ADHOC

```
library(ggplot2)
```

Introduction

Wireless networks are all around us. Cell phones communicate with a base-station to send and receive calls. Calls are relayed from base-station to base-station as the cell phone moves away from one and closer to another. A new idea of organizing networks is to avoid the need for a central base-station that coordinates communications. Instead, messages are relayed by "hopping" from one node to the next to the next until it reaches its destination. In other words, one can send a message by using other devices in the network to relay the message to the next device, and so on. These are called ad hoc networks because there is no centralized node or fixed structure or topology for the network. Instead, devices can move over time, and dynamically enter and exit the network. And so the route a message takes from one device to another depends on the other nodes. Ad hoc networks are very promising and becoming important. At their most immediately practical, ad hoc networks can allow nodes outside of a regular network to communicate by piggybacking off of nodes within the network. Think of driving along and being between base-stations and so your cell phone call would be dropped. But because of ad hoc networks, your data is relayed through cell phones in other cars closer to the base stations. More ambitously, ad hoc networks might be used in controlling traffic on highways by allowing cars communicate with each other. A very basic aspect of ad hoc networks that people need to understand is how the communication and complete connectivity changes with respect to the broadcasting power. Increased power levels allow one to send a message over a larger distance.

Goal

Set up a random nodes and find the best efficient way to generate the Ad hoc network in terms of radius

1

Create a getNodes function which generates random Nodes.

```
getNodes = function(n){
# n is numeric singleton vector
# Nodes is a matrix with n by 2

  Nodes = matrix(runif(2*n,0,100), ncol = 2)
  colnames(Nodes) <- c("x","y")
  rownames(Nodes) <- 1:n

  return(Nodes)
}
```

Check the getNodes function with random values

```
seeds = vector(length = 5, mode = "list")
set.seed(12345678)

test.node = list()

for (i in 1:5){
```

```
  seeds[[i]] = .Random.seed
  node = getNodes(5)
  test.node[[i]] = node
}

test.node
```

```
## [[1]]
##            x         y
## 1 87.4493237 52.189195
## 2 28.8790082 39.870544
## 3 94.7211719 96.001960
## 4 88.4520706 81.390361
## 5  0.8678354  9.244898
##
## [[2]]
##          x        y
## 1 22.59221 82.44000
## 2 99.31582 86.42073
## 3 22.48521 70.13601
## 4 54.65986 99.46446
## 5 83.73536 56.21510
##
## [[3]]
##          x        y
## 1 29.98234 39.87809
## 2 28.59007 47.91054
## 3 41.23781 75.04288
## 4 40.49474 30.97696
## 5 38.97566 69.44591
##
## [[4]]
##           x        y
## 1 93.313683 24.26854
## 2  6.309057 62.63485
## 3 37.157258 90.39474
## 4 60.657133 69.63902
## 5 95.841602 27.49603
##
## [[5]]
##           x         y
## 1 42.268338 85.154321
## 2 59.756995 24.049858
## 3  2.071166 64.252719
## 4 21.317885  1.449243
## 5 40.160201 76.372598
```

2

Find the smallest radius Rc. First, Set up a helper function, findTranMat, to find a transition Matrix(Marcov Matrix for random walk).

```
findTranMat = function(mat, R){
  trans.mat = as.matrix(mat)
  trans.mat[trans.mat <= R] = NA
  trans.mat[trans.mat > R] = 0
  trans.mat[is.na(trans.mat)] = 1
  trans.mat = apply(trans.mat, 1, "/", rowSums(trans.mat))
  return(trans.mat)
}
```

Second, find a second largest eigen values since the largest eigen value gives us the trivial solution as well. Set up a helper function, genEigen2() which finds a second largest eigen value.

```
getEigen2 = function(mat){
  eigenvalues = eigen(mat)$values
  SecondLargest = sort(eigenvalues, decreasing = TRUE)[2]
  return(Mod(SecondLargest))
}
```

Third, set up a function findRange() to find the range of Rc

```
findRange = function(mat){
  Mini = min(mat)
  Maxi = max(mat)
  Min.Max = c(Mini, Maxi)
  names(Min.Max) = c("Lower Bound","Upper Bound")
  return(Min.Max)
}
```

Set up a function, findRc, to find the smallest radius to make the network fully connected

```
findRc = function(nodes, tol = 0.05){
  mat = dist(nodes)
  lower = findRange(mat)[1]
  upper = findRange(mat)[2]
  midpt = sum(upper, lower) / 2

  while(upper-lower > tol){
    TranMat = findTranMat(mat, midpt)
    EigenValue = round(getEigen2(TranMat), digits = 5)
    if(EigenValue < 1){
      upper = midpt
    }else{
      lower = midpt
    }
    midpt = sum(upper, lower) / 2
  }
  names(upper) = "The smallest Radius"
  return(upper)
}
```

Check the findRc function with random Node

```
nodes = getNodes(100)
findRc(nodes)
```

```
## The smallest Radius
##            14.98706
```

3

Generate 1000 networks and for each find the value for Rc. Examine the distribution of these Rc values.

sample with size n = 20, n = 50, n = 100

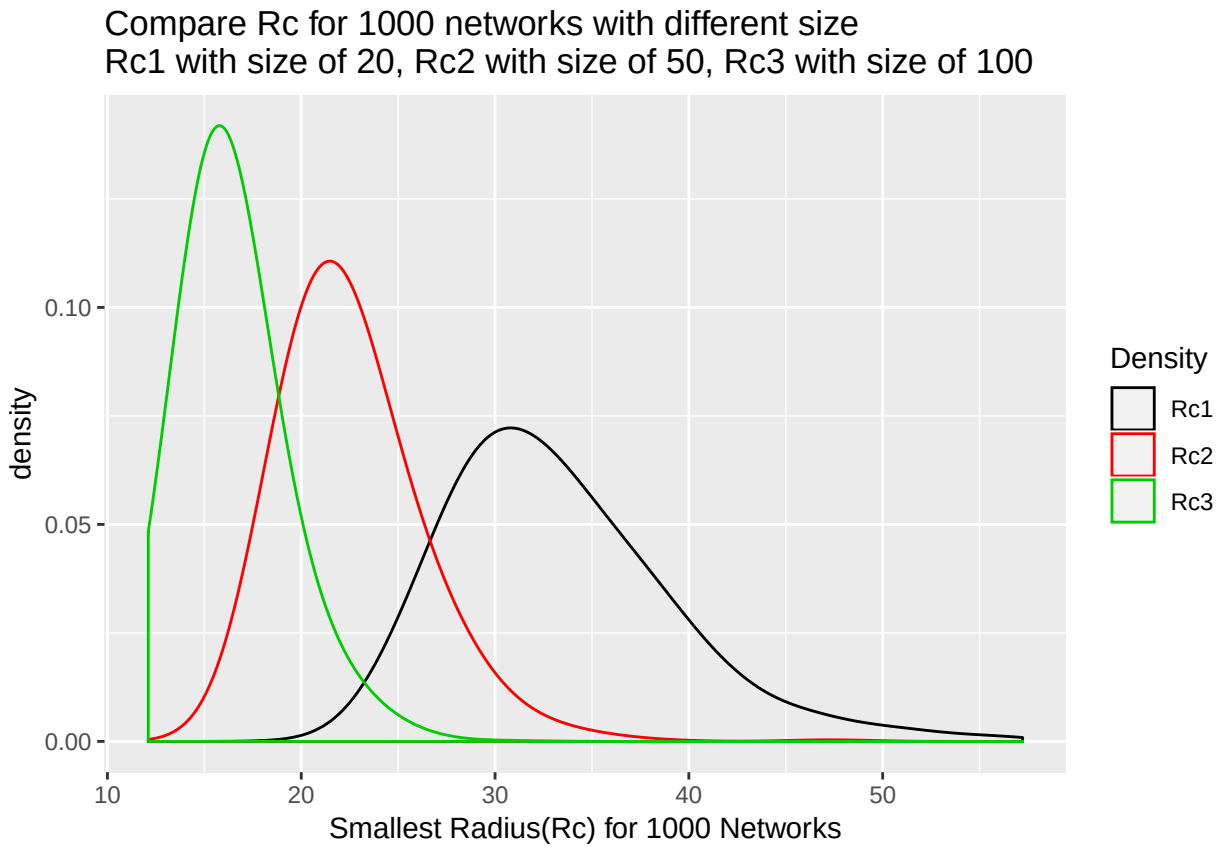
```
Rc1 = c()
Rc2 = c()
Rc3 = c()

seeds = vector(length = 2, mode = "list")
set.seed(12345678)

n1 = 20
n2 = 50
n3 = 100

networks1 = vector("list",1000)
networks2 = vector("list",1000)
networks3 = vector("list",1000)

for(i in 1:1000){
  seeds[[i]] = .Random.seed
  networks1[[i]] = getNodes(n1)
  networks2[[i]] = getNodes(n2)
  networks3[[i]] = getNodes(n3)
}



networks = list(networks1, networks2, networks3)

for(i in 1:1000){
  Rc1[i] = findRc(networks1[[i]])
  Rc2[i] = findRc(networks2[[i]])
  Rc3[i] = findRc(networks3[[i]])
}

Rc = list(Rc1, Rc2, Rc3)
```

