbors that need to be of the same type for the individual to be happy from .1 to .9/



6. Make a plot showing how the similar-neighbor index changes as you increase the number of individuals in each population from 50 to 500.



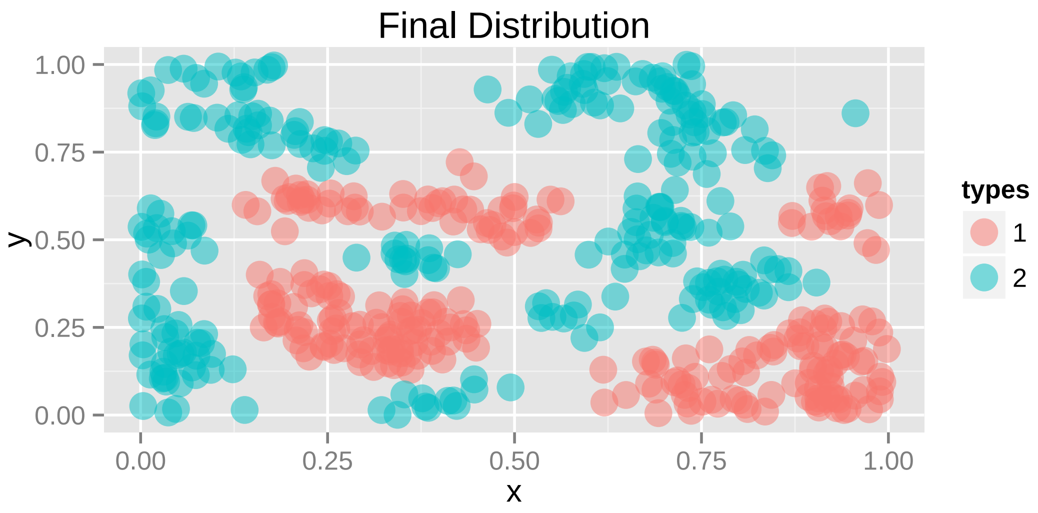
7. Make a plot showing how the similar-neighbor index changes based on the number of nearest neighbors considered (from 5 to 30).



Differences between populations: Now lets consider how the model changes when the two populations do not have the same characteristics.

1. Keeping the populations at 250, let the first population be happy if $\frac{6}{8}$ of it's nearest neighbors are of the similar type while the second population is happy if $\frac{3}{8}$ of its nearest neighbors are of similar type.

2. Produce a plot showing the new allocation of individuals.



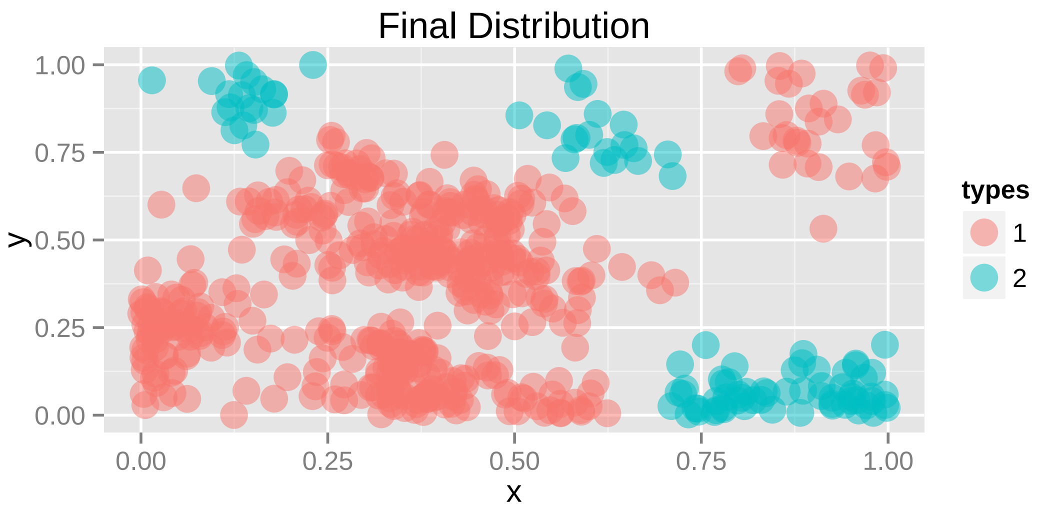
3. What is the value of the similar-neighbor index? (run the function a few times with different random seeds to see how much this varies). Briefly discuss.

```
## [1] "Over 5 simulations:"
## [1] "Similar Neighbor Index, Mean: 0.874 S.D: 5e-04"
## [1] "GINI Index             , Mean: 0.97 S.D: 0.022"
## [1] "Dissimilarity Index    , Mean: 0.873 S.D: 0.056"
```

We seem to observe higher levels of segregation here, which suggest that the aggregate level of segregation may be driven by the group with the stronger prefrences to be nearby similar types.

