Project 1

```r
knitr::opts_chunk$set(echo = TRUE)
library(ggplot2)
```

R Markdown

1.getNodes() is a function to generate nodes in an ad hoc network.

```r
getNodes = function(n){

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

  return(Nodes)
}
```

test for getNodes() with set.seed()

```r
seeds = vector(length = 2, mode = "list")
set.seed(12345678)
seeds[[1]] = .Random.seed
a = getNodes(20)
a
```

```
##             x         y
## 1  87.4493237 29.982339
## 2  28.8790082 28.590068
## 3  94.7211719 41.237805
## 4  88.4520706 40.494738
## 5   0.8678354 38.975662
## 6  52.1891946 39.878085
## 7  39.8705438 47.910538
## 8  96.0019596 75.042876
## 9  81.3903614 30.976955
## 10  9.2448982 69.445912
## 11 22.5922084 93.313683
## 12 99.3158175  6.309057
## 13 22.4852071 37.157258
## 14 54.6598573 60.657133
## 15 83.7353629 95.841602
## 16 82.4399956 24.268543
## 17 86.4207304 62.634850
## 18 70.1360101 90.394736
## 19 99.4644627 69.639022
## 20 56.2150992 27.496029
```

```r
seeds[[2]] = .Random.seed
b = getNodes(20)
b
```

```
##             x         y
## 1  42.2683378 29.196525
```

```
## 2  59.7569945 12.608106
## 3   2.0711658  9.509243
## 4  21.3178845 52.901605
## 5  40.1602007 50.156703
## 6  85.1543210 45.360089
## 7  24.0498578 87.554870
## 8  64.2527189 15.973032
## 9   1.4492429 53.081874
## 10 76.3725982 65.129062
## 11 65.0854883 53.602606
## 12 75.3058456 47.912122
## 13 66.9060568 35.759815
## 14 79.4419097 12.203396
## 15 88.7776134  6.747329
## 16 29.4328171 90.223491
## 17 29.6154688 47.912027
## 18 81.8685511 62.517564
## 19 37.0807810 30.105379
## 20  0.1431841 81.965464
```

```r
c = list()
for(i in 1:5){
  seeds[[i]] = .Random.seed
  c[[i]] = getNodes(20)
}
c
```

```
## [[1]]
##            x         y
## 1  64.919092 34.356140
## 2  15.313280 86.181739
## 3  64.891547 83.271387
## 4  21.879852 56.106832
## 5  18.788627 66.519449
## 6  27.486245  1.733445
## 7   3.208993 78.660279
## 8  88.374019 47.425148
## 9  69.765118 43.476751
## 10 35.661908 74.361398
## 11 19.894567 55.727267
## 12 61.646260 58.249529
## 13 65.506061 34.141898
## 14 71.187210 26.443712
## 15 95.897494 12.257095
## 16 43.596716 65.516940
## 17 30.484897 49.245188
## 18 97.291839 35.791316
## 19 97.529571 34.798694
## 20  5.086186 42.379411
##
## [[2]]
##           x         y
## 1  16.77465  8.725073
## 2  27.45563 47.524396
## 3  41.12056  9.866489
```

```
## 4  12.98257 77.563325
## 5  84.67931 49.901314
## 6  95.06463 66.724499
## 7  72.74187 79.172510
## 8  10.71909 23.191722
## 9  48.42705 69.152383
## 10 57.53600 32.211071
## 11 44.15183 11.117543
## 12 88.90359 65.602297
## 13 64.07959 48.564311
## 14 73.17706 89.355198
## 15 64.72701 17.505839
## 16 40.63751 88.303744
## 17 42.27683 70.814781
## 18 77.22188 76.527847
## 19 15.85888 13.871029
## 20 25.32963 28.232158
##
## [[3]]
##            x        y
## 1  90.119710 38.83732
## 2  44.097766 67.74998
## 3   3.824228 37.00233
## 4  96.089834 22.99347
## 5  38.289030 76.85345
## 6  87.838459 33.23524
## 7   4.621997 54.64270
## 8  51.520630 25.64641
## 9  33.850495 69.48262
## 10 25.604178 47.17619
## 11 55.291446 24.88501
## 12 35.440153 89.61096
## 13 58.514281 28.76681
## 14 42.764461 83.17342
## 15 67.858405 63.24012
## 16 65.280825 45.67003
## 17 10.127053 42.02347
## 18 67.766806 50.17618
## 19  7.989750 88.84173
## 20 32.728735 80.92076
##
## [[4]]
##            x           y
## 1  52.899090 24.77550190
## 2  56.712010  8.23864497
## 3  61.562774 51.76472031
## 4  26.825110 37.79191659
## 5  61.954874 83.41220398
## 6  59.243435  3.65556320
## 7  34.012711 32.01756345
## 8  21.164636  0.01629377
## 9  45.203971 75.76292611
## 10  7.781972 12.77989536
## 11 34.914441 63.57215296
```

```
## 12 44.125929 32.23365829
## 13 34.421507 40.34413781
## 14 55.662606 68.06419110
## 15 83.533818 69.07512594
## 16 70.617004  8.25039274
## 17 53.975747 65.87651419
## 18 47.466958 48.14159947
## 19 94.054360 93.40203479
## 20 13.233958  3.22938995
##
## [[5]]
##            x         y
## 1  65.931957 49.171480
## 2  18.529969 16.243494
## 3  37.819706 17.794437
## 4  79.811760  5.040383
## 5  29.859357 25.394902
## 6   1.375787 63.184176
## 7  42.552435 11.549304
## 8  46.744285 73.582388
## 9  40.006651 92.958492
## 10 69.945810 23.013440
## 11 35.739172 42.090031
## 12 47.818529 90.494514
## 13  5.307165 44.247013
## 14  4.191299 90.942304
## 15  1.700623 55.575333
## 16 51.908387 35.486617
## 17 81.751332 41.235703
## 18 26.867522 10.953680
## 19 70.959155 87.065739
## 20 75.733151 97.488510
```