As n goes up, the radius of the Rc is getter smaller. Set up a function to visualize the distribution of Rcs.

```
plotRc = function(Rc1, Rc2 = NULL, Rc3 = NULL, bw = 2, title = 'Compare Rc with different size'){
  Rcs = data.frame(Rc123 = c(Rc1, Rc2, Rc3),
                   strat = rep(c(1,2,3), times = c(length(Rc1), length(Rc2), length(Rc3))))
  ggplot(data = Rcs) +
    geom_density(aes(x = Rc123, y = ..density.., color = factor(strat)), bw = bw) +
    scale_x_continuous("Smallest Radius(Rc) for 1000 Networks") +
    ggtitle(title) +
    scale_colour_manual(values = c(1,2,3), name = 'Density', labels = c('Rc1','Rc2','Rc3'))
}
```

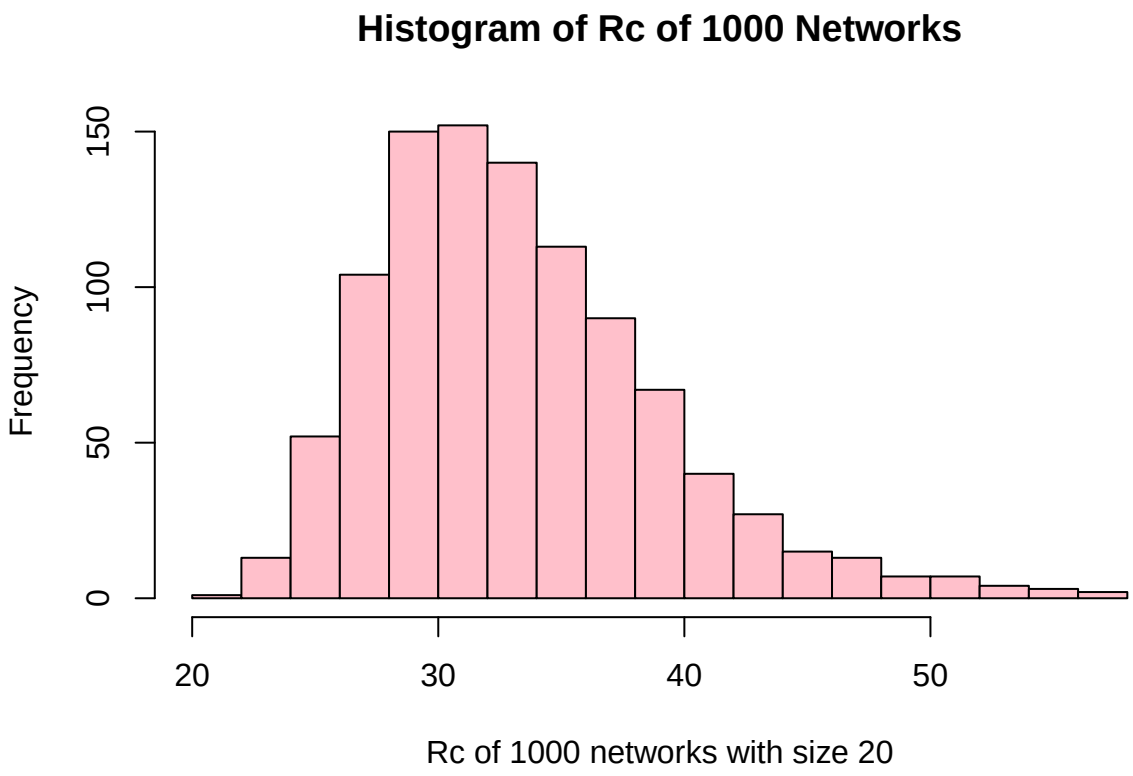
Plot distribution of Rcs.

```
plotRc(Rc1, Rc2, Rc3, title = "Compare Rc for 1000 networks with different size\nRc1 with size of 20, R
```



Plot Histogram of Rcs with summary.

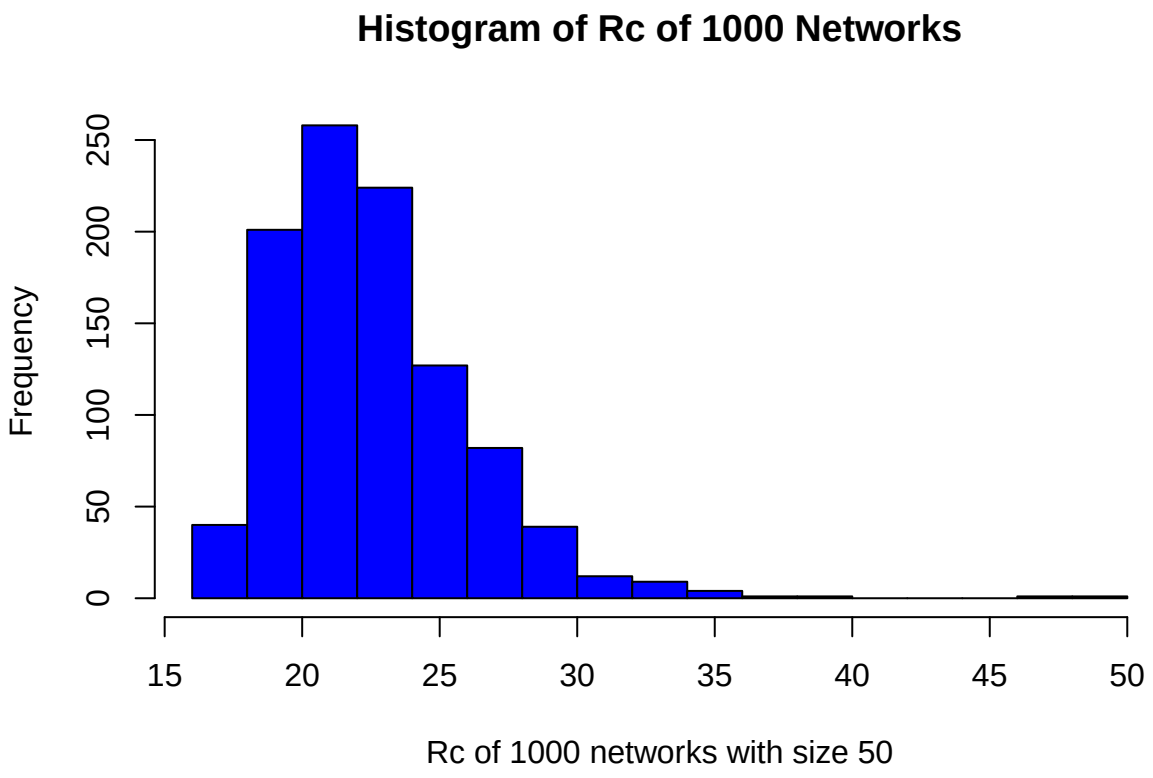
```
Rc = list(Rc1, Rc2, Rc3)
Hist1 = hist(Rc1, breaks = 20, col = "pink",
             xlab = "Rc of 1000 networks with size 20",
             main = "Histogram of Rc of 1000 Networks")
```




```
Hist1
```

```
## $breaks
##  [1] 20 22 24 26 28 30 32 34 36 38 40 42 44 46 48 50 52 54 56 58
##
## $counts
##  [1]   1  13  52 104 150 152 140 113  90  67  40  27  15  13   7   7   4
## [18]   3   2
##
## $density
##  [1] 0.0005 0.0065 0.0260 0.0520 0.0750 0.0760 0.0700 0.0565 0.0450 0.0335
## [11] 0.0200 0.0135 0.0075 0.0065 0.0035 0.0035 0.0020 0.0015 0.0010
##
## $mids
##  [1] 21 23 25 27 29 31 33 35 37 39 41 43 45 47 49 51 53 55 57
##
## $xname
## [1] "Rc1"
##
## $equidist
## [1] TRUE
##
## attr(,"class")
## [1] "histogram"
```

