Ad Hoc Wireless Network

Elvin Liu (2239429)

2024-03-07

```
set.seed(123)
# n is the number of nodes to generate
genNodes = function(n) {
  source("nodeDensity.R")

  nodeDensity_max = max(replicate(1000, nodeDensity(runif(1, 0, 100), runif(1, 0, 100))))

  mat = matrix(NA, nrow = n, ncol = 2)

  while (n > 0) {
    x = runif(1, 0, 100)
    y = runif(1, 0, 100)
    u = runif(1, 0, nodeDensity_max)

    if (u < nodeDensity(x, y)) {
      mat[n, 1] = x
      mat[n, 2] = y
      n = n - 1
    }
  }

  return(mat)
  # return value is n x 2 matrix with first column x and second column y
}
```

```
findTranMat = function(mat, R) {
  n <- nrow(mat)
  tranMat <- matrix(0, nrow = n, ncol = n)

  for (i in 1:n) {
    Si <- which(mat[i,] <= R)
    num_connected <- length(Si)
    if (num_connected > 0) {
      tranMat[i, Si] <- 1 / num_connected
    }
  }

  return(tranMat)
}
```

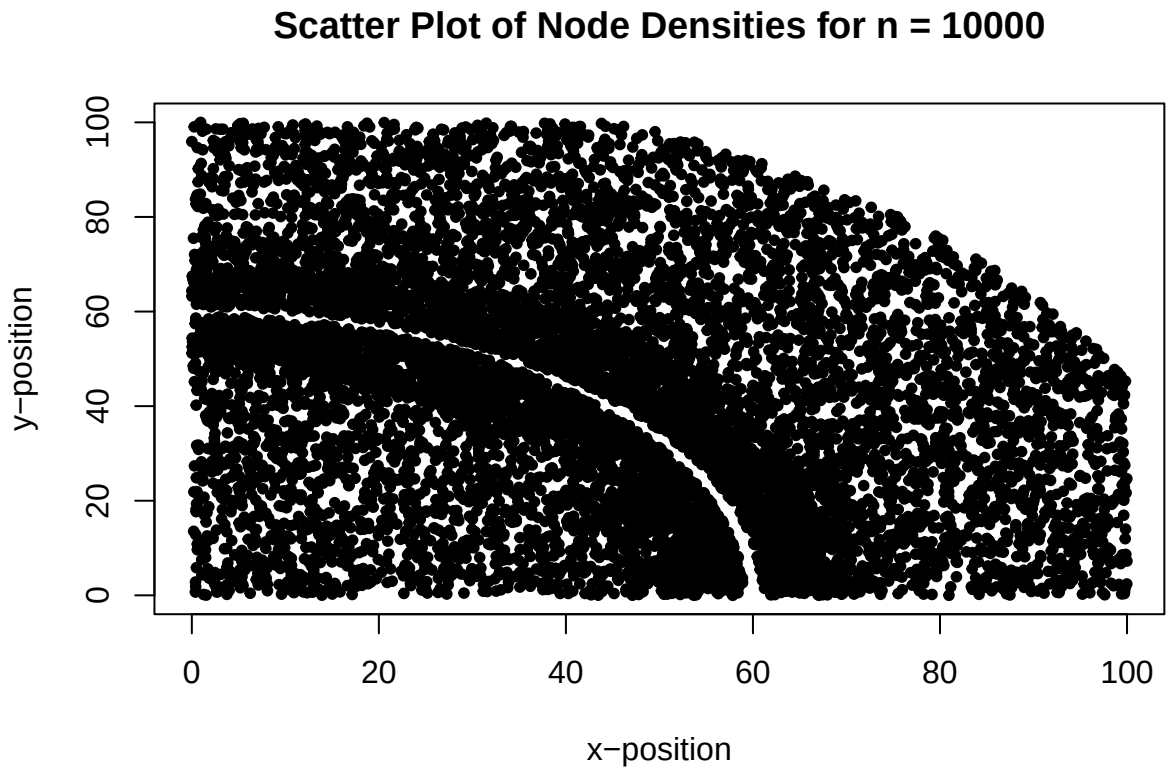
```r
getEigen2 = function(mat) {
  e_values <- eigen(mat)$values

  eigen2 <- e_values[2]
  return(eigen2)
}
```

```r
# Keep in mind diagonals are 0
findRange = function(mat) {
  rowMin = rep(Inf, nrow(mat))
  rowMax = rep(-1 * Inf, nrow(mat))

  for (r in 1:nrow(mat)) {