findTranMat() is a function to find the transition matrix based on a distance matrix and a value for R.

```
findTranMat = function(mat, R){
  n = dim(mat)[1]*dim(mat)[2]
  trans.mat = mat
  for(i in 1:n){
    trans.mat[i] = mat[i]<=R
  }
  trans.mat = apply(trans.mat, 1, "/", rowSums(trans.mat))
  return(trans.mat)
}
```

getEigen2() is a function to return the second largest eigenvalue of a matrix.

```
getEigen2 = function(mat){

  eigenvalues = eigen(mat)$values
  SecondLargest = eigenvalues[order(-eigenvalues)[2]]
  return(SecondLargest)

}
```

findRange() is a function to find the range of Rc to search over based on possible values.

```r
findRange = function(mat){
  n = dim(mat)[1]
  IgnoreDiag = mat
  diag(IgnoreDiag) = NA

  IgnoreDiag = IgnoreDiag[which(!is.na(IgnoreDiag))]

  newMat = matrix(IgnoreDiag,nrow = n, byrow = TRUE)

  Min = max(apply(newMat,1,min))
  Max = min(apply(mat,1,max))
  Min.Max = c(Min,Max)

  return(Min.Max)

}
```

2.(2)findRc() is a function to find the smallest radius, Rc, such that the network is completely connected.

```r
findRc = function(nodes, tol = 0.05){

  distBNodes1 = as.matrix(dist(nodes))

  lower = findRange(distBNodes1)[1]
  upper = findRange(distBNodes1)[2]
  Radius = sum(c(lower,upper))/2
  TranMat = findTranMat(distBNodes1,Radius)
  EigenValue = getEigen2(TranMat)


  while(abs(upper-lower) > tol){

    TranMat = findTranMat(distBNodes1,Radius)
    EigenValue = getEigen2(TranMat)

    if(Mod(EigenValue) != EigenValue){
      upper = Radius
    }else{
      lower = Radius
    }
    Radius = sum(c(lower,upper))/2
  }
  return(upper)

}
```

```r
x = c(0,0,1,1,10,10,11,11)
y = c(0,1,0,1,0,1,0,1)
yong = matrix(c(x,y),ncol = 2)
findRc(yong,tol = 1)
```

```
## [1] 10.04988
```