4. Now let there be 500 of the first population, but only 100 of the second population. Again make a plot showing the final spatial allocation and calculate the similar-neighborhood index. How did this change? Discuss in 2-8 sentences.

```
## [1] "Over 5 simulations:"
## [1] "Similar Neighbor Index, Mean: 0.874 S.D: 0.001"
## [1] "GINI Index            , Mean: 0.985 S.D: 0.021"
## [1] "Dissimilarity Index   , Mean: 0.94 S.D: 0.071"
```

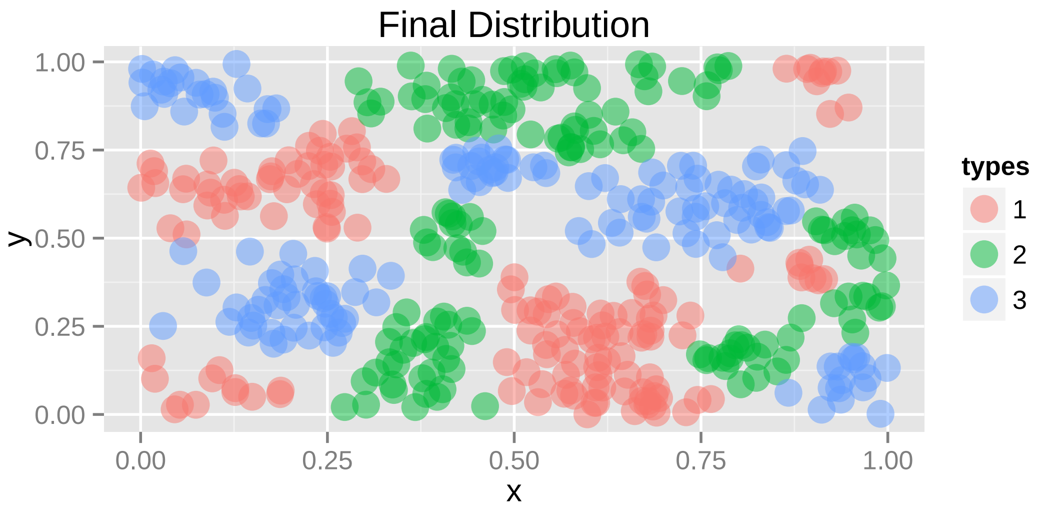


When the sub-population with stronger prefrences is more numeros, inequality measures seem to increase further

Extending the model to three populations Now lets evaluate if the model changes when we introduce a third population.

1. Let each population have 150 individuals. Assume they are happy if $\frac{3}{8}$ of their neighbors are the same as them.

2. Produce a plot showing the new final allocation of individuals.



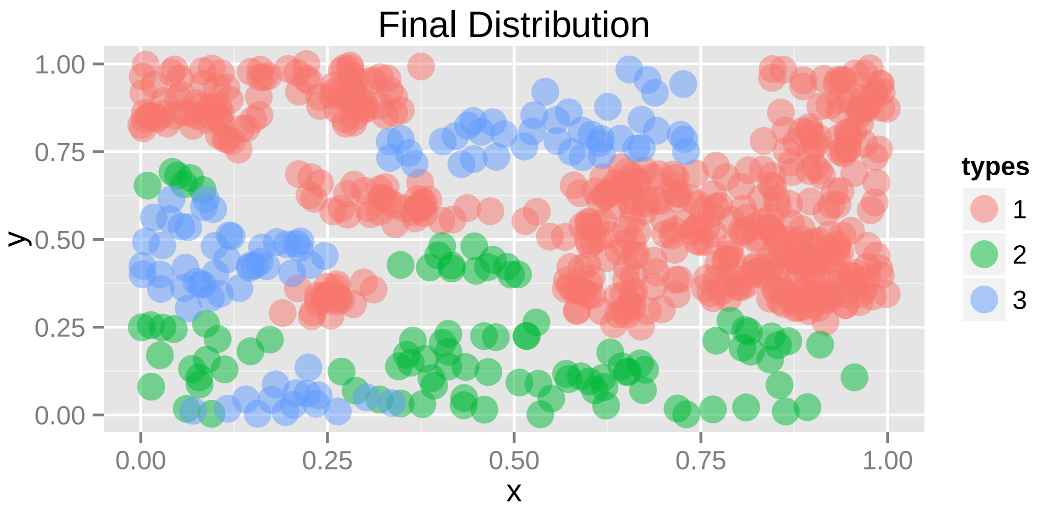
3. What is the similar-neighborhood index? (run the function a few times with different random seeds to see how much this varies). Briefly discuss.

```
## [1] "Similar Neighbor Index over 5 simulations:"
## [1] "Mean: 0.873 S.D: 0.001"
```