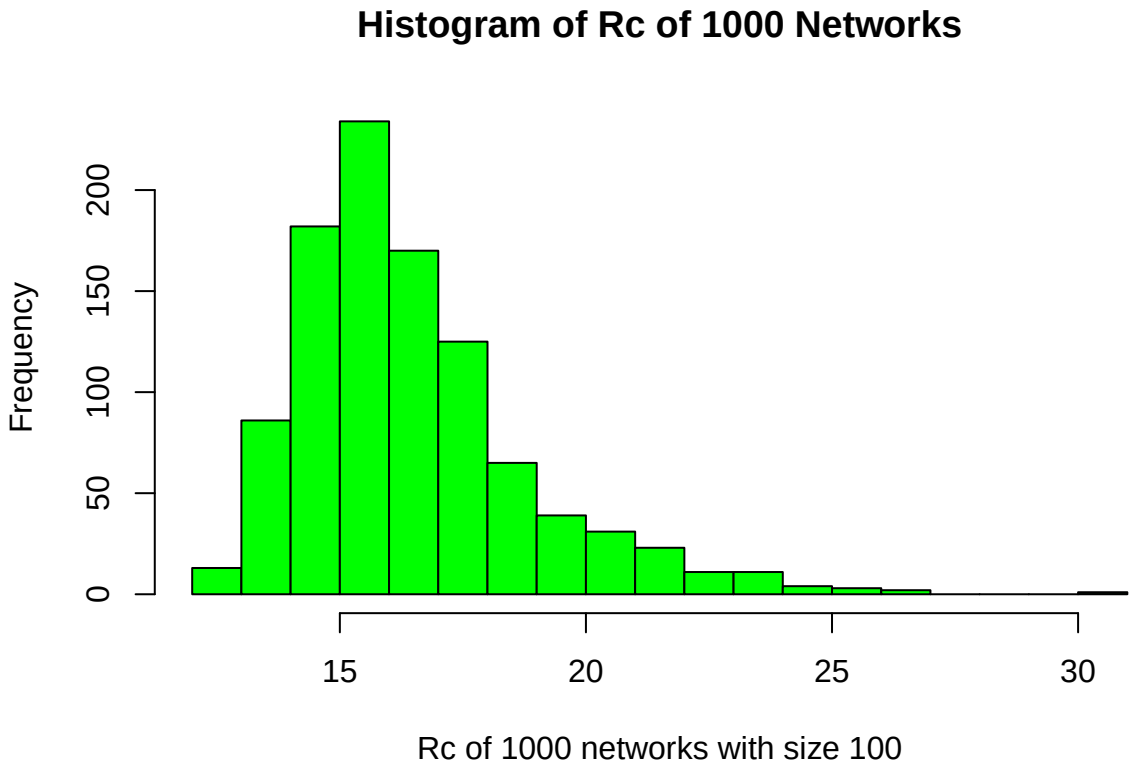
```
Hist2 = hist(Rc2, breaks = 20, col = "blue",
             xlab = "Rc of 1000 networks with size 50",
             main = "Histogram of Rc of 1000 Networks")
```




```
Hist2
```

```
## $breaks
##  [1] 16 18 20 22 24 26 28 30 32 34 36 38 40 42 44 46 48 50
##
## $counts
##  [1]  40 201 258 224 127  82  39  12   9   4   1   1   0   0   0   1   1
##
## $density
##  [1] 0.0200 0.1005 0.1290 0.1120 0.0635 0.0410 0.0195 0.0060 0.0045 0.0020
## [11] 0.0005 0.0005 0.0000 0.0000 0.0000 0.0005 0.0005
##
## $mids
##  [1] 17 19 21 23 25 27 29 31 33 35 37 39 41 43 45 47 49
##
## $xname
## [1] "Rc2"
##
## $equidist
## [1] TRUE
##
## attr(,"class")
## [1] "histogram"
```

```
Hist3 = hist(Rc3, breaks = 20, col = "green",
             xlab = "Rc of 1000 networks with size 100",
             main = "Histogram of Rc of 1000 Networks")
```




```
Hist3
```

```
## $breaks
##  [1] 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
##
## $counts
##  [1]  13  86 182 234 170 125  65  39  31  23  11  11   4   3   2   0   0
## [18]   0   1
##
## $density
##  [1] 0.013 0.086 0.182 0.234 0.170 0.125 0.065 0.039 0.031 0.023 0.011
## [12] 0.011 0.004 0.003 0.002 0.000 0.000 0.000 0.001
##
## $mids
##  [1] 12.5 13.5 14.5 15.5 16.5 17.5 18.5 19.5 20.5 21.5 22.5 23.5 24.5 25.5
## [15] 26.5 27.5 28.5 29.5 30.5
##
## $xname
## [1] "Rc3"
##
## $equidist
## [1] TRUE
##
## attr(,"class")
## [1] "histogram"
```

Set up min, median, mean, and max values for Rc

```
Min = c()
Median = c()
Mean = c()
```

```
Max = c()

for (i in 1:3){
  Min[i] = min(Rc[[i]])
  Median[i] = median(Rc[[i]])
  Mean[i] = mean(Rc[[i]])
  Max[i] = max(Rc[[i]])
}

i = c()
j = c()
k = c()
l = c()

for (h in 1:3){
  i[h] = which(abs(Rc[[h]] - Min[h]) == min(abs(Rc[[h]] - Min[h])))[1]
  j[h] = which(abs(Rc[[h]] - Median[h]) == min(abs(Rc[[h]] - Median[h])))[1]
  k[h] = which(abs(Rc[[h]] - Mean[h]) == min(abs(Rc[[h]] - Mean[h])))[1]
  l[h] = which(abs(Rc[[h]] - Max[h]) == min(abs(Rc[[h]] - Max[h])))[1]
}
```