    if (r == 1) {
      rowMin[r] = mat[r, 2]
      rowMax[r] = mat[r, 2]
    }
    else {
      rowMin[r] = mat[r, 1]
      rowMax[r] = mat[r, 1]
    }

    for (c in 1:nrow(mat)) {
      if (c != r) {
        if (rowMin[r] > mat[r, c]) {
          rowMin[r] = mat[r, c]
        }
        if (rowMax[r] < mat[r, c]) {
          rowMax[r] = mat[r, c]
        }
      }
    }
  }

  c(max(rowMin), min(rowMax))
}
```

```r
findRc = function(nodes, tol = 0.05) {
  distance_mat <- as.matrix(dist(nodes))

  range_r <- findRange(distance_mat)

  r_min <- range_r[1]
  r_max <- range_r[2]
  r_mid <- (r_min + r_max) / 2

  while (abs(r_min - r_max) >= tol) {
    r_mid = (r_min + r_max) / 2

    tranMat <- findTranMat(distance_mat, r_mid)

    eigen2 <- Mod(getEigen2(tranMat))
```

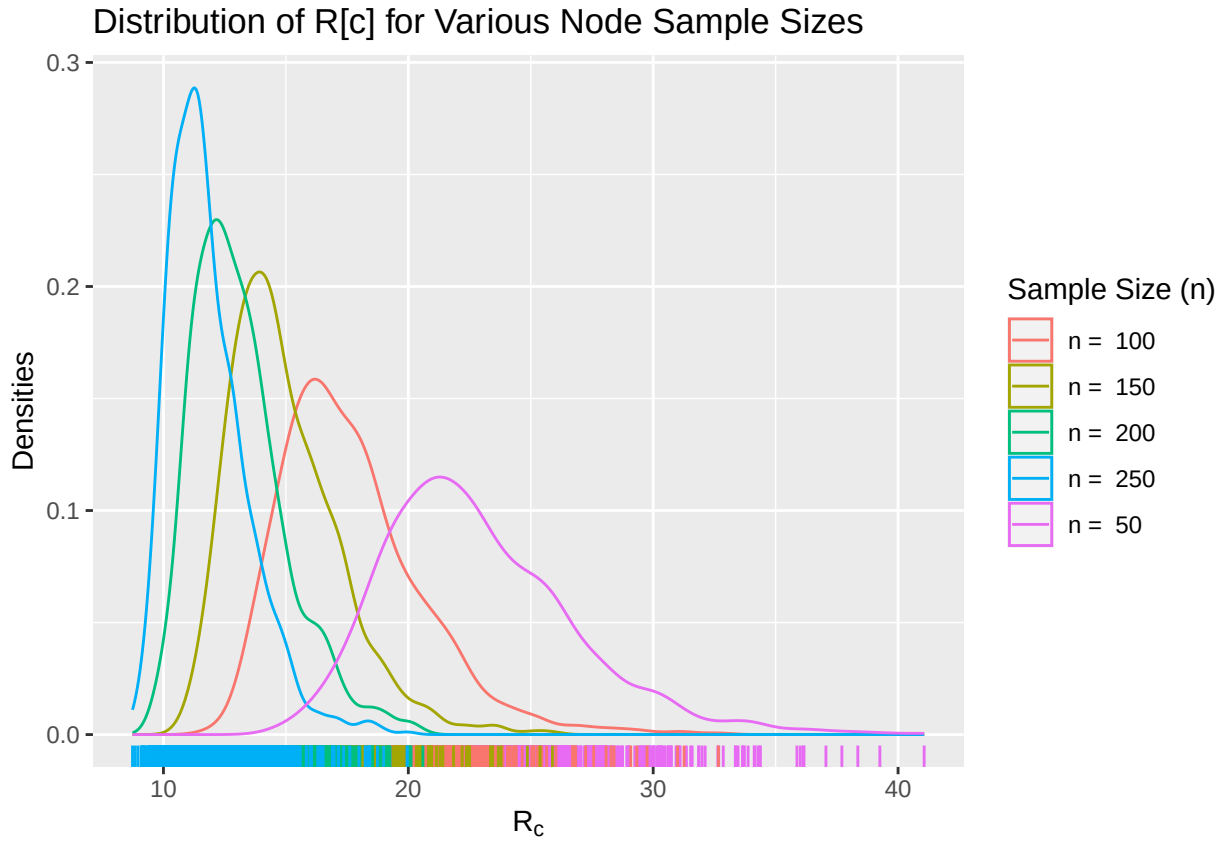
```
    if(abs(eigen2 - 1) > 1e-10 & eigen2 < 1) {
      r_max = r_mid
    }
    else {
      r_min = r_mid
    }
  }

  return(r_mid)
}
```

```
a = genNodes(10000)
plot(a[,1], a[,2], main = "Scatter Plot of Node Densities for n = 10000", xlab = "x-position", ylab = "
```




```
n <- c(50, 100, 150, 200, 250)
N <- 1000

hist <- lapply(n, function(x) {