nodeDensity() is a function to take two inputs of numeric vectors of the same length, and returns a numeric vector of values that are proportional to node density at the (x,y) pairs.

```r
nodeDensity = function (x, y) {
# x is a numeric vector
# y is a numeric vector the same length as x
# z is returned.
# It is a numeric vector that provides the value of the node density function

# Check the inputs for the correct format
if (mode(x) != "numeric" | mode(y) != "numeric") stop("x and y must be numeric")
if (length(x) != length(y)) stop("x and y must be same length")

  a = 3
  b = 1
  band = 15
  bank = 1
  inBoundary = (0 <= x & x <= 100) &
               (0 <= y & y <= 100 & y < sqrt(110^2 - x^2))

  river = abs(sqrt(x^2 + y^2) - 60)
  hiArea = river> bank & river < band & inBoundary

  hiDensity = a * cos(river[hiArea] * pi / (2 * band)) + b

  z = b * inBoundary
  z[hiArea] = hiDensity
  z[river <= bank] = 0
  z
}
```

test for nodeDensity

```r
nodeDensity(41:60,rep(5,20))
```

```
##  [1] 1.000000 1.000000 1.000000 1.000000 1.086987 1.398100 1.704981
##  [8] 2.004297 2.292793 2.567329 2.824916 3.062745 3.278221 3.468991
## [15] 3.632967 3.768354 3.873667 3.947748 0.000000 0.000000
```

```r
c = getNodes(20)
nodeDensity(c[,1],c[,2])
```

```
##        1        2        3        4        5        6        7        8
## 1.000000 1.000000 0.000000 1.000000 1.876244 1.000000 1.000000 1.000000
##        9       10       11       12       13       14       15       16
## 0.000000 1.000000 1.946234 1.000000 1.000000 1.000000 1.000000 1.000000
##       17       18       19       20
## 1.000000 3.872171 1.000000 1.000000
```

test

```r
x = c(1, 2, 3)
y = c(5, 5, 2)

nodes3 = matrix(c(x,y), nrow = 3)

distBNodes = as.matrix(dist(nodes3))

# If R = 3.5 then
```

```r
findTranMat(distBNodes, 3.5)
```

```
##           1         2         3
## 1 0.5000000 0.5000000 0.0000000
## 2 0.3333333 0.3333333 0.3333333
## 3 0.0000000 0.5000000 0.5000000
```

```r
#returns
tranR3.5 = matrix(c(1/2, 1/2, 0, 1/3, 1/3, 1/3, 0, 01/2, 1/2),
                  byrow = TRUE, nrow = 3)


# and
getEigen2(tranR3.5)
```

```
## [1] 0.5
```

```r
#returns 0.5

# If R = 2 then
findTranMat(distBNodes, 2)
```

```
##     1   2 3
## 1 0.5 0.5 0
## 2 0.5 0.5 0
## 3 0.0 0.0 1
```

```r
#returns
tranR2 = matrix(c(1/2, 1/2, 0, 1/2, 1/2, 0, 0, 0, 1),
                byrow = TRUE, nrow = 3)

#And
getEigen2(tranR2)
```

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

```r
#returns 1

#Additionally,
findRange(distBNodes)
```

```
## [1] 3.162278 3.162278
```

```r
#returns approximately
# 3.16227766 3.16227766

#AND
findRc(nodes3, tol = 0.001)
```

```
## [1] 3.162278
```

```r
# Returns
# 3.162

################
#ANother example
nodes5 = matrix(c(1,3,2,1,3,3,3,2,0,0), nrow =5)
distBNodes5 = as.matrix(dist(nodes5))
```

```r
findTranMat(distBNodes5, 2)
```

```
##           1         2         3   4   5
## 1 0.3333333 0.3333333 0.3333333 0.0 0.0
## 2 0.3333333 0.3333333 0.3333333 0.0 0.0
## 3 0.3333333 0.3333333 0.3333333 0.0 0.0
## 4 0.0000000 0.0000000 0.0000000 0.5 0.5
## 5 0.0000000 0.0000000 0.0000000 0.5 0.5
```

```r
#returns
tranR2 = matrix(c(1/3, 1/3, 1/3, 0, 0,
                  1/3, 1/3, 1/3, 0, 0,
                  1/3, 1/3, 1/3, 0, 0,
                  0, 0, 0, 1/2, 1/2,
                  0, 0, 0, 1/2, 1/2),
                byrow = TRUE, nrow = 5)

getEigen2(tranR2)
```

