Actividad 3: Simulación de la geometría de ciudad y estructura de datos para las personas

E. Crescio, J. Montalvo, E. Uresti

2024-11-13

En esta fase del reto, iniciarás un esquema en el que los individuos considerados son personas que se encuentran localizados en un espacio geográfico delimitado en una geometría conocida.

Para este fin, cada miembro del equipo deberá elegir una de las siguientes configuraciones de su problema:

Problema 1:

- Crear una ciudad cuadrada donde cada lado tiene tamaño D con distribución uniforme de personas.
- Crear un arreglo de posiciones x, y de posiciones y considerando N personas. Asignar una posición inicial para cada una de las personas de la población por medio de un número aleatorio uniforme. En `R` puede realizarse como: `runif(N,min=0,max=1)`. Note que debe elegir valores apropiados para el mínimo y el máximo para asegurar que todas las personas estén dentro de los límites de la ciudad.

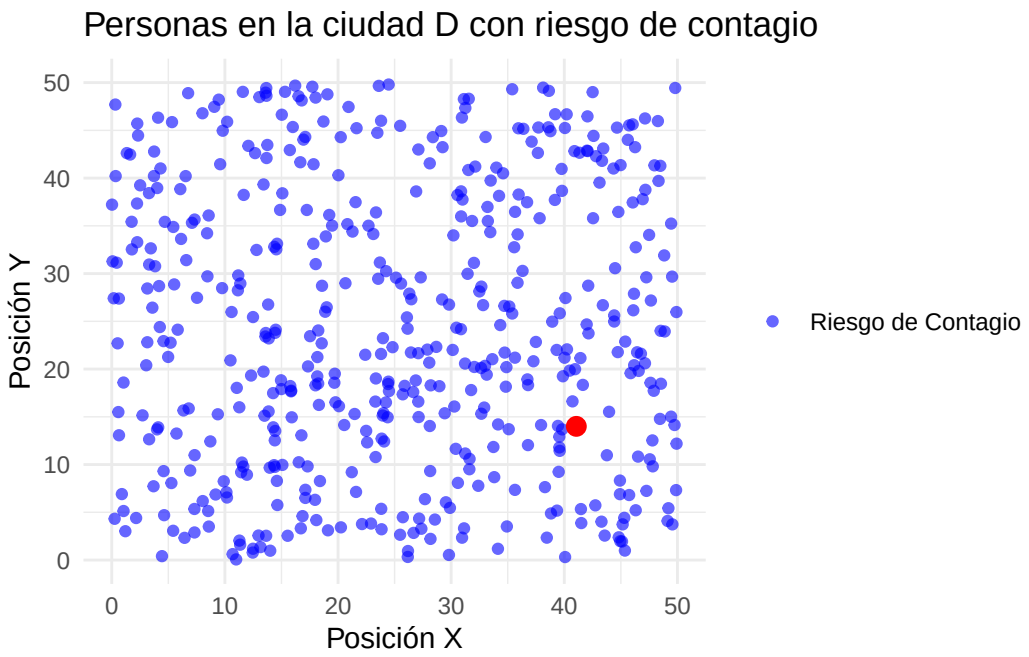
```
# PACKAGES:
library(ggplot2)

D <- 50             #tamaño de la ciudad
N <- 100000         #población total

# estados de la población
I <- 1              #infectados iniciales
R <- 0              #recuperados iniciales
S <- N - I - R      #susceptibles iniciales

r <- 0.6            #radio de infección
```

```r
gamma <- 0.1       # azón de recuperación

#posiciones de las personas en la ciudad
posicion_x <- runif(N, min = 0, max = D)
posicion_y <- runif(N, min = 0, max = D)

#asignar el estado (I, R, S) a cada persona
estado <- factor(c(rep("Infectado", I), rep("Recuperado", R), rep("Susceptible", S)),
                 levels = c("Susceptible", "Infectado", "Recuperado"))

id <- 1:N
iteracion <- rep(0, N)

#dataframe con todas las variables
poblacion <- data.frame(id, posicion_x, posicion_y, estado, iteracion)


#función para calcular la distancia euclidiana y verificar si está dentro r
calcular_infectados <- function(df, r) {
  #seleccionar una persona aleatoria como infectada
  set.seed(1)
  infectado <- sample(which(df$estado == "Infectado"), 1)

  #calcular distancia
  df$riesgo_contagio <- sqrt((df$posicion_x - df$posicion_x[infectado])^2 +
                             (df$posicion_y - df$posicion_y[infectado])^2) < r

  return(list(data = df, infectado = infectado))
}

#aplicar la función y obtener el resultado
resultado <- calcular_infectados(poblacion, r)
poblacion <- resultado$data
infectado <- resultado$infectado
#df
print(head(poblacion))
```

```
  id posicion_x posicion_y      estado iteracion riesgo_contagio
1  1  41.063797  14.000326   Infectado         0            TRUE
2  2  16.254107   3.995288 Susceptible         0           FALSE
3  3   1.814612  24.288803 Susceptible         0           FALSE
4  4  44.233772  13.180601 Susceptible         0           FALSE
```

```
5  5  23.018619  16.832463 Susceptible         0           FALSE
6  6  39.714729   1.975882 Susceptible         0           FALSE
```

```r
#gráfica
poblacion_sample <- poblacion[sample(1:N, 500), ]  # muestra de 500 personas

ggplot(poblacion_sample, aes(x = posicion_x, y = posicion_y)) +
  geom_point(aes(color = riesgo_contagio), alpha = 0.6) +
  geom_point(data = poblacion[infectado, ], aes(x = posicion_x, y = posicion_y), color = "red
  labs(title = "Personas en la ciudad D con riesgo de contagio", x = "Posición X", y = "Posi
  xlim(0, D) +
  ylim(0, D) +
  theme_minimal() +
  scale_color_manual(values = c("TRUE" = "orange", "FALSE" = "blue"), labels = c("Riesgo de C
  theme(legend.title = element_blank())
```



Problema 2:

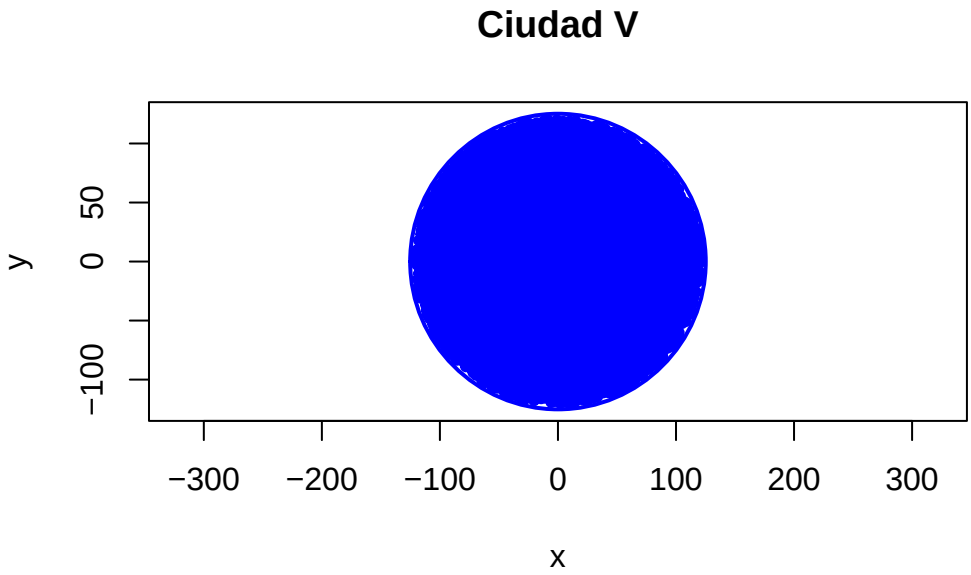
- Crear una ciudad circular de radio $D/2$ con distribución uniforme de personas, misma que no puede generarse de manera estándar sino que debe considerarse una distribución modificada tal como se ve en https://programming.guide/random-point-within-circle.html.

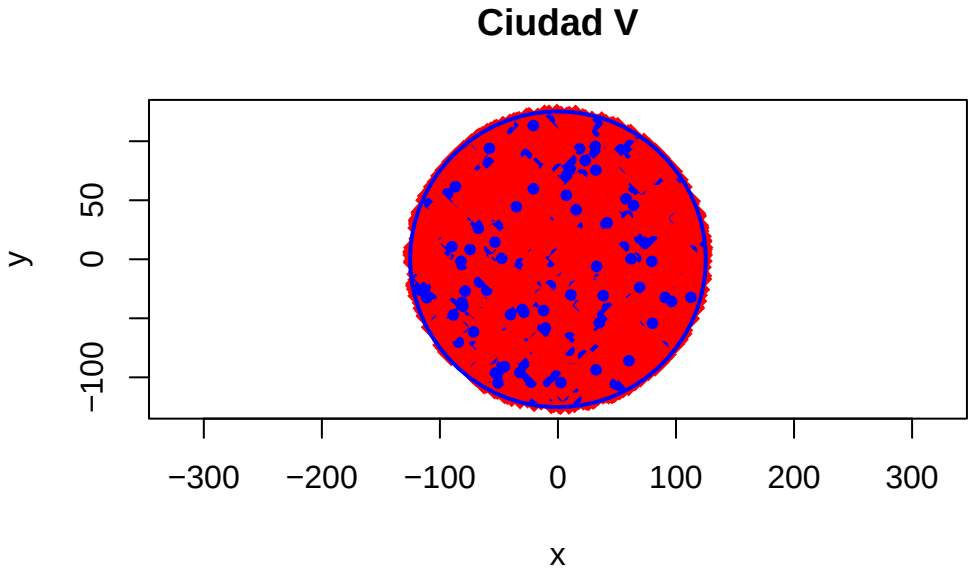
- Crear un arreglo de posiciones x, y de posiciones y considerando N personas. Asignar una posición inicial aleatoria para cada una de las personas de la población.

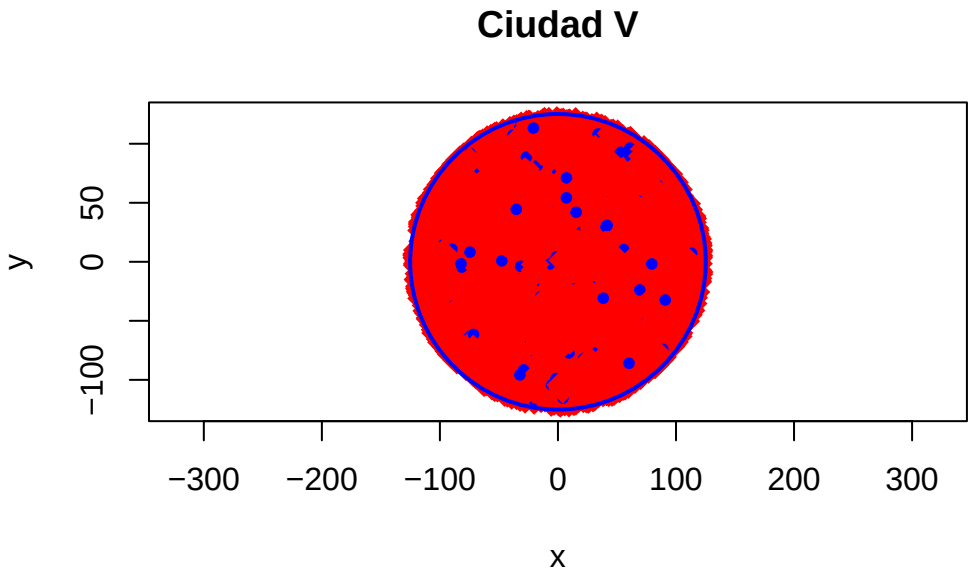
```r
# Parámetros de la simulación
I <- 1  # Número inicial de infectados
S <- 9999  # Número inicial de susceptibles
Rec <- 0  # Número inicial de recuperados
N <- I + S + Rec  # Tamaño total de la población
D <- 250  # Diámetro de la ciudad
R <- D / 2  # Radio de la ciudad
ajuste_borde <- 0.96  # Ajuste para evitar que personas estén en el borde exacto de la ciuda
k <- 0.15  # Factor de radio inicial
radius <- N * k  # Radio de infección inicial
recov <- 0.2  # Probabilidad de recuperación (sin uso en esta simulación)
num_iteraciones <- 8  # Número de iteraciones
prob_infección <- 0.6  # Probabilidad de infección

# Generación de posiciones usando coordenadas polares
ang <- runif(N, 0, 2 * pi)
rad <- R * ajuste_borde * sqrt(runif(N))
x_N <- rad * cos(ang)
y_N <- rad * sin(ang)

# Creación del dataframe personas
personas <- data.frame(
  id = 1:N,
  posición_x = x_N,
  posición_y = y_N,
  Estado = sample(c(rep("Susceptible", S), rep("Infectado", I), rep("Recuperado", Rec))),
  it = 0
)

# Función para obtener coordenadas de una persona por su ID
coords <- function(id_persona) {
  persona <- personas[personas$id == id_persona, ]
  if (nrow(persona) > 0) {
    return(c(x = persona$posición_x, y = persona$posición_y))
  } else {
    return("ID no encontrado")
  }
}

# Función para calcular la distancia euclidiana entre dos puntos
```