  replicate(N, findRc(genNodes(x), tol = 0.001))
})

combined_data <- bind_rows(
  lapply(seq_along(hist),
         function(i) {
           data.frame(value = hist[[i]],
```

```
                    source_list = paste("n = ", i * 50))
          }))

ggplot(combined_data, aes(x = value, color = source_list)) +
  geom_density() + geom_rug() +
  labs(x = expression(R[c]),
       y = "Densities",
       title = paste("Distribution of", expression(R[c]),
                     "for Various Node Sample Sizes"),
       color = "Sample Size (n)")
```



(a) How does R_c, the smallest radius such that the network is connected, change with different node configurations?

At larger inputs of n in genNodes(), the smallest radius such that the network is connected (R_c) is, on average, smaller.

(b) Explore the distribution of R_c (for each n). Is it symmetric, skewed, heavy-tailed, and multimodal.

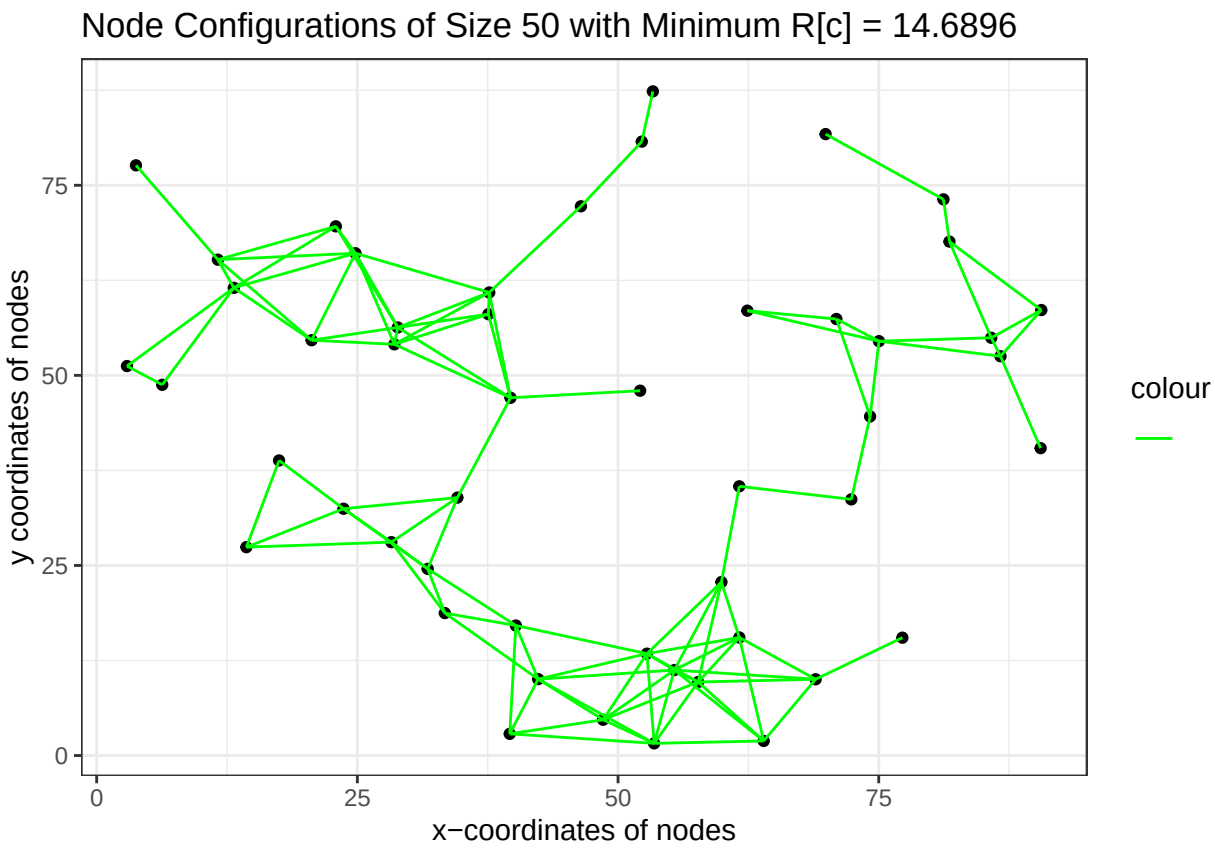
For small values of n in genNodes(), the distribution of R_c is pretty symmetric, and grows more and more right-skewing as n increases. It is not particularly heavy-tailed. They are all generally unimodal with only some small bumps in the graphs.