Even with a third population, we see measure similar levels of segregation as before. This is perhaps unsurprising since we observe very similar clustering patterns. On the other hand perhaps we would expect to see a lower Similar Neighbor Index because there are fewer individuals of each type.

4. Now let there be 500 of the first population, but only 100 in the second and third population. Let the first population be happy if $\frac{9}{12}$ of it's nearest neighbors are of the similar type while the second and third population are happy is happy if $\frac{2}{12}$ of its nearest neighbors are the same. What sort of model or scenario would this correspond to? Run this model several times and look at the final distribution of results. What is the "take away" of this model which we could use to test or inform our work with real data? Discuss (no more than a page).

```
## [1] "Similar Neighbor Index over 5 simulations:"
## [1] "Mean: 0.834 S.D: 0.007"
```



We measure slightly lower levels of segregation here, but this is perhaps only becuase the two minorities are willing to intermingle. If we aggregated the two minorities into one I suspect you would measure similar levels of inequality as we did in the *Differences between populations* section. NOte that this should tell us that we need to think carefully about how we aggregate subgroups.
You could analogize this simulation to a real world city with say *white, black and latino* populations. However before you did this you would have to stop and think very carefully about what would be the appropriate way to model the prefrences of each group. Are real world cities consistent with the results of this simulation?
It is also worth thinking about how the results of this simulation compare to the Schelling grid simulations we did in class. In this model individuals can live arbitrarily and you will notice that these simulations lead to a lot of hetrogeneity in density within cities. It seems that the majority in these simulaions tend to live on average in higher densities in this simulation in order to avoid the minorities.Is this realistic? How would you measure the density of each subpopulation in this simulation?

Alternative Segregation Indexes Take your functions which calculate the dissimilarity-index and gini-index and compare how these vary compared to the similar-neighborhood index.[2] Start by using the baseline model, but also compare the three indexes for some of the alternative two-population models.[3]

3 Code Review

A surprisingly large part of coding is learning from and incorporating code others have already written. In this problem you will download code for a function which “simulates a peer-effects model”. Its your job to back out how the model works and answer some general questions about the code.

[2]As mentioned above, this portion of the homework will be worth notably fewer points than the other sections, so do this last, and do not worry too much if you are not able to get your alternative functions to work.
[3]This is purposefully unstructured (like most real work). Figure out what is interesting about the differences and discuss.

1. Study the code and give an over-view of how this peer-effects model works. Don't discuss how the code works, but rather, sketch the model I used when implementing this code.

    - Each individual is ordered from $i = 1, \ldots, 200$, she is uniformly randomly placed at a point x_i, y_i in cartesian space on the unit interval $U[0,1] \times U[0,1]$. This individual also receives a preference shock $\epsilon_i \sim U[-0.5, 0.5]$.

    - In the initial plot, each individual chooses to be red if their preference shock $\epsilon > 0$

    - In the middle plot individuals act sequentially from $1, \ldots, 200$.
Of all the individuals who have acted before individual i (i.e $\forall j < i$). Individual i looks at the actions of the twenty nearest to him and calculates the fraction who chose to be red p_{red} [4]. Then individual i chooses to be red if

$$-\frac{1}{2.5} + \frac{2}{2.5}p_{red} + \epsilon_i > 0$$

    - The second plot again iterates sequentially through all of the individuals. However now all individuals consider their 20 closest neighbors (regardless of where their order. This time they make a decision according to a slightly different criteria.

$$-\frac{1}{s} + \frac{2}{s}p_{red} + \epsilon_i > 0$$

2. Are some people more affected by peers than others in this model?
Yes. In two manners

    - Firstly: In the middle plot, the first individual $i = 1$ is not affected by his peers, and subsequent individuals $i \in 2, ldots, 20$ will be influenced by a smaller number of peers than individuals $i \in 21, \ldots, 200$ individuals. This will make the individuals who act first relatively more important.