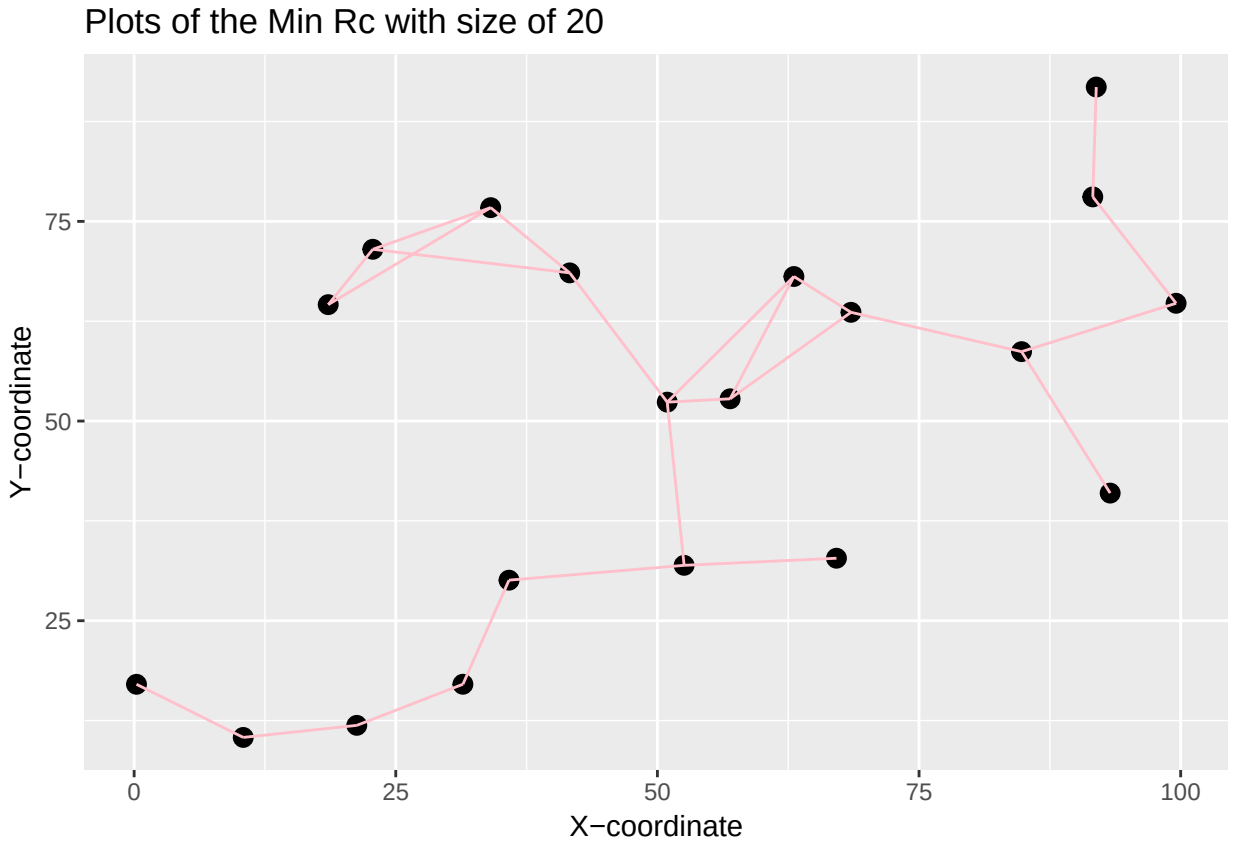
4

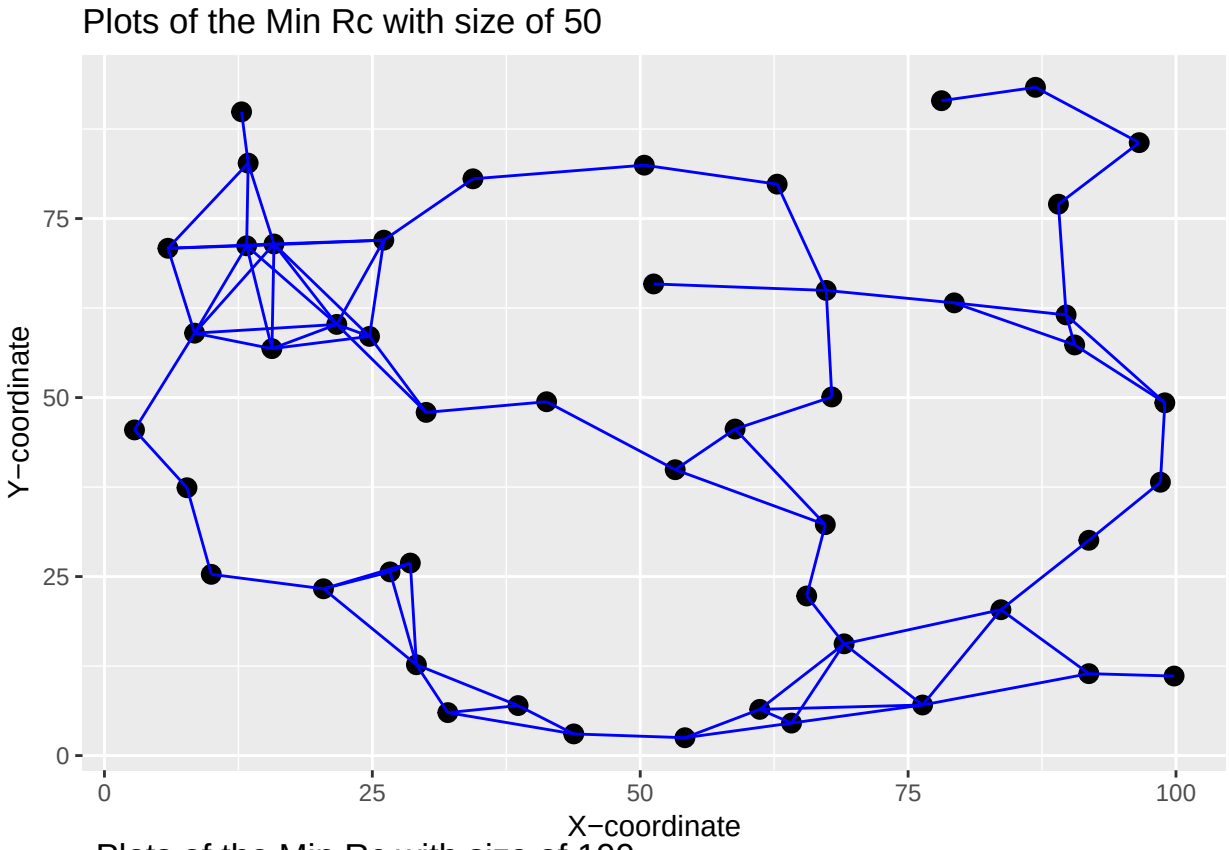
Plots of Rc with the four values(min, median, mean, max)

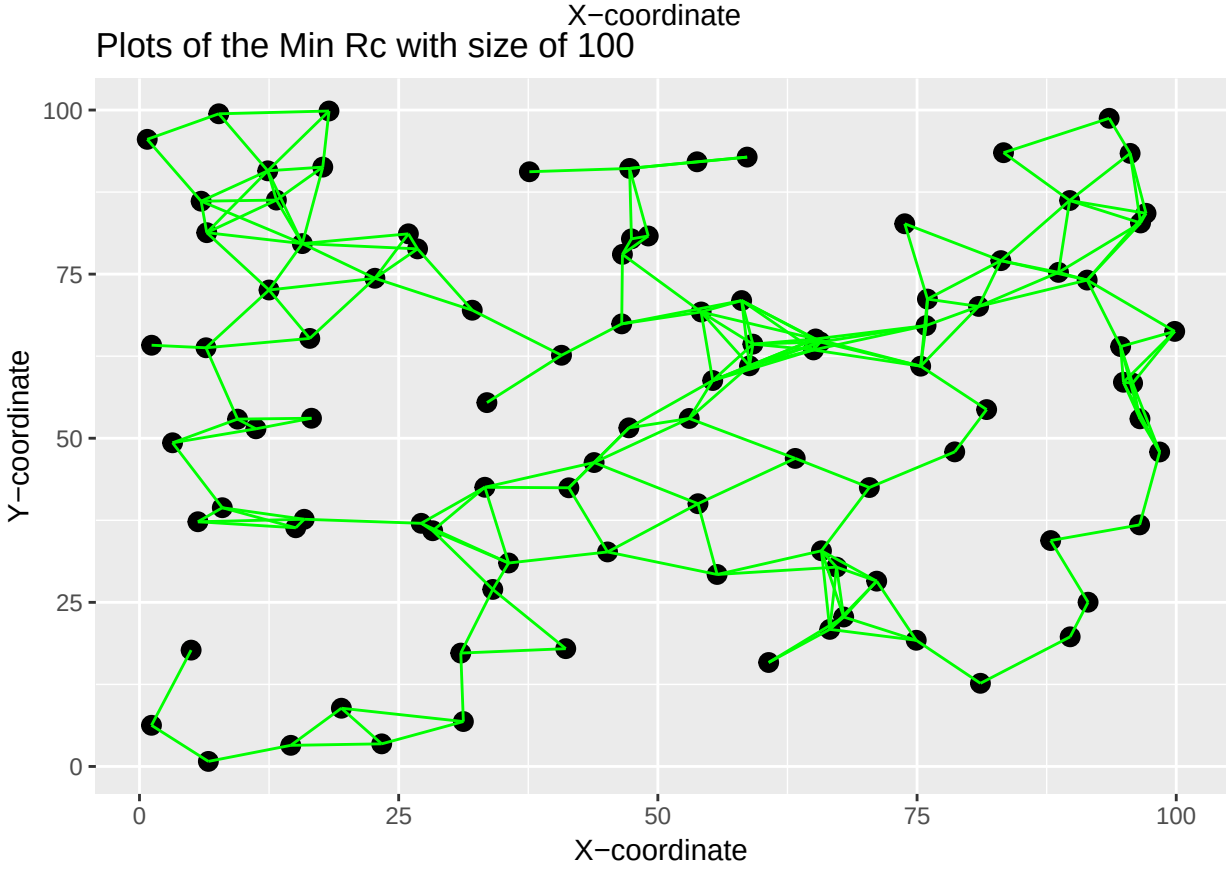
Plots of Rc with the min

```
mindf = list()
connected = list()
connections = list()

for (h in 1:3){
  mindf[[h]] = data.frame(networks[[h]][i[h]])
  comb = combn(nrow(mindf[[h]]), 2)
  connected[[h]] = comb[,which(as.numeric(dist(data.frame(networks[[h]][i[h]])))<=Rc[[h]][i[h]])]
  connections[[h]] = data.frame(
    from = mindf[[h]][connected[[h]][1, ], 1:2],
    to = mindf[[h]][connected[[h]][2, ], 1:2]
    )
  names(connections[[h]]) = c("x1", "y1", "x2", "y2")
}

color = c("pink", "blue", "green")
title = c("Plots of the Min Rc with size of 20", "Plots of the Min Rc with size of 50", "Plots of the M

for (h in 1:3){
  plot =
    ggplot(mindf[[h]], aes(x = mindf[[h]]$x, y = mindf[[h]]$y)) +
    geom_point(col = "black", size = 3) +
    geom_segment(data = connections[[h]],
                 aes(x = x1, y = y1, xend = x2, yend = y2), col = color[h]) +
    labs(x = "X-coordinate", y = "Y-coordinate", title = title[h])
```

```
  print(plot)
}
```