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

```r
# returns 1

#Additionally,
findRange(distBNodes5)
```

```
## [1] 2.000000 2.236068
```

```r
#returns approximately
# 2   2.23608

#Also
tranR2.23 = findTranMat(distBNodes5, 2.23)
getEigen2(tranR2.23)
```

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

```r
# returns 1

#AND
findRc(nodes5, tol = 0.0001)
```

```
## [1] 2.236068
```

```r
# Returns approximately
#2.23608

# Note that with the tolerances we want to only check
# values that are 0.0001 apart in our Rc range
```

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

test the network with size 20

```r
Rc = c()
seeds = vector(length = 2, mode = "list")
set.seed(12345678)
n = 20
```

```r
networks = vector("list",1000)
for(i in 1:1000){
  seeds[[i]] = .Random.seed
  networks[[i]] = getNodes(n)
}
for(i in 1:1000){
  Rc[i] = findRc(networks[[i]])
}
Rc
```

```
##    [1] 64.24232 63.66989 66.03767 61.84596 68.60262 64.95301 68.32649
##    [8] 72.91352 64.91150 79.88781 66.33371 69.50506 67.57853 56.30006
##   [15] 59.86474 58.05912 60.81810 63.88150 58.85623 66.76791 59.92015
##   [22] 63.80753 49.69549 66.57619 56.70000 66.98719 56.66257 68.65215
##   [29] 58.44763 75.43611 65.71654 65.22771 67.24655 65.14783 54.47228
##   [36] 61.12652 61.52858 60.77461 68.49891 68.92252 59.75548 71.01398
##   [43] 66.15302 57.28601 63.21200 64.85416 71.70538 74.56849 70.13629
##   [50] 65.59519 69.34615 59.30609 71.30897 71.61983 68.21514 64.98694
##   [57] 61.58801 80.28118 55.90353 68.78909 65.96597 70.41603 63.89789
##   [64] 64.45382 58.54556 57.90118 45.38933 69.74280 62.15363 77.06341
##   [71] 77.25305 72.23096 64.90172 68.46104 56.35009 69.79085 65.98539
##   [78] 64.70804 54.70975 71.17529 65.62973 56.51522 58.26897 52.40495
##   [85] 69.87071 59.87870 55.52843 62.54086 63.98640 75.19244 69.52029
##   [92] 62.26186 62.76489 60.78772 64.81751 69.04845 66.85784 68.69205
##   [99] 60.65259 76.25663 61.47687 66.18695 58.33133 63.37002 61.82641
##  [106] 60.12611 66.24001 72.01036 60.13734 65.93787 65.09882 58.40588
##  [113] 62.45955 59.95631 66.54124 55.32103 77.13453 69.19347 63.98791
##  [120] 63.04582 65.05917 75.34717 76.61055 62.40342 56.15272 74.01922
##  [127] 64.83523 81.18300 54.01619 64.54605 57.99601 79.18810 59.96352
##  [134] 66.06212 61.38542 66.28425 72.08852 53.39148 63.36681 60.26313
##  [141] 56.48641 63.90212 63.18328 59.99894 69.39783 71.12389 57.37857
##  [148] 60.30408 61.32433 60.96653 62.09911 63.76139 63.93985 51.87409
##  [155] 56.55653 62.30844 60.57543 70.53851 66.13554 62.86966 60.13732
##  [162] 62.17337 60.54170 63.96366 65.83771 54.86565 63.28866 60.04834
##  [169] 73.13133 73.35005 65.34983 58.30896 63.17343 61.45077 70.53492
##  [176] 67.95321 73.09919 58.77565 56.32807 65.15601 60.27888 