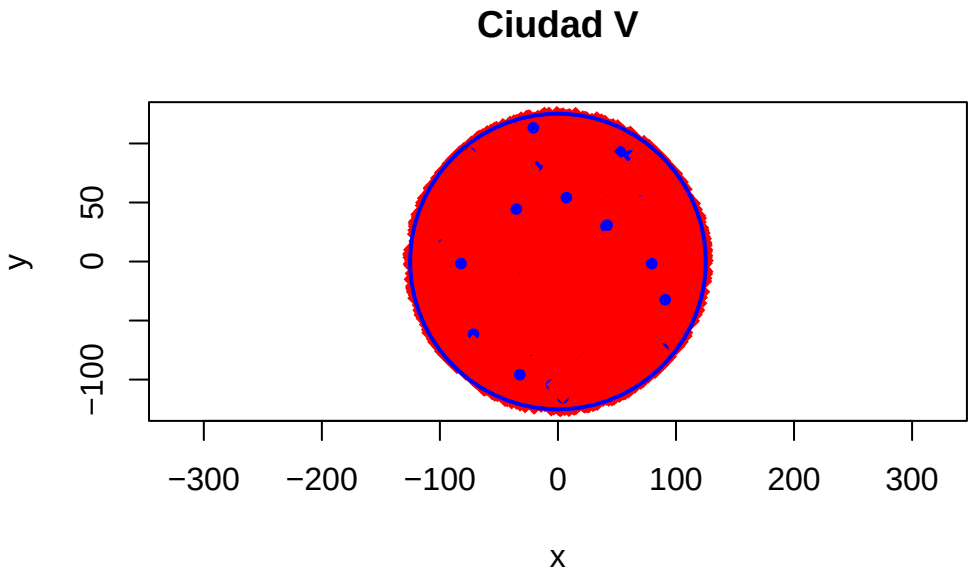
```r
euclid <- function(x1, x2, y1, y2) {
  sqrt((x2 - x1)^2 + (y2 - y1)^2)
}

# Función para verificar si una persona está dentro del radio de infección
infección <- function(x1, y1, x2, y2, radius) {
  euclid(x1, x2, y1, y2) < radius
}

# Función para graficar la situación de la ciudad en cada iteración
graficar <- function(personas, infectado_inicial, R) {
  plot(personas$posición_x, personas$posición_y,
       xlim = c(-R, R), ylim = c(-R, R),
       xlab = "x", ylab = "y",
       main = "Ciudad V",
       pch = ifelse(personas$Estado == "Susceptible", 20,
                    ifelse(personas$Estado == "Infectado", 18, 19)),
       col = ifelse(personas$Estado == "Susceptible", "blue",
                    ifelse(personas$Estado == "Infectado", "red", "green")),
       cex = ifelse(personas$Estado == "Susceptible", 1,
                    ifelse(personas$Estado == "Infectado", 2, 2)),
       asp = 1)
  # Graficar una frontera de la ciudad (un círculo como límite)
  symbols(0, 0, circles = R + 0.2, inches = FALSE, add = TRUE, lwd = 2, fg = "blue")
}

# Selección del infectado inicial
infectado_inicial <- which(personas$Estado == "Infectado")[1]

# Iteración de la infección con probabilidad
for (j in 1:num_iteraciones) {
  graficar(personas, infectado_inicial, R)
  for (i in 1:N) {
    if (personas$Estado[i] == "Susceptible") {
      infected_coords <- coords(infectado_inicial)
      normal_coords <- coords(i)
      distance <- euclid(infected_coords["x"], normal_coords["x"], infected_coords["y"], nor
      if (infección(infected_coords["x"], infected_coords["y"], normal_coords["x"], normal_c
          runif(1) < prob_infección) {
        personas$Estado[i] <- "Infectado"
      }
    }
```