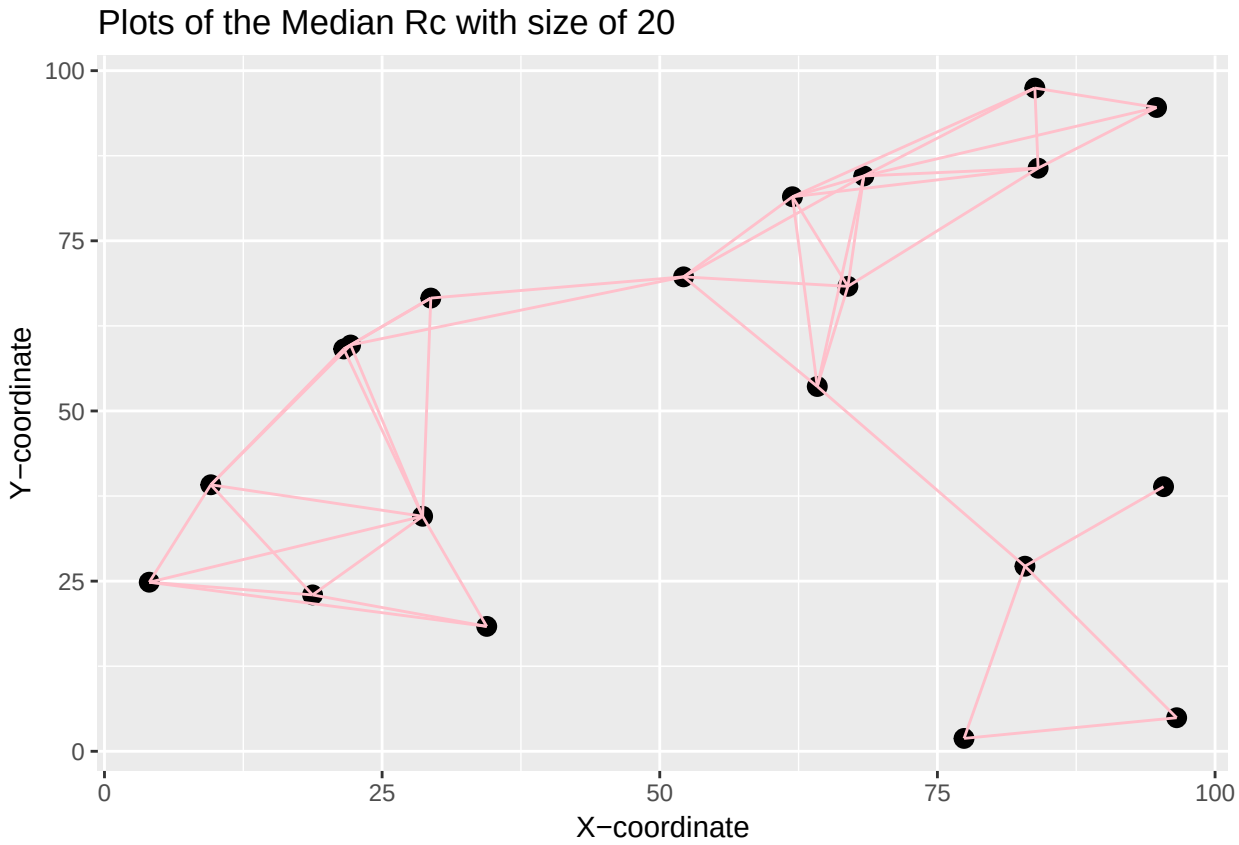

Plots of the Min Rc with size of 20
Y−coordinate
75
50
25
0
25
50
75
100
X−coordinate