57.03298
##  [183] 68.33689 60.73080 64.03846 65.73148 52.56963 64.28952 61.95341
##  [190] 72.78707 66.12847 58.53040 69.42179 62.62491 66.31023 79.59616
##  [197] 56.73845 51.76609 68.90198 72.67737 64.97208 72.51505 57.10674
##  [204] 60.12669 56.65836 73.17670 68.07011 62.79921 50.47128 64.07672
##  [211] 66.78527 65.24825 73.97407 75.63990 67.23389 64.34396 71.84850
##  [218] 60.85106 61.15368 71.65716 67.45444 66.08815 63.40642 62.59431
##  [225] 63.98822 65.26334 60.64501 68.07740 58.90525 74.47771 71.84150
##  [232] 63.19015 65.09966 63.35317 54.23005 71.66856 66.91373 61.40753
##  [239] 59.80433 66.04459 67.28779 63.06814 66.56701 74.35592 72.77066
##  [246] 65.42976 59.15344 60.98086 66.51624 57.43896 64.09199 68.80101
##  [253] 72.66881 70.98828 60.34232 77.74724 62.73655 66.70854 74.43246
##  [260] 69.23805 65.60927 64.75066 62.22758 62.65740 59.81643 64.23576
##  [267] 68.61367 52.68659 59.86244 63.78477 64.78189 66.04169 59.11201
##  [274] 68.84354 65.17214 57.26479 87.25271 64.12019 66.73247 66.83477
##  [281] 66.98341 71.72993 63.91225 72.50967 59.72533 73.92052 54.83262
##  [288] 74.41722 63.86502 69.98753 68.77900 62.80075 67.99927 59.74503
##  [295] 59.93803 66.36390 60.02100 58.30068 66.91527 66.69634 71.78687
##  [302] 62.30977 74.39384 56.57887 69.71262 66.84807 58.11251 63.74750
```

```
## [309] 54.50080 58.06232 68.96440 55.94059 69.04212 66.09717 68.96090
## [316] 58.72185 61.19312 58.29897 65.54251 69.49538 74.53059 59.13658
## [323] 71.11135 82.40018 72.17936 65.25421 67.26669 69.76023 67.04213
## [330] 69.17611 81.07650 65.79613 65.36314 72.06802 60.15455 60.09430
## [337] 55.70125 60.16777 56.85518 60.54597 67.03832 57.51801 67.99484
## [344] 57.35224 68.84087 68.20311 61.49828 63.42733 66.74557 61.19305
## [351] 62.37285 71.03664 69.97953 54.16830 70.48138 63.64789 66.07804
## [358] 56.54817 75.94824 59.24317 65.39496 50.80368 57.70436 56.77647
## [365] 63.46063 67.05485 73.09294 73.07605 69.01083 66.72662 60.00871
## [372] 69.66500 68.02187 76.62104 60.75214 54.83601 71.79918 62.04114
## [379] 68.06524 65.23632 72.37480 71.56030 64.92732 61.13168 76.69416
## [386] 59.66576 57.01379 59.57914 60.03959 63.17195 57.65940 70.70047
## [393] 64.46896 61.54987 70.42578 57.93721 67.20389 68.45054 68.12209
## [400] 58.81614 67.64615 74.25819 58.12909 68.46867 57.06584 55.25103
## [407] 61.51276 62.37477 62.58133 59.41443 74.24790 62.10649 72.05665
## [414] 60.07972 63.43455 62.84947 52.47913 68.05900 66.61955 59.71139
## [421] 53.47659 60.49748 62.41542 63.88507 59.32227 81.81319 55.72219
## [428] 78.41011 60.22908 68.34370 64.55367 74.83136 59.54384 67.35132
## [435] 68.29987 67.76956 71.52291 67.33308 61.15173 69.85544 75.97537
## [442] 64.72932 68.32630 62.43793 65.39129 67.96840 60.19439 60.33649
## [449] 60.10712 63.11713 