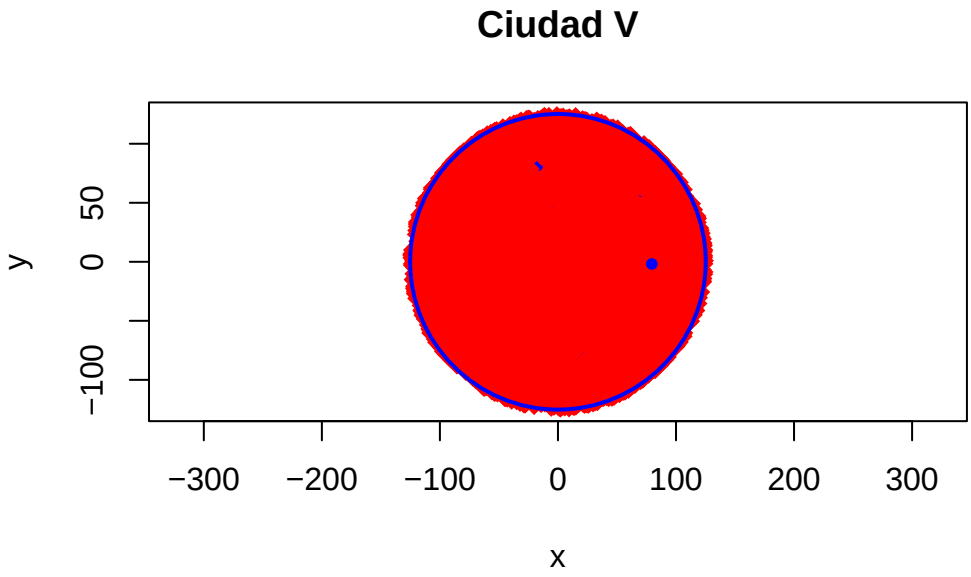
```
  }
  Sys.sleep(1)
}
```

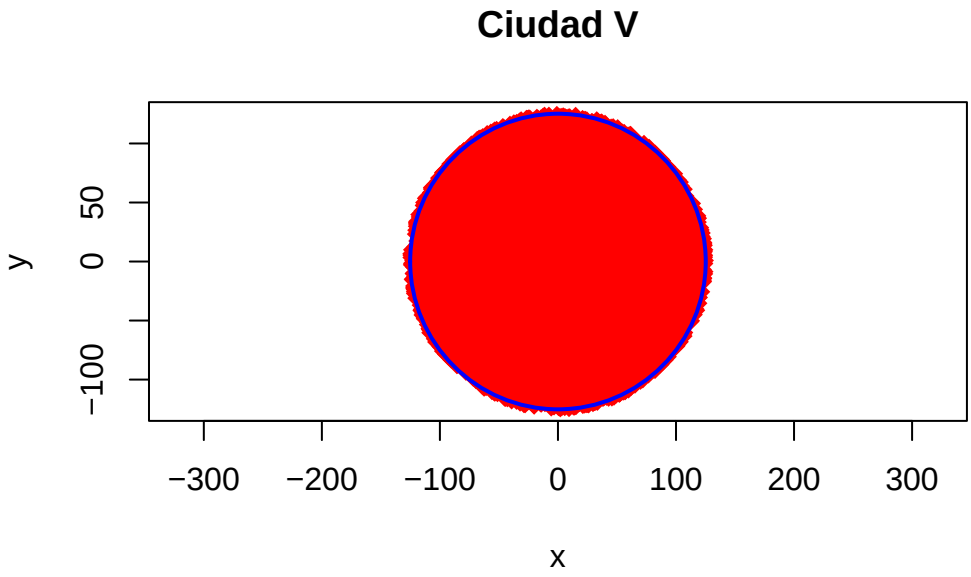

Ciudad V
y
50
0
−100
−300
−200
−100
0
100
200
300
x

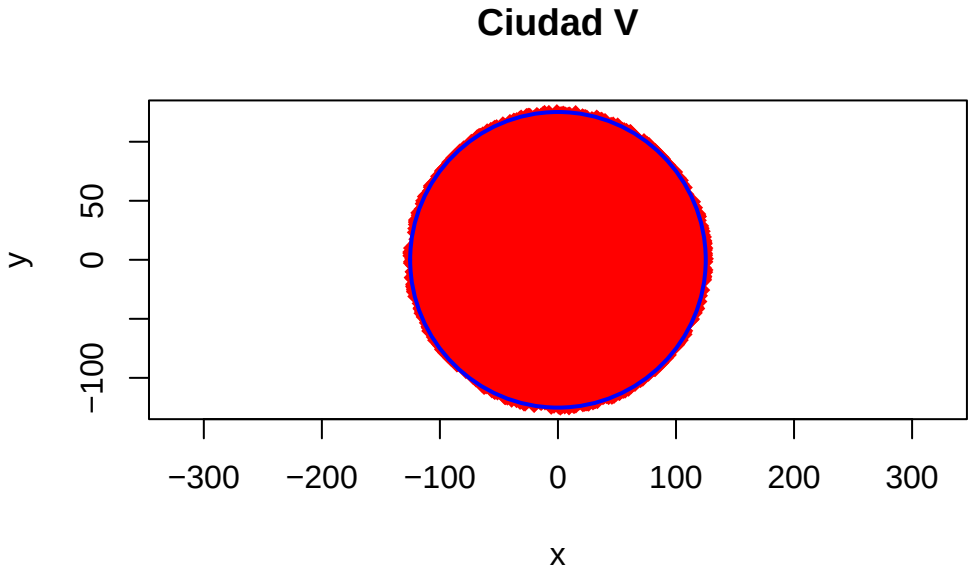

Ciudad V
y
50
0
−100
−300
−200
−100
0
100
200
300
x

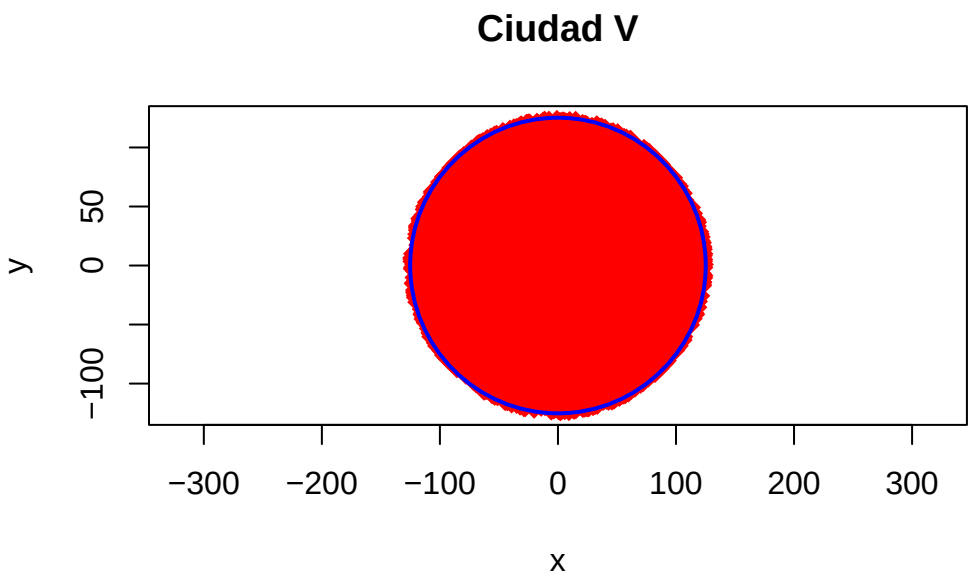

Ciudad V
y
50
0
−100
−300
−200
−100
0
100
200
300
x


Ciudad V
y
50
0
−100
−300
−200
−100
0
100
200
300
x


Ciudad V
y
50
0
−100
−300
−200
−100
0
100
200
300
x


Ciudad V
y
50
0
−100
−300
−200
−100
0
100
200
300
x





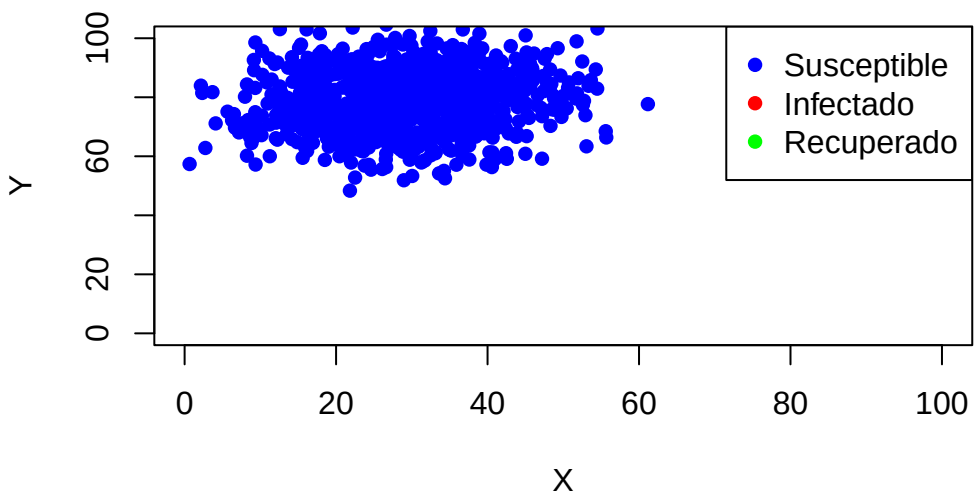
Problema 3:

- Crear una ciudad cuadrada de lado D en la que las personas están distribuidas en forma de "cluster" en donde hay una preferencia de las personas para estar ubicadas en cierta zona dentro de la ciudad. Para esto, defina un lugar de preferencia en forma aleatoria (x_0, y_0) y determine la posición aleatoria de la posición de N personas distribuidas de acuerdo a una distribución normal.
- Considere la función de la distribución normal (en `R` la función para generar números normalmente distribuidos con media 0 y desviación estándar 1 es `rnorm(N,mean=0,sd=1)`).

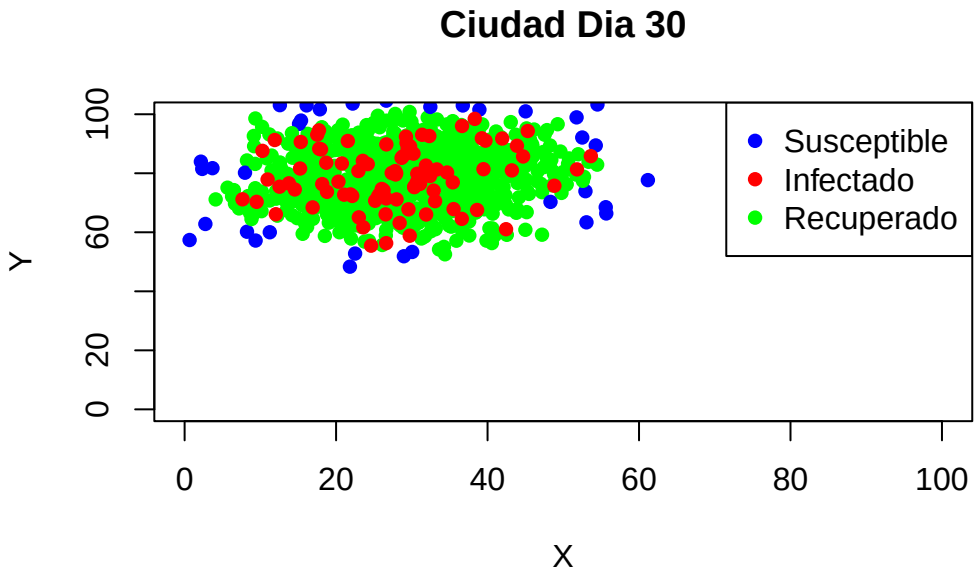
Con esto, puede elegir coordenadas en x que se concentran alrededor de x_0 y coordenadas y que se concentran alrededor de y_0. Considere una desviación estándar de tamaño $D/20$.

```
# PACKAGES:
library(reshape2)
library(ggplot2)

# Parametros
D <- 100
sigma <- D / 10

N <- 1000      # Total de personas
S <- N - 1    # susceptibles
I <- 1        # infectados
R <- 0        # recuperados

gamma <- 0.1  # razón de recuperación
r <- 3        # radio de infección

# Cluster
set.seed(123)  # para empezar el con el mismo punto (quitar para aleatoridad)
x0 <- runif(1, min = 0, max = D)
y0 <- runif(1, min = 0, max = D)

# Generador de posiciones alrededor del cluster
x <- rnorm(N, mean = x0, sd = sigma)
y <- rnorm(N, mean = y0, sd = sigma)

# Estado de cada persona
estado <- factor(c(rep("Infectado", I), rep("Susceptible", S), rep("Recuperado", R)),
                 levels = c("Susceptible", "Infectado", "Recuperado"))

# Distancia euclidiana entre dos puntos
distancia <- function(x1, y1, x2, y2) {sqrt((x2 - x1)^2 + (y2 - y1)^2)}

# DataFrame
ciudad <- data.frame(id = 1:N,
                     x = x,
                     y = y,
                     estado = estado,
                     iteracion = 0)

# Actualizar DataFrame
```

```r
actualizar_ciudad <- function(ciudad, r, gamma) {

  nueva_ciudad <- ciudad

  I <- which(ciudad$estado == "Infectado")
  n_I <- length(I)

  n_R <- floor(gamma * n_I)

  if (n_R > 0) {
    nueva_ciudad$estado[I[1:n_R]] <- "Recuperado"
  }

  for (i in I) {
    for (j in which(nueva_ciudad$estado == "Susceptible")) {
      if (distancia(nueva_ciudad$x[i], nueva_ciudad$y[i], nueva_ciudad$x[j], nueva_ciudad$y[
        nueva_ciudad$estado[j] <- "Infectado"
      }
    }
  }

  return(nueva_ciudad)
}

# Ciudad inicial
plot(x, y, pch = 16, col = ifelse(ciudad$estado == "Susceptible", "blue",
                                  ifelse(ciudad$estado == "Infectado", "red",
    xlim = c(0, D), ylim = c(0, D),
    xlab = "X", ylab = "Y")
    title(main = "Ciudad Dia 1")
    legend("topright", legend = c("Susceptible", "Infectado", "Recuperado"),
    col = c("blue", "red", "green"), pch = 16)
```




```r
# Simulación de múltiples iteraciones
num_iteraciones <- 30
historial <- data.frame()

for (t in 1:num_iteraciones) {
  ciudad$iteracion <- t
  historial <- rbind(historial, ciudad)

  ciudad <- actualizar_ciudad(ciudad, r, gamma)
}

# Ciudad Final
plot(ciudad$x, ciudad$y, pch = 16, col = ifelse(ciudad$estado == "Susceptible", "blue",
                                                ifelse(ciudad$estado == "Infectado", "red",
    xlim = c(0, D), ylim = c(0, D),
    xlab = "X", ylab = "Y")
    title(main = "Ciudad Dia 30")
    legend("topright", legend = c("Susceptible", "Infectado", "Recuperado"),
    col = c("blue", "red", "green"), pch = 16)
```




```
# Visualizar el historial
head(historial,100)
```

```
    id         x         y      estado iteracion
1    1 26.455977  68.43096   Infectado         1
2    2 44.344835  78.65071 Susceptible         1
3    3 29.462836  77.50876 Susceptible         1
4    4 30.050629  53.33709 Susceptible         1
5    5 45.908402  89.23625 Susceptible         1
6    6 33.366914  81.32777 Susceptible         1
7    7 16.107140 102.99259 Susceptible         1
8    8 21.889223  85.68250 Susceptible         1
9    9 24.301132  74.36092 Susceptible         1
10  10 40.998570 106.80443 Susceptible         1
11  11 32.355890 107.15277 Susceptible         1
12  12 32.765467  66.64340 Susceptible         1
13  13 29.864579  83.52083 Susceptible         1
14  14 23.199341  76.71804 Susceptible         1
15  15 46.626883  80.70103 Susceptible         1
16  16 33.736257  81.10594 Susceptible         1
17  17  9.091580  66.21151 Susceptible         1
18  18 35.771311  81.68641 Susceptible         1
19  19 24.029838  96.32299 Susceptible         1
20  20 18.079515  77.18961 Susceptible         1
21  21 26.578003  77.20125 Susceptible         1
22  22 18.497708  92.81623 Susceptible         1
```