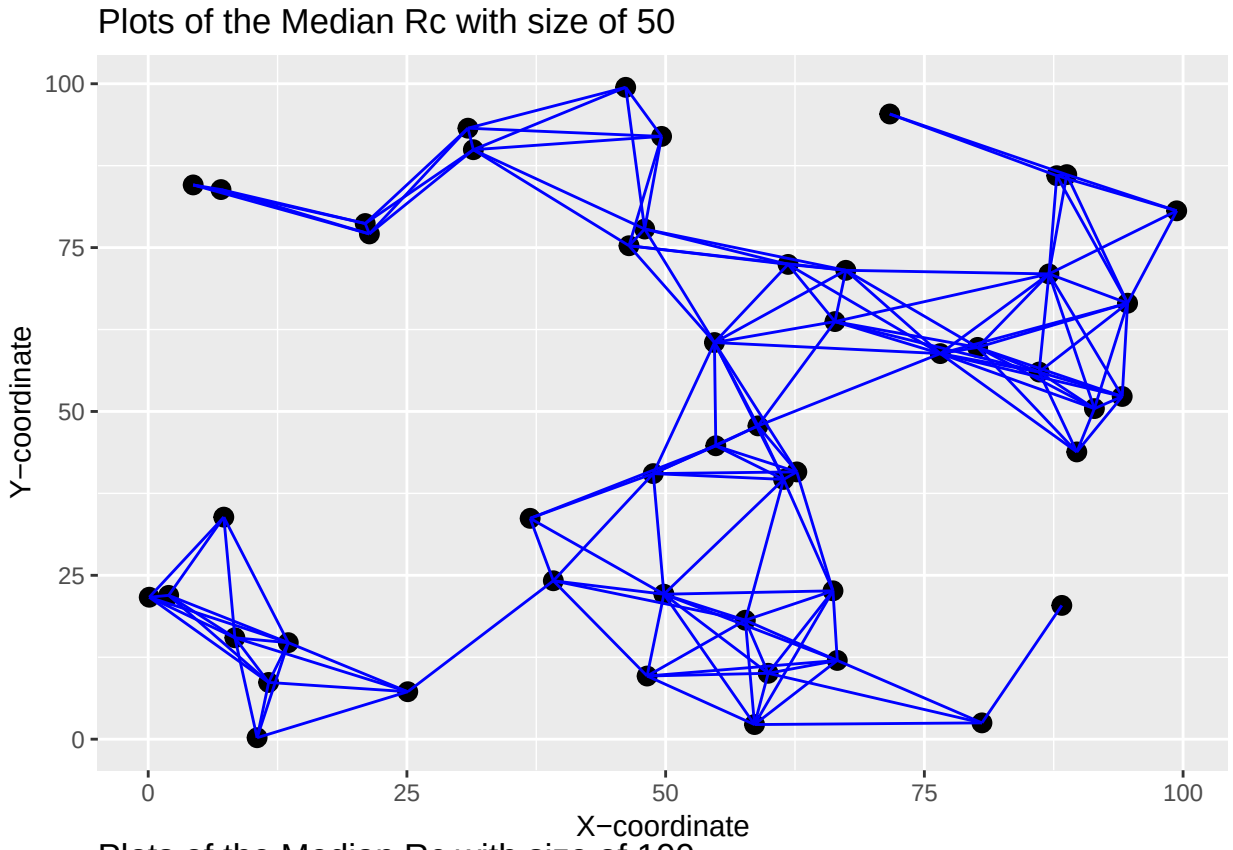


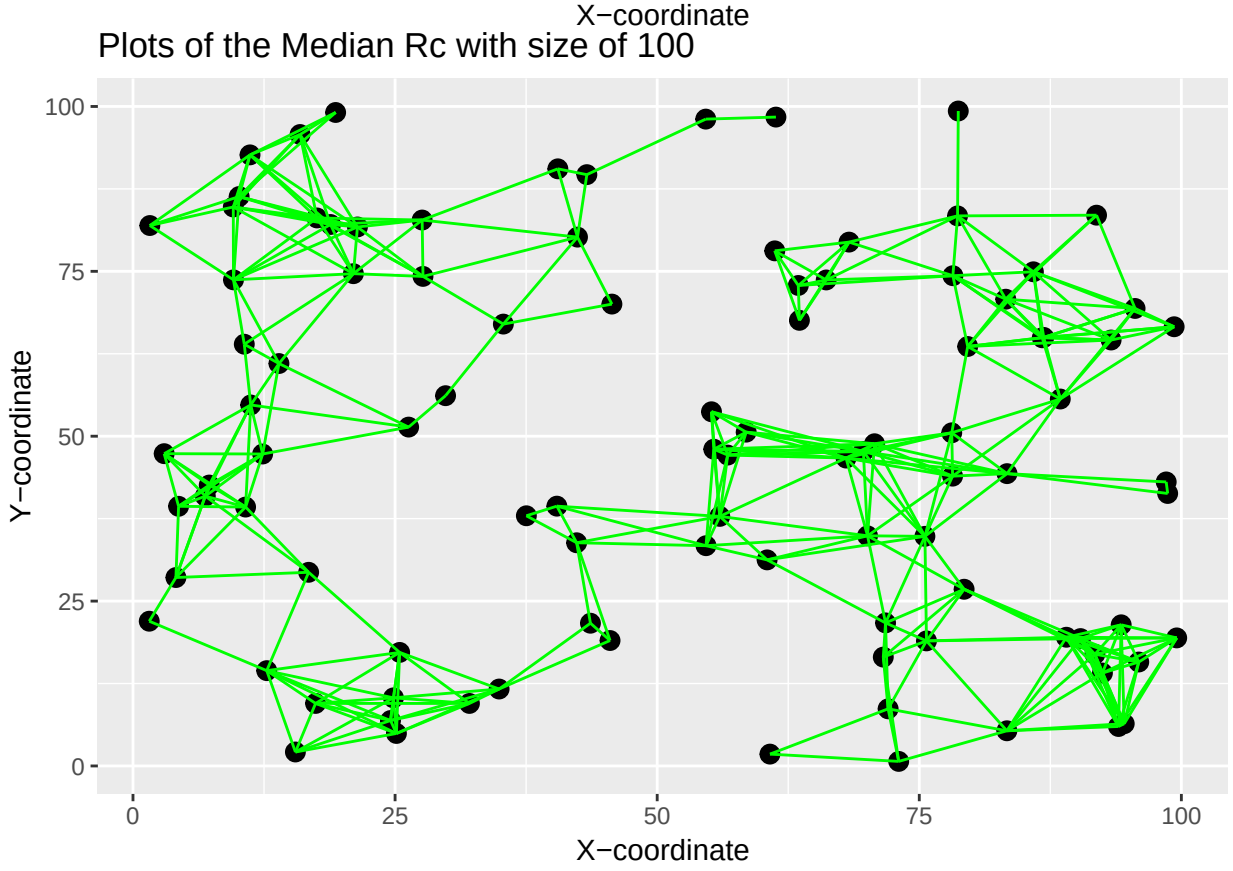


Plots of Rc with the median

```
mindf = list()
connected = list()
connections = list()

for (h in 1:3){
  mindf[[h]] = data.frame(networks[[h]][j[h]])
  comb = combn(nrow(mindf[[h]]), 2)
  connected[[h]] = comb[,which(as.numeric(dist(data.frame(networks[[h]][j[h]])))<=Rc[[h]][j[h]])]
  connections[[h]] = data.frame(
    from = mindf[[h]][connected[[h]][1, ], 1:2],
    to = mindf[[h]][connected[[h]][2, ], 1:2]
    )
  names(connections[[h]]) = c("x1", "y1", "x2", "y2")
}

color = c("pink", "blue", "green")
title = c("Plots of the Median Rc with size of 20", "Plots of the Median Rc with size of 50", "Plots of

for (h in 1:3){
  plot =
    ggplot(mindf[[h]], aes(x = mindf[[h]]$x, y = mindf[[h]]$y)) +
    geom_point(col = "black", size = 3) +
    geom_segment(data = connections[[h]],
                 aes(x = x1, y = y1, xend = x2, yend = y2), col = color[h]) +
    labs(x = "X-coordinate", y = "Y-coordinate", title = title[h])
  print(plot)
}
```