    - Secondly: Individuals whose prefrence shocks ϵ_i are larger in absolute value will be less influenced by their peers than individuals with ϵ_i closer to zero

3. What do the for-statement on line 42 and the while-statement on line 57 do (1-2 sentences each)?

    - In the for-loop on line 42, we move through all the individuals once and they each make a decision to choose red or not.

    - In the while loop on line 57 is redundant. Since at the end of the loop we set data.old = data the loop will execute once and then be trivially satisfied

4. Why do I output three different plots in the code? What do they each show? Producing all three plots allows us to compare the three different models, with no peer effects, after iterating through with peer effects once, and after iterating through with peer effects a second time.

5. What do k, n, and s do in the code?

    - k: The number of nearby individuals you are influenced by

[4] if $i = 1$ then $p_{red} = 0$, if $i < 20$ then individual i looks at everybody who came before him regardless of distance

- n: The total number of individuals
- $s \in (0, \infty)$: Is a sensitivity parameter which determines the size of the peer effects, the larger s the smaller the peer effects are relative to the individual shocks

6. How does changing s and k affect the code (1-3 sentences)?
   Changing s will decrease individual's sensitivity to the peer effects, this will result in a more random (less segregated) distribution of types.
   Changing k will increase the number of people each individual is influenced by. It is not clear what impact this will have on segregation ex ante. Very large k's will lead to large unifom areas of influence and so little segregation, but very small k's will lead to small areas of ifluence and so this may not lead to much clustering.

7. Reuse you similar-neighbor index function from above to calculate the degree of segregation in the baseline model I run in my .R file. How does this model's level of segregation compare to our baseline Schelling model?

```
## [1] "Over 10 simulations:"
## [1] "Middle Plot Similar Neighbor Index, mean: 0.605 , sd: 0.074"
## [1] "Final Plot Similar Neighbor Index, mean: 0.573 , sd: 0.061"
```

Research

- Suggest one research idea based around the models we considered in this homework (no more than 3 sentences).
- List three topics you may be interested in doing research on. These can be broad, such as "The returns to community college", or very narrow, such as "Advertising for for the Xbox One and the PlayStation 4". As undergraduates, its helpful to stick to things that really interest you or things you really care about (anything going on in your home state?).[5]
- If you are a 4th year who wrote a BA which you want to extend for this course, write a 3 sentence summary of your BA, then write 1-5 sentences on how you may extend it for this course.[6]

Side Projects

1. Download census tract data on race for the city of Chicago. Calculate the gini and dissimilarity indexes using your function. If possible make a map in R of these results. Write up no more than 1 page (not including figures) discussing your results (3 points)

[5]For example, as an undergraduate I could have maybe have tried to write a hedonic pricing model for resell of high-end acoustic guitars (maybe scraping the data off of sales sites), or I could have written a paper on how Alaska's economy is counter-cyclical with rest of the Nation and the resulting impact of United States' monetary policy on Alaska's economy.

[6]Broad ideas are fine. If you don't have a good answer, come talk to us at office hours sometime in the next week or two

2. Rewrite some or all of the code for this problem in Julia or C++ with Rcpp[7] (up to 3 points)

3. write a wiki entry with the examples for any of the following R commands: "order()", "which()", "apply()", "sapply()/lapply()" (u p to 1 point, no more than 1 per person).

4. write a wiki entry about how to time your code (up to 1 point).

5. write a wiki entry on ggplot2 (and the wrapper qplot). This is an extremely powerful plotting tool. Please provide examples and their corresponding plots. (up to 1 point, but multiple people can contribute and extend with different examples, different plot types, etc)

6. Build a shiny app (http://www.rstudio.com/shiny/) which lets users configure and run the Schelling model (or the grid Schelling Model) using a graphical user interface (up to 3 points).

7. Use knitr to add animated plots to your pdfs (animations only work in newer versions of Adobe products) (1.5 points)

8. Read the Thomas Schelling's 1971 and/or 1969 paper and write a short (2-5 page) summary and review of his paper(s). If you read both papers, discuss the difference between the two (up to 3 points). (http://www.tandfonline.com/doi/pdf/10.1080/0022250X.1971.9989794) (http://www.jstor.org/stable/pdfplus/1823701.pdf)

Useful Resources

www.census.gov/hhes/www/housing/housing_patterns/pdf/app_b.pdf

[7]If you tackle Rcpp send me an email and I can try to help you get started. You will most likely want to use RcppArmadillo, which links to the fantastic Armadillo C++ library which has syntax somewhat similar to matlab