```r
size <- 50
N <- 1000

set.seed(123)
seed_list <- sample(1:100000, N, replace = TRUE)

mat1 <- matrix(data = 0, ncol = 2, nrow = N)

for (i in 1:length(seed_list)) {
  curr_seed = seed_list[i]
  set.seed(curr_seed)
  Rc = findRc(genNodes(size), tol = 0.001)
  mat1[i, ] = c(curr_seed, Rc)
}

median_single <- function(x) {
  if (length(x) %% 2 == 0) {
    sorted_x <- sort(x)
    return(sorted_x[(length(x) + 1) / 2])
  }
  else {
    return(median(x))
  }
}

closest_mean <- function(x) {
  sorted_x <- sort(x)
  mean_x <- mean(x)

  closest_x <- x[1]
  closest_diff <- abs(closest_x - mean_x)

  for(i in 2:length(x)) {
    curr_x = x[i]
    curr_diff = abs(closest_x - mean_x)
    if (curr_diff < closest_diff) {
      closest_x = curr_x
      curr_diff = curr_diff
    }
  }

  return(closest_x)
}

s1 <- mat1[min(mat1[, 2]) == mat1[, 2]][1]
s2 <- mat1[median_single(mat1[, 2]) == mat1[, 2]][1]
s3 <- mat1[max(mat1[, 2]) == mat1[, 2]][1]
s4 <- mat1[closest_mean(mat1[, 2]) == mat1[, 2]][1]
```

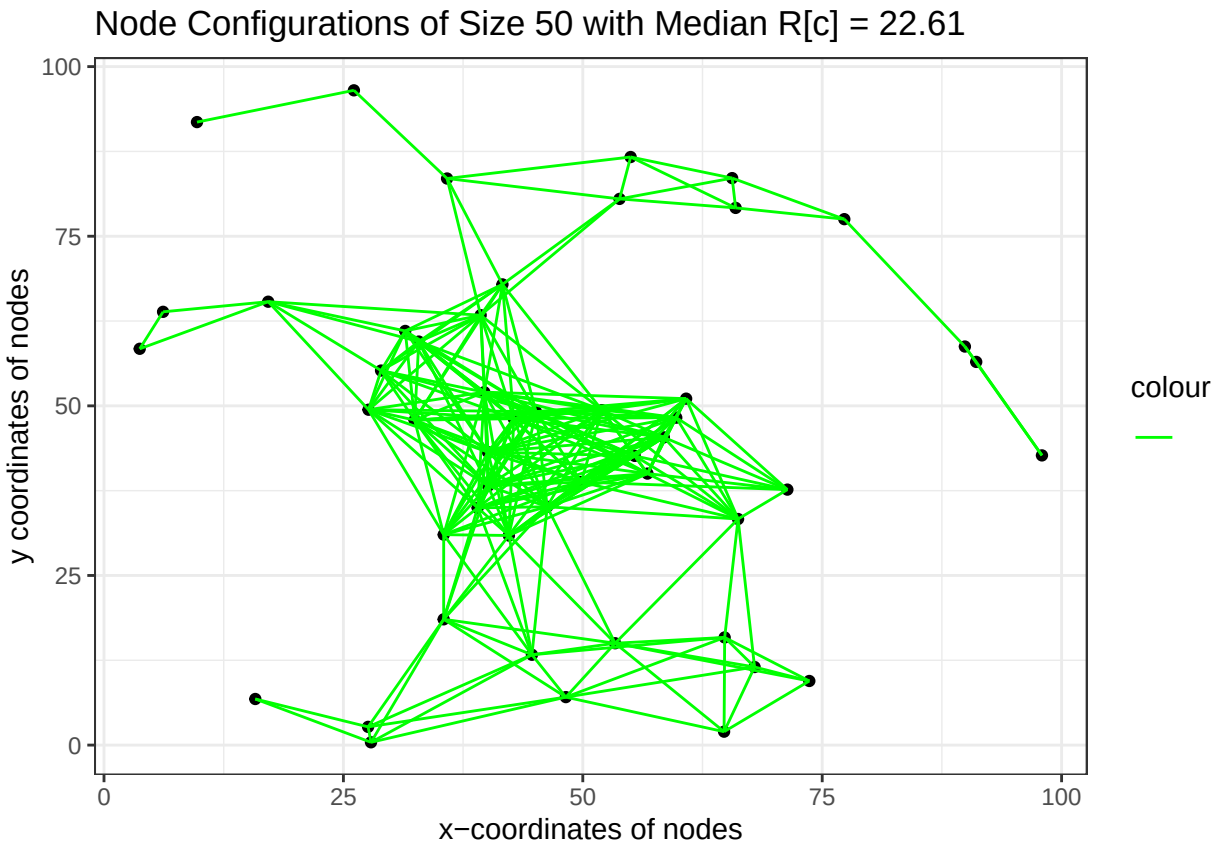
```r
connectedGraph = function(s, size = 50, name) {
  set.seed(s)

  nodes <- genNodes(size)
```

```
  Rc <- findRc(nodes, tol = 0.0001) + 0.005

  distances <- as.matrix(dist(nodes))