23	23	21.468840	87.81448	Susceptible	1
24	24	22.507359	62.34557	Susceptible	1
25	25	11.890819	81.11608	Susceptible	1
26	26	37.135622	95.36599	Susceptible	1
27	27	30.291483	92.98328	Susceptible	1
28	28	17.376383	83.03003	Susceptible	1
29	29	41.295901	86.04272	Susceptible	1
30	30	33.022394	66.86116	Susceptible	1
31	31	25.807037	81.83183	Susceptible	1
32	32	37.709009	69.28602	Susceptible	1
33	33	37.539087	74.25033	Susceptible	1
34	34	36.973563	88.18655	Susceptible	1
35	35	35.644155	67.46158	Susceptible	1
36	36	34.296929	81.49970	Susceptible	1
37	37	28.138635	83.11383	Susceptible	1
38	38	25.698125	79.37963	Susceptible	1
39	39	24.953042	97.05240	Susceptible	1
40	40	21.810682	68.60704	Susceptible	1
41	41	26.678579	84.89182	Susceptible	1
42	42	16.103788	77.94121	Susceptible	1
43	43	50.447312	76.22219	Susceptible	1
44	44	40.837372	83.47143	Susceptible	1
45	45	17.526666	68.62651	Susceptible	1
46	46	24.728904	65.69600	Susceptible	1
47	47	24.091198	73.88570	Susceptible	1
48	48	36.557403	96.34809	Susceptible	1
49	49	27.924061	79.38816	Susceptible	1
50	50	31.290937	82.14486	Susceptible	1
51	51	28.472284	76.93205	Susceptible	1
52	52	28.329047	83.53544	Susceptible	1
53	53	42.443775	69.31372	Susceptible	1
54	54	26.500042	90.40962	Susceptible	1
55	55	43.922458	84.67757	Susceptible	1
56	56	13.270224	70.76599	Susceptible	1
57	57	34.603890	79.37605	Susceptible	1
58	58	29.996294	85.99383	Susceptible	1
59	59	30.917168	84.40782	Susceptible	1
60	60	32.554147	93.64985	Susceptible	1
61	61	23.734517	72.70064	Susceptible	1
62	62	25.425678	89.99188	Susceptible	1
63	63	18.571998	89.19599	Susceptible	1
64	64	18.039840	77.20568	Susceptible	1
65	65	31.793038	69.07125	Susceptible	1

```
66   66 33.239850  67.93906 Susceptible         1
67   67 29.287794  83.40838 Susceptible         1
68   68 37.980427  78.11925 Susceptible         1
69   69 49.258599  96.62154 Susceptible         1
70   70 23.847440  84.18189 Susceptible         1
71   71  5.666063  75.11106 Susceptible         1
72   72 38.815137  68.57509 Susceptible         1
73   73 21.665744  73.00650 Susceptible         1
74   74 21.877666  82.25940 Susceptible         1
75   75 39.013466  74.32117 Susceptible         1
76   76 25.910022  83.97281 Susceptible         1
77   77 16.550575  75.48713 Susceptible         1
78   78 30.570787  77.77491 Susceptible         1
79   79 27.368838  71.52542 Susceptible         1
80   80 28.815394  97.88095 Susceptible         1
81   81 32.610556  82.15673 Susceptible         1
82   82 25.051152  81.13685 Susceptible         1
83   83 35.201517  61.91189 Susceptible         1
84   84 26.552886  85.42843 Susceptible         1
85   85 32.075572  68.59428 Susceptible         1
86   86 39.726142  69.91530 Susceptible         1
87   87 33.109567  88.01393 Susceptible         1
88   88 25.498436  74.30351 Susceptible         1
89   89 40.245828  61.34679 Susceptible         1
90   90 38.692791  96.52955 Susceptible         1
91   91 34.241722  55.05644 Susceptible         1
92   92 31.145069  84.55863 Susceptible         1
93   93 22.478691  89.00301 Susceptible         1
94   94 42.364276  72.52083 Susceptible         1
95   95 22.755156  83.27338 Susceptible         1
96   96 50.631082  83.22182 Susceptible         1
97   97 44.083858  89.23675 Susceptible         1
98   98 26.400748  83.67151 Susceptible         1
99   99 18.493543  76.38168 Susceptible         1
100 100 21.653686  87.99043 Susceptible         1
```

Problema 4:

- Crear una ciudad circular de radio $D/2$ con distribución de personas en "cluster". Considere la generación de números aleatorios en un círculo de acuerdo a https://programming.guide/random-point-within-circle.html.

- Considere un arreglo para posiciones x, y otro para posiciones y correspondientes a N personas que se concentran de acuerdo a una distribución normal en un ángulo y distancia al centro seleccionados de manera aleatoria.