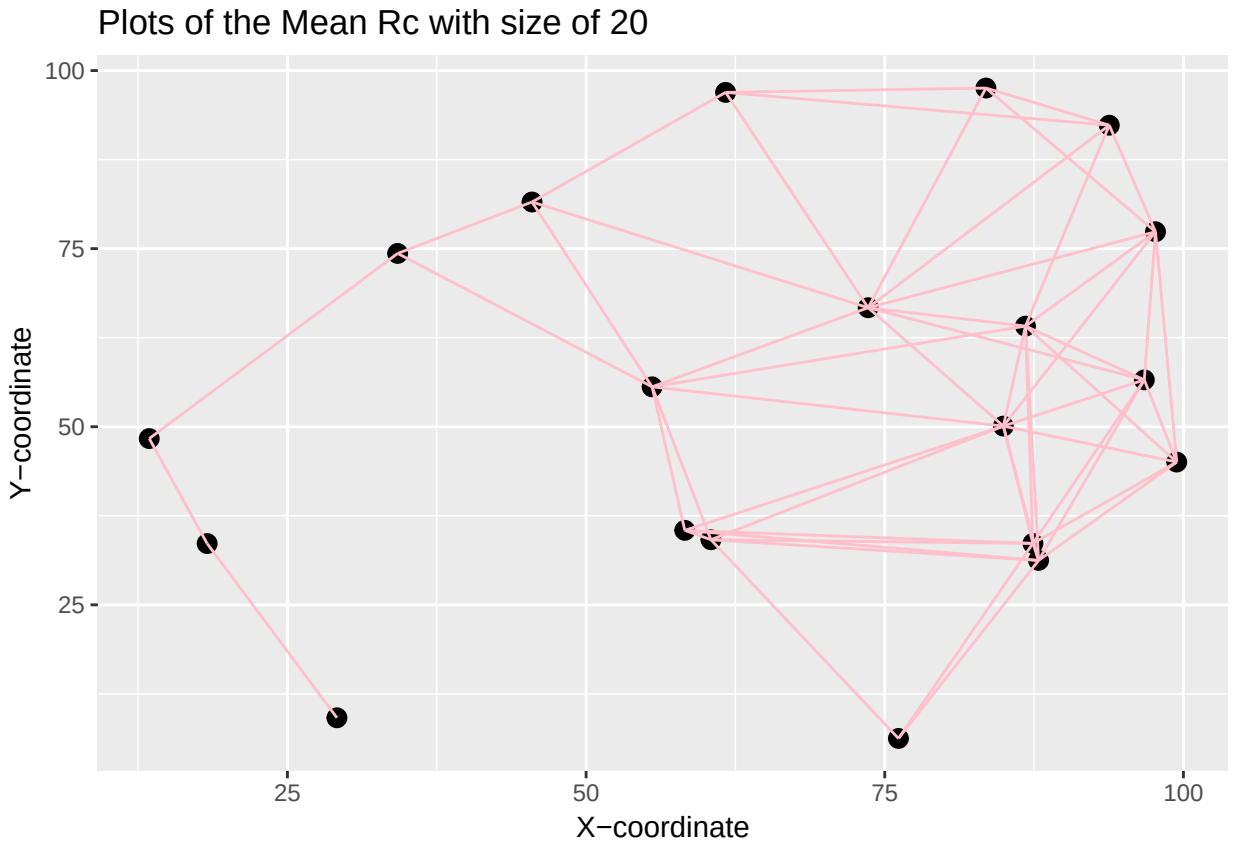


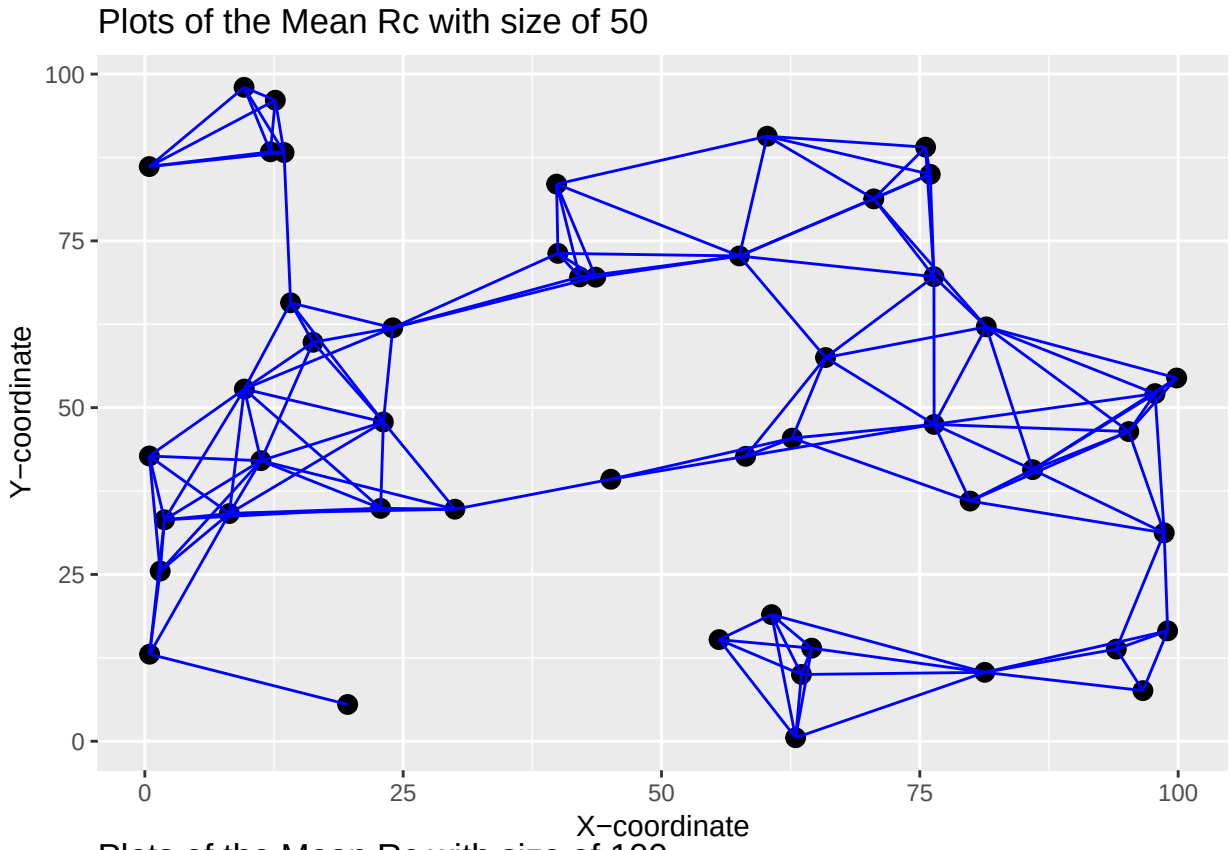


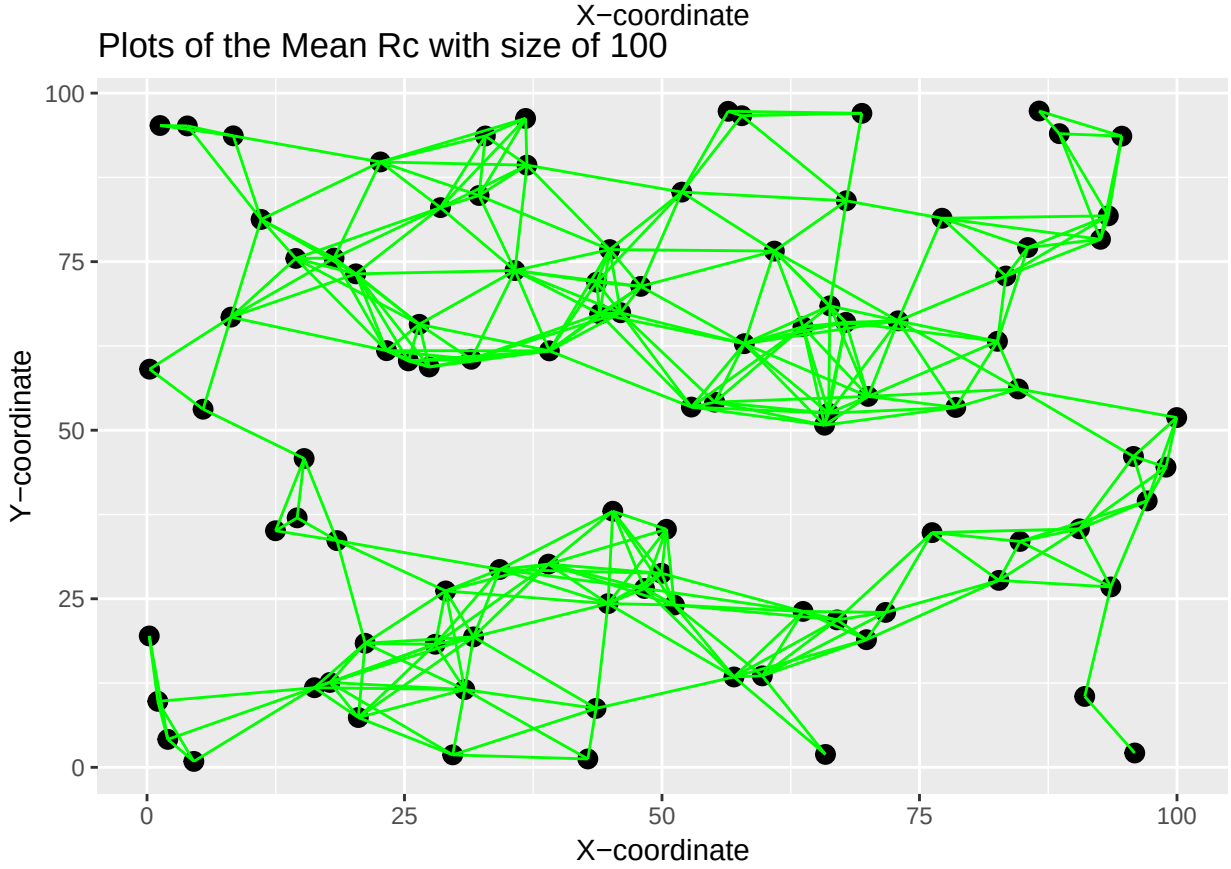


Plots of Rc with mean

```
mindf = list()
connected = list()
connections = list()

for (h in 1:3){
  mindf[[h]] = data.frame(networks[[h]][k[h]])
  comb = combn(nrow(mindf[[h]]), 2)
  connected[[h]] = comb[,which(as.numeric(dist(data.frame(networks[[h]][k[h]])))<=Rc[[h]][k[h]])]
  connections[[h]] = data.frame(
    from = mindf[[h]][connected[[h]][1, ], 1:2],
    to = mindf[[h]][connected[[h]][2, ], 1:2]
    )
  names(connections[[h]]) = c("x1", "y1", "x2", "y2")
}

color = c("pink", "blue", "green")
title = c("Plots of the Mean Rc with size of 20", "Plots of the Mean Rc with size of 50", "Plots of the

for (h in 1:3){
  plot =
    ggplot(mindf[[h]], aes(x = mindf[[h]]$x, y = mindf[[h]]$y)) +
    geom_point(col = "black", size = 3) +
    geom_segment(data = connections[[h]],
                 aes(x = x1, y = y1, xend = x2, yend = y2), col = color[h]) +
    labs(x = "X-coordinate", y = "Y-coordinate", title = title[h])
  print(plot)
}
```