  # Most edges in a simply connected graph is n * (n - 1) / 2 for n vertices
  pairs <- rbind(cbind(nodes,
                       matrix(data = NA,
                              nrow = nrow(distances),
                              ncol = 2)),
                 matrix(data = NA, nrow = (size^2) / 2, ncol = 4))

  # Extra pairs keeps track of the index of the pairs past the initial 50
  extra_pairs <- size + 1
  for (i in 1:nrow(distances)) {
    for (j in 1:ncol(distances)) {
      if (j > i) {
        if (distances[i, j] <= Rc) {
          if (is.na(pairs[i, 3])) {
            pairs[i, 3:4] = nodes[j, ]
          }
          else {
            pairs[extra_pairs, 1:2] = nodes[i, ]
            pairs[extra_pairs, 3:4] = nodes[j, ]
            extra_pairs = extra_pairs + 1
          }
        }
      }
    }
  }

  # x and y positions of the nodes
  df <- data.frame(x = nodes[, 1], y = nodes[, 2])

  # pairs of nodes that are connected by an edge
  end <- data.frame(x = pairs[, 1], y = pairs[, 2],
                    x_end = pairs[, 3], y_end = pairs[, 4])

  end <- na.omit(end)

  graph <- ggplot(data = df, mapping = aes(x = x, y = y)) +
    geom_point() +
    geom_segment(data = end,
                 mapping = aes(x = x, y = y,