Generación de I, S, R, además de sus coordenadas polares

```
# Definimos el radio del círculo
I <- 1
S <- 999
Rec <- 0
N <- I + S + Rec
R <- 75
ajuste <- 0.98 # Agrego esto para que ningún punto esté justo en el límite de la ciudad
N_central <- floor(N*0.66)
N_resto <- floor(N*0.34)
# centro cluster
x0 <- runif(1, -R * 0.1, R * 0.1)
y0 <- runif(1, -R * 0.1, R * 0.1)

# los del cluster, distribuidos normal
x_N_central <- rnorm(N_central, mean = x0, sd = R * 0.15)
y_N_central <- rnorm(N_central, mean = y0, sd = R * 0.15)

# el resto, uniformes
ang_resto <- runif(N_resto, 0, 2 * pi)
rad_resto <- R * ajuste * sqrt(runif(N_resto))
x_N_resto <- rad_resto * cos(ang_resto)
y_N_resto <- rad_resto * sin(ang_resto)

#combinamos los N de cluster y N de fuera
x_N <- c(x_N_central, x_N_resto)
y_N <- c(y_N_central, y_N_resto)
```

Definir radio de infección y Razón de recuperación

```
radius <- 12
recov <- 0.2 # Radio de infección y Razón de recuperación
inicio_recov <- 3
```

Variables categóricas

```
estados <- factor(c("Susceptible", "Infectado", "Recuperado"))
estado_inicial <- c(rep("Susceptible", S), rep("Infectado", I), rep("Recuperado", Rec))
estado_inicial <- sample(estado_inicial) #
```

Dataframe para saber Datos sobre cada Persona

```
personas <- data.frame(
  id = 1:N, posición_x = x_N, posición_y = y_N, Estado = factor(estado_inicial, levels = esta
)
#Dataframe inicial que genera el id único, estado, posiciones x,y y la iteración
infectado_inicial <- which(personas$Estado == "Infectado")[1]
#Función para obtener coordenadas
coords <- function(id_persona) {
  persona <- personas[personas$id == id_persona, ]

  if (nrow(persona) > 0) {
    return(c(x = persona$posición_x, y = persona$posición_y))
  } else {
    return("ID no encontrado bro")
  }
}
```

Calcular la distancia euclidiana

```
euclid <- function(x1,x2,y1,y2) {
  sqrt((x2-x1)^2+(y2-y1)^2)
}
```

Saber si está dentro del radio de infección

```
infección <- function(x1,x2,y1,y2,radius){
  euclid(x1,x2,y1,y2) < radius
}
#Función que revisa si la distancia entre dos personas es dentro del radio de infección
```

Inicio (Se asigna un infectado al azar para comenzar)

```
#Función gigante para poder graficar siempre que quiera
graficar <- function(personas, infectado_inicial, R){
  # Graficar todos los puntos
  plot(personas$posición_x, personas$posición_y,
     xlim = c(-R, R), ylim = c(-R, R),
     xlab = "x", ylab = "y",
     main = "Ciudad V",
     pch = ifelse(personas$Estado == "Susceptible", 20,
                  ifelse(personas$Estado == "Infectado", 18, 19)),
     col = ifelse(personas$Estado == "Susceptible", "blue",
                  ifelse(personas$Estado == "Infectado", "red", "green")),
     cex = ifelse(personas$Estado == "Susceptible", 1,
                  ifelse(personas$Estado == "Infectado", 2, 2)),
     asp = 1)
  symbols(0, 0, circles = R+0.2, inches = FALSE, add = TRUE, lwd = 2, fg = "blue")
}
```

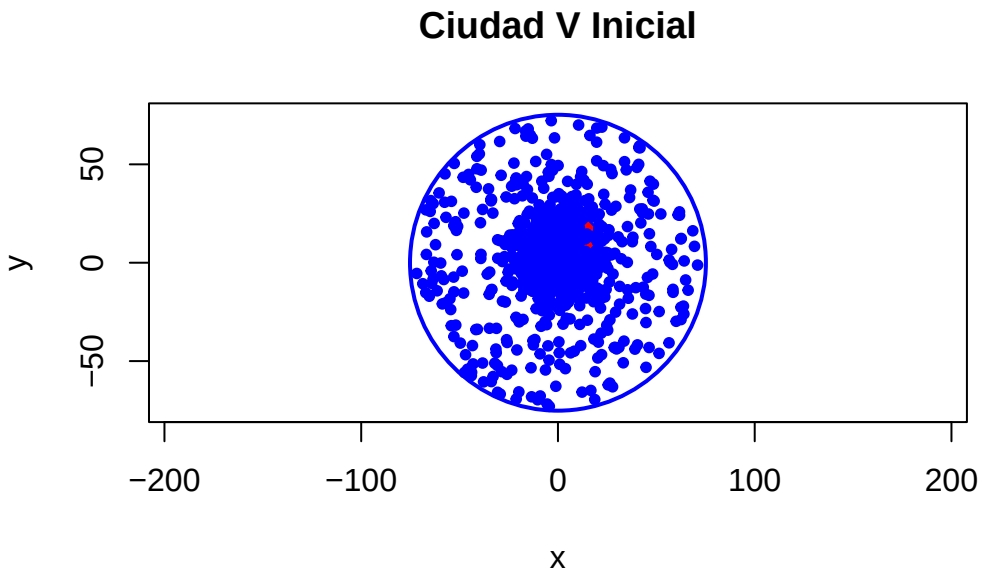
Este sería el dataframe inicial, en los que hay N personas, 999 Susceptibles y 1 Infectado, además de 0 recuperados. (Se toman 15 aleatorios, pero para mostrar el formato)

```
print(personas[sample(nrow(personas), 15), ])
```

```
     id posición_x  posición_y      Estado it
938 938 -59.597980  -0.1659878 Susceptible  0
774 774  22.305143  68.9087469 Susceptible  0
919 919  26.416535 -29.0132451 Susceptible  0
413 413 -21.240061   9.1376405 Susceptible  0
866 866  -5.470940 -71.8190620 Susceptible  0
509 509  24.376712   2.6385457 Susceptible  0
722 722  18.904140 -55.3769532 Susceptible  0
841 841  18.700433 -69.5345793 Susceptible  0
702 702 -71.767475  -5.4565763 Susceptible  0
367 367  14.293626   5.2740469 Susceptible  0
964 964 -47.937949 -15.4232365 Susceptible  0
330 330   5.123988 -20.1855708 Susceptible  0
62   62   5.332227   9.1323735 Susceptible  0
945 945  37.818639  10.6609253 Susceptible  0
770 770 -58.039162  -8.5400280 Susceptible  0
```