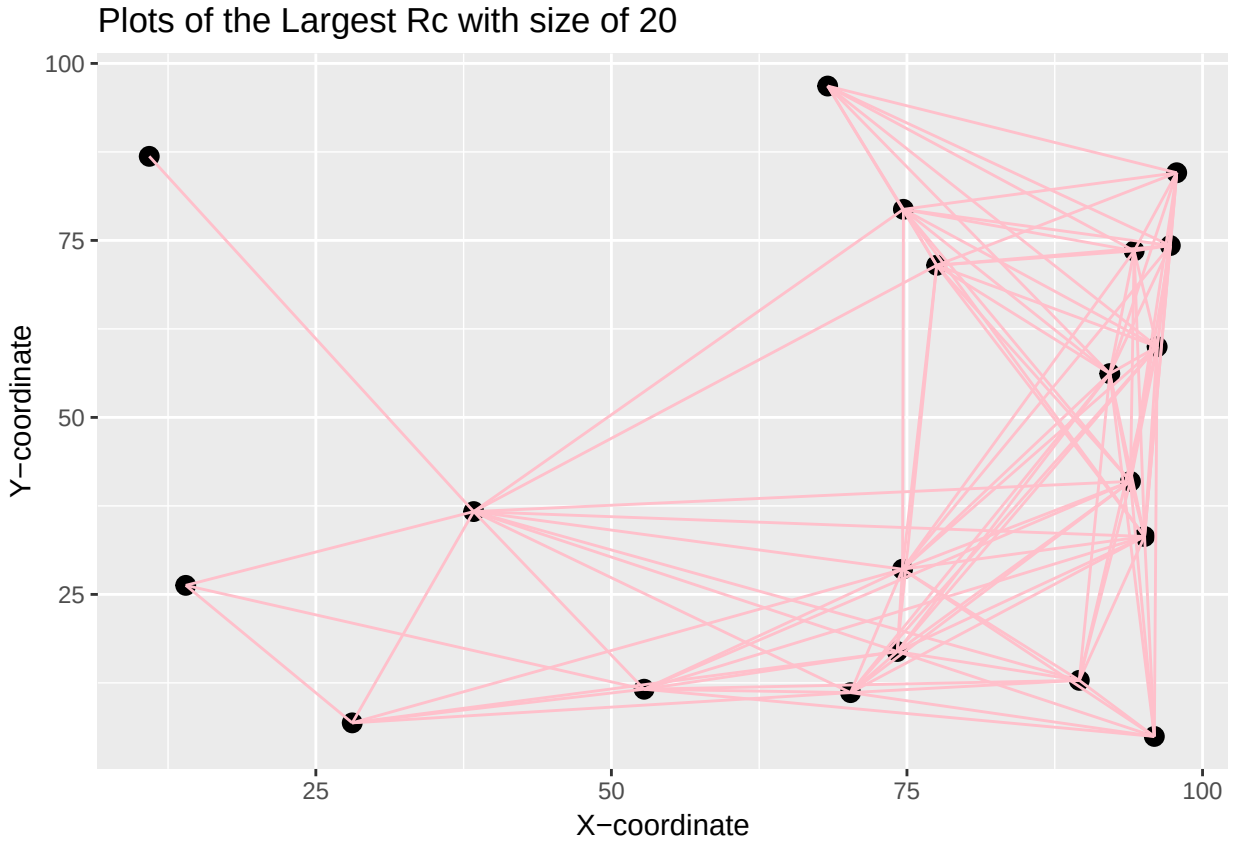


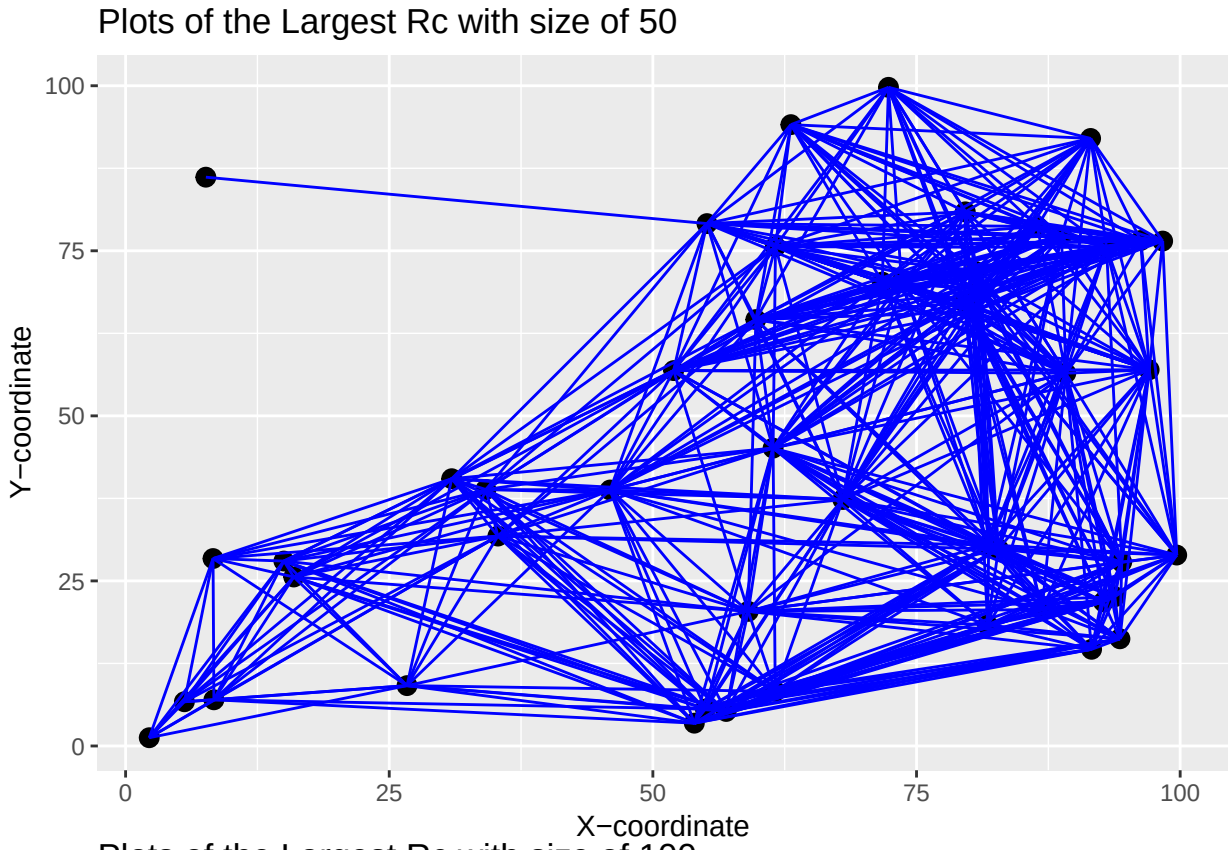


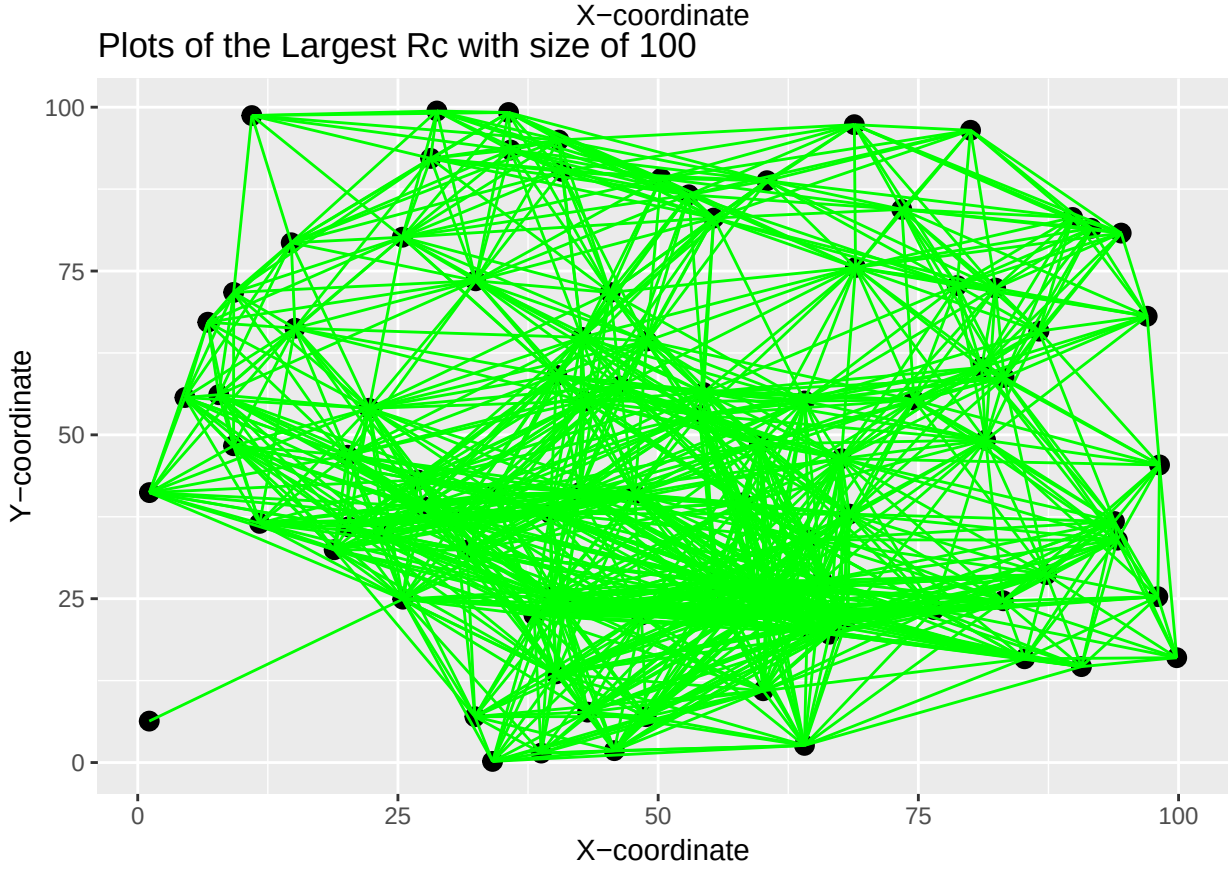


Plots of Rc with the largest

```
mindf = list()
connected = list()
connections = list()

for (h in 1:3){
  mindf[[h]] = data.frame(networks[[h]][1[h]])
  comb = combn(nrow(mindf[[h]]), 2)
  connected[[h]] = comb[,which(as.numeric(dist(data.frame(networks[[h]][1[h]])))<=Rc[[h]][1[h]])]
  connections[[h]] = data.frame(
    from = mindf[[h]][connected[[h]][1, ], 1:2],
    to = mindf[[h]][connected[[h]][2, ], 1:2]
    )
  names(connections[[h]]) = c("x1", "y1", "x2", "y2")
}

color = c("pink", "blue", "green")
title = c("Plots of the Largest Rc with size of 20", "Plots of the Largest Rc with size of 50", "Plots

for (h in 1:3){
  plot =
    ggplot(mindf[[h]], aes(x = mindf[[h]]$x, y = mindf[[h]]$y)) +
    geom_point(col = "black", size = 3) +
    geom_segment(data = connections[[h]],
                 aes(x = x1, y = y1, xend = x2, yend = y2), col = color[h]) +
    labs(x = "X-coordinate", y = "Y-coordinate", title = title[h])
  print(plot)
}
```




Plots of the Largest Rc with size of 50
Y−coordinate
X−coordinate
0
25
50
75
100


Plots of the Largest Rc with size of 100
Y−coordinate
X−coordinate
0
25
50
75
100

Conclusion

There are two factors we consider; The broadness, how broad Ad hoc should cover and the efficiency, how much power should Ad hoc should take from each node. As the result shows above, the more power which is begger radius, the more area being covered. It implies that Ad hoc can cover most of the new coming nodes if it is too far away. But if the only consideration of the Ad hoc is efficiency, then the smallest radius is the best way to set up. But it could lose potential nodes since the distance allowed to be connected to the network should be less than the smallest radius. Since there are two factor, one is random variable and one is the variable which can be controlled, it is better to set up the system based on the number of nodes which is random variable.
For the city which has lots of people, it is better to set up the radius in between the smallest and mean since the number of nodes is enough to cover all of the potential nodes. But for the city which has a few people, it is better to set up in between mean and the largest since the new coming node could be far away in terms of distance.