                               xend = x_end, yend = y_end, color = "")) +
    scale_color_manual(values = "green") +
    theme_bw() +
    labs(x = "x-coordinates of nodes",
         y = "y coordinates of nodes",
         title = paste("Node Configurations of Size 50 with", name,
                       expression(R[c]), "=", round(Rc, 4)))

  return(graph)
}
```

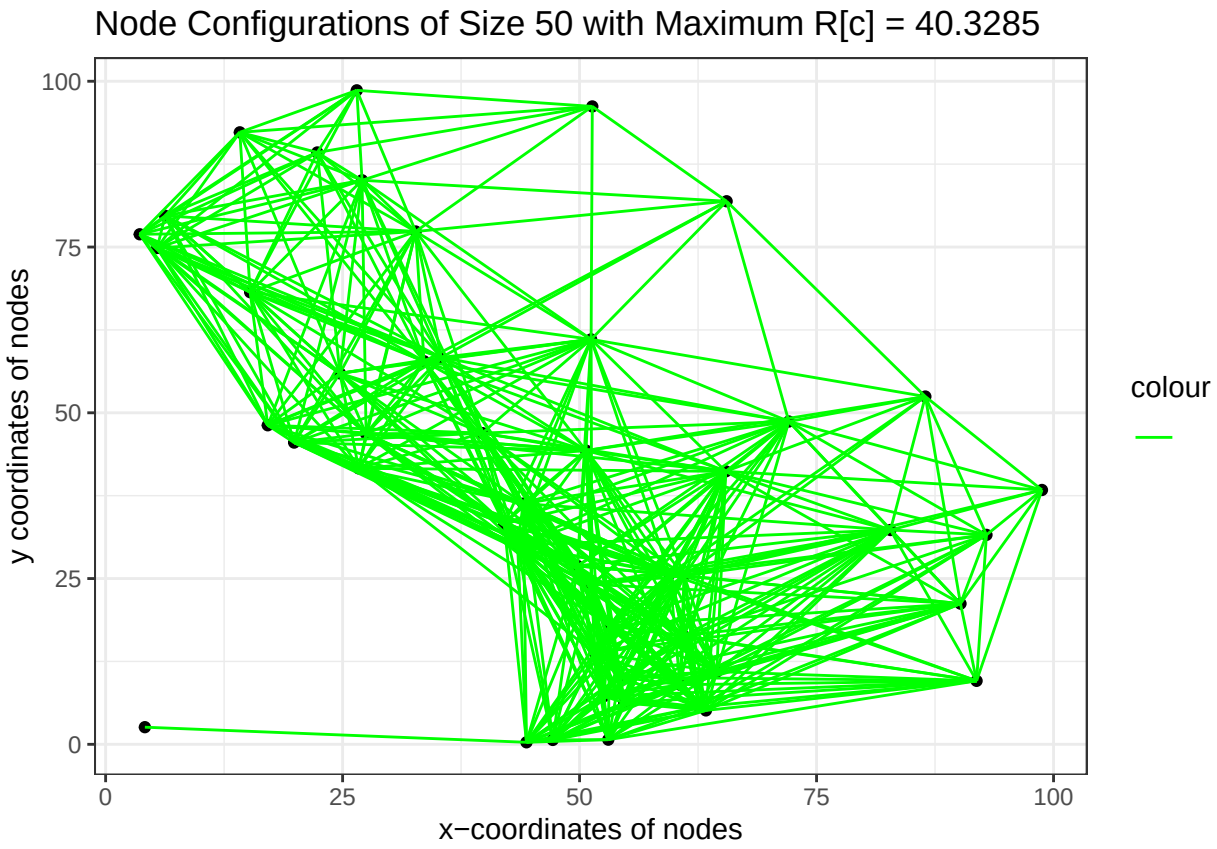
```
connectedGraph(s = s1, name = "Minimum")
```



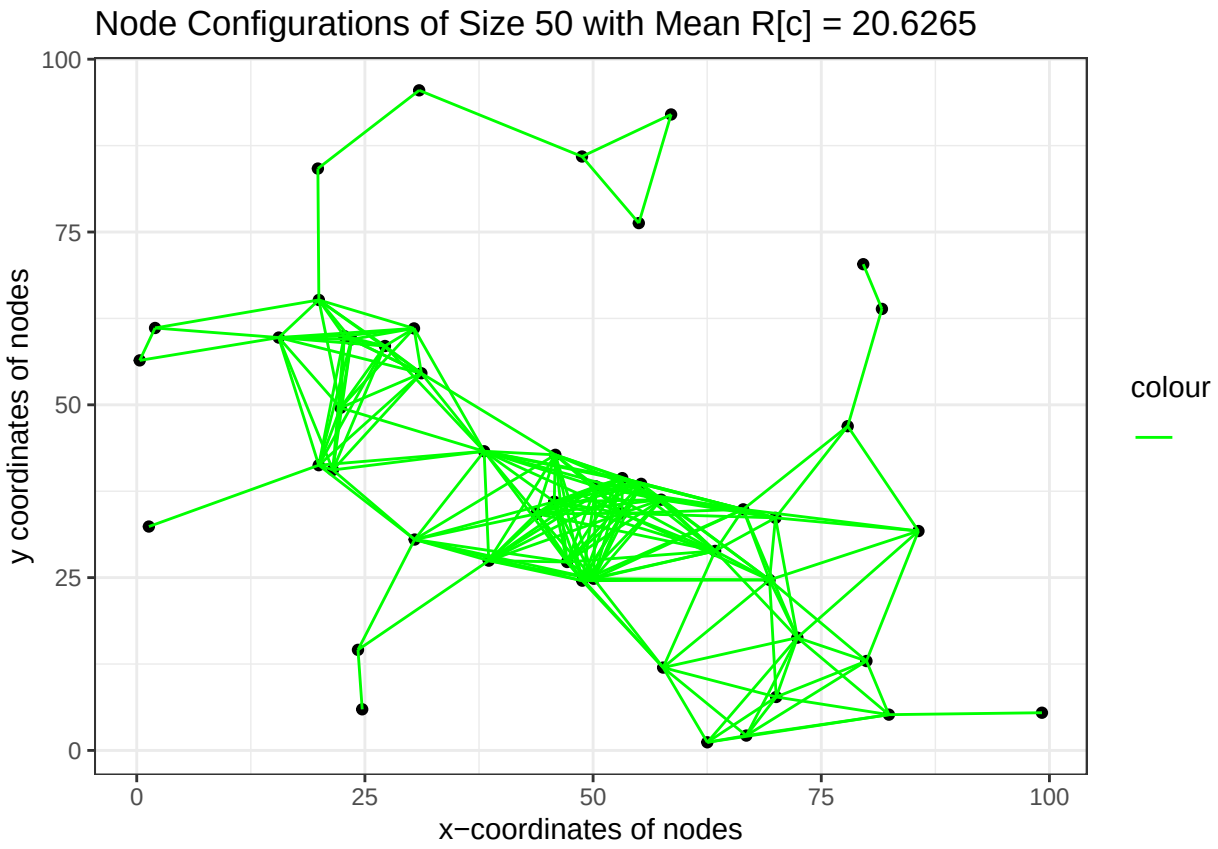

```
connectedGraph(s = s2, name = "Median")
```




```
connectedGraph(s = s3, name = "Maximum")
```




```
connectedGraph(s = s4, name = "Mean")
```




```
# Naive way that only works for s1:

# set.seed(s1)
# s1_nodes <- genNodes(size)
# s1_Rc <- findRc(s1_nodes, tol = 0.001)
# s1_distances <- as.matrix(dist(s1_nodes))
#
# s1_pairs <- rbind(cbind(s1_nodes, matrix(data = NA, nrow = nrow(s1_distances), ncol = 2)), matrix(dat
#
# extra_pairs <- size + 1
# for (i in 1:nrow(s1_distances)) {
#   for (j in 1:ncol(s1_distances)) {
#     if (j > i) {
#       if (s1_distances[i, j] <= s1_Rc) {
#         if (is.na(s1_pairs[i, 3])) {
#           s1_pairs[i, 3:4] = s1_nodes[j, ]
#         }
#         else {
#           s1_pairs[extra_pairs, 1:2] = s1_nodes[i, ]
#           s1_pairs[extra_pairs, 3:4] = s1_nodes[j, ]
#           extra_pairs = extra_pairs + 1
#         }
#       }
#     }
#   }
# }
```