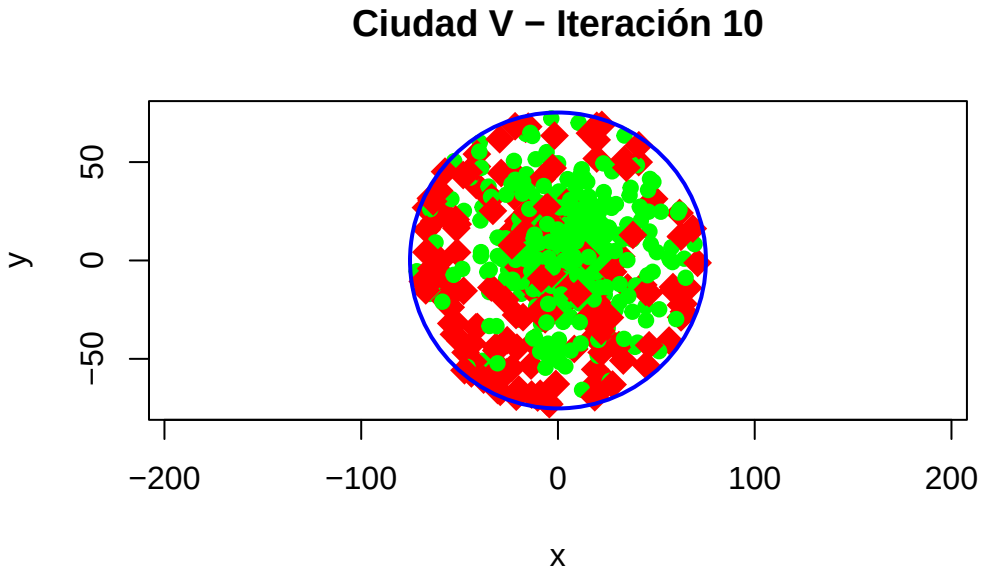
Infección e iteraciones

```
# Infección e iteraciones
 plot(personas$posición_x, personas$posición_y,
         xlim = c(-R, R), ylim = c(-R, R),
         xlab = "x", ylab = "y",
         main = paste("Ciudad V Inicial"),
         pch = ifelse(personas$Estado == "Susceptible", 20,
                      ifelse(personas$Estado == "Infectado", 18, 19)),
         col = ifelse(personas$Estado == "Susceptible", "blue",
                      ifelse(personas$Estado == "Infectado", "red", "green")),
         cex = ifelse(personas$Estado == "Susceptible", 1,
                      ifelse(personas$Estado == "Infectado", 2, 2)),
         asp = 1)
    # Graficar una frontera de la ciudad (un círculito vaya, para que sea el límite)
    symbols(0, 0, circles = R + 0.2, inches = FALSE, add = TRUE, lwd = 2, fg = "blue")
```




```
for (iter in 1:10) {
  cambios <- FALSE  # Indicador de cambios
  personas_temp <- personas  # un df temporal para ir checando

  # Susceptibles iterados
  for (i in which(personas$Estado == "Susceptible")) {
    for (k in which(personas$Estado == "Infectado")) {
      normal_coords <- coords(i)
```

```
      infected_coords <- coords(k)

      # Dentro del radio de infección?
      if (infección(infected_coords["x"], normal_coords["x"], infected_coords["y"], normal_co
        personas_temp$Estado[i] <- "Infectado"  # Cambiamos el estado a infectado
        cambios <- TRUE
        break
      }
    }
  }

  for (i in which(personas$Estado == "Infectado")) {
    if (iter >= inicio_recov) {  # Solo después de la iteración de recov
      if (runif(1) < recov) {  # Comprobamos la probabilidad de recuperación
        personas_temp$Estado[i] <- "Recuperado"  # La persona se recupera
        cambios <- TRUE
      }
    }
  }

  # Si hubo cambios, actualizamos el personas
  if (cambios) {
    personas <- personas_temp

  } else {
    print("No hubo cambios en esta iteración.")
  }
}
    # Graficar el estado de la ciudad en esta iteración
    plot(personas$posición_x, personas$posición_y,
         xlim = c(-R, R), ylim = c(-R, R),
         xlab = "x", ylab = "y",
         main = paste("Ciudad V - Iteración", iter), # Título con número de iteración
         pch = ifelse(personas$Estado == "Susceptible", 20,
                      ifelse(personas$Estado == "Infectado", 18, 19)),
         col = ifelse(personas$Estado == "Susceptible", "blue",
                      ifelse(personas$Estado == "Infectado", "red", "green")),
         cex = ifelse(personas$Estado == "Susceptible", 1,
                      ifelse(personas$Estado == "Infectado", 2, 1)),
         asp = 1)
    symbols(0, 0, circles = R + 0.2, inches = FALSE, add = TRUE, lwd = 2, fg = "blue")
```




```r
print(personas[sample(nrow(personas), 15), ]) #Imprimir 15 personas aleatorias
```

```
     id  posición_x posición_y     Estado it
581 581   7.7414133  7.0289384 Recuperado  0
518 518   9.1826121 34.4625328 Recuperado  0
379 379 -10.2222899  4.7150034 Recuperado  0
45   45   0.7306329  9.7942455 Recuperado  0
938 938 -59.5979797 -0.1659878  Infectado  0
514 514  -4.2425868 -8.8125356  Infectado  0
222 222 -24.4480844 14.0292113  Infectado  0
405 405 -19.4793440 21.3504378 Recuperado  0
554 554 -14.2913524  6.8177213  Infectado  0
509 509  24.3767116  2.6385457 Recuperado  0
97   97   9.7431532 12.7378400 Recuperado  0
630 630   1.6902298 -7.7756708  Infectado  0
498 498   8.6221662 11.2594979 Recuperado  0
183 183   1.4182934  8.5937086  Infectado  0
315 315   8.4481278  0.2361938 Recuperado  0
```

Para cada problema de esta actividad:

- Dado un valor inicial de N, crear 3 variables para contar el número inicial de infectados, de susceptibles y de recuperados. Las variables deben ser tales que $N = I + S + R$ y que haya por lo menos una persona infectada. Inicialmente no hay recuperados ($R = 0$).

- Crear 2 variables para definir el "radio" de infección r, que representará la distancia para poderse infectar (por ejemplo $r = 0.6$), y para la razón de recuperación.
- Crear una variable categórica (factor variable) que represente el estado de la persona (suceptible, infectada o recuperada).
- Crear una estructura de datos (dataframe o varios arreglos) para representar las variables `posición x`, `posición y`, `estado`, `id único de la persona` y `número de iteración` de las N personas de manera que haya I personas infectadas, S susceptibles y R recuperados.
- Escribir una función que revise la distancia euclidiana entre dos puntos y regrese `TRUE` (o `1`) si la distancia es menor que r y regrese `FALSE` (o `0`) si la distancia es mayor o igual que r.