```
#
# s1_df <- data.frame(x = s1_nodes[, 1], y = s1_nodes[, 2])
#
# s1_end <- data.frame(x = s1_pairs[, 1], y = s1_pairs[, 2], x_end = s1_pairs[, 3], y_end = s1_pairs[, 4])
# s1_end <- na.omit(s1_end)
#
# ggplot(data = s1_df,
#        mapping = aes(x = x, y = y)) +
#   geom_point() +
#   geom_segment(data = s1_end,
#                mapping = aes(x = x, y = y, xend = x_end, yend = y_end, color = "red"))

# Method using adjacency matrix:

# # Distance between nodes
# compute_distance <- function(node1, node2) {
#   return(sqrt((node1[1] - node2[1])^2 +
#               (node1[2] - node2[2])^2))
# }
#
# num_nodes <- nrow(s1_nodes)
# adj_matrix <- matrix(0, nrow = num_nodes, ncol = num_nodes)
#
# # Populate adjacency matrix based on distance criterion
# for (i in 1:num_nodes) {
#   for (j in 1:num_nodes) {
#     if (j >= i) {
#       if (compute_distance(s1_nodes[i, ], s1_nodes[j, ]) <= s1_Rc) {
#         adj_matrix[i, j] <- 1
#         adj_matrix[j, i] <- 1
#       }
#     }
#   }
# }
#
# graph <- graph_from_adjacency_matrix(adj_matrix)
```