70.49136 68.81332 56.91493 62.33264 65.96718
## [456] 57.09345 74.30390 58.92160 74.17939 68.60351 62.89435 65.17183
## [463] 62.16958 67.80888 60.50470 62.65634 68.60805 54.18701 70.51099
## [470] 74.38493 67.24731 84.69583 61.96509 69.23421 66.65240 76.42914
## [477] 61.95925 69.57355 68.24265 56.76205 65.43455 63.49532 65.38171
## [484] 63.64466 67.02641 62.65605 72.11970 68.26894 69.04246 63.71958
## [491] 66.03501 72.10713 70.56210 59.44093 56.72980 68.94618 57.20350
## [498] 59.31840 66.00466 62.30388 70.09642 64.42605 62.30694 69.65973
## [505] 59.70506 61.26606 64.10204 72.76914 68.77826 69.83264 51.87498
## [512] 65.20165 60.78778 84.83986 61.58623 66.97018 61.35786 63.43556
## [519] 72.77610 64.78011 70.69360 74.57482 57.88796 59.13591 73.68217
## [526] 71.01134 60.57629 59.31248 76.25312 60.23637 67.34090 54.08981
## [533] 65.35900 65.16057 64.74418 70.48190 68.86124 64.13070 70.63192
## [540] 59.20872 56.95440 69.49915 57.01045 59.29683 59.78410 71.21157
## [547] 67.69838 69.66191 56.66255 58.75688 60.73377 56.41246 72.84128
## [554] 70.71032 63.34625 62.37356 73.11378 72.29301 67.19210 61.86464
## [561] 60.12799 73.96357 70.34082 59.91395 73.01148 57.91865 70.39394
## [568] 63.93635 62.57585 66.09729 67.35892 60.49768 54.25003 69.33212
## [575] 64.59793 61.61797 61.83119 62.66913 56.90238 64.56470 49.10215
## [582] 62.72856 66.49538 59.76274 67.77690 61.24398 68.27529 66.37838
## [589] 61.21428 70.52994 66.96716 67.42877 58.85605 69.49860 65.67811
## [596] 57.85405 63.99565 64.88104 56.40315 88.58908 61.65744 56.53313
## [603] 58.14841 71.82286 64.78875 56.11650 71.34062 55.58548 73.90398
## [610] 66.72230 62.60178 79.23070 69.78309 59.76921 62.66951 72.72142
## [617] 51.85376 58.23019 72.71836 61.04914 58.94363 54.31164 61.16531
## [624] 64.75092 62.74422 63.35868 67.85678 68.60180 63.26658 70.17572
## [631] 75.91203 66.09853 67.58642 66.94162 62.01328 52.18922 59.33617
## [638] 56.63025 88.28030 57.92014 70.81337 59.24754 59.43446 55.99775
## [645] 65.83083 59.55837 69.56136 59.44517 59.19809 61.01453 63.36140
## [652] 61.02619 72.44349 61.77058 67.06920 62.13544 66.45563 66.39336
## [659] 66.89710 65.28211 67.14270 70.66792 64.31208 64.21180 64.88147
## [666] 59.48762 71.84887 57.27572 58.34615 64.61497 56.57536 71.77466
## [673] 64.31548 52.86282 67.81810 59.60716 61.29909 65.81926 60.87014
## [680] 55.67804 72.44961 55.09957 65.54617 71.70067 61.77916 59.96285
```

```
## [687] 62.79887 64.92148 64.13807 71.98879 80.85872 61.57825 68.89544
## [694] 64.56043 72.01127 60.21473 68.62326 52.14242 60.54773 65.90040
## [701] 56.88872 62.20350 70.52771 64.26611 63.48694 71.10664 74.74515
## [708] 69.63426 73.66890 73.75075 63.49334 62.24833 62.45135 71.82621
## [715] 68.17191 65.06666 68.14559 65.33293 69.77835 62.11968 63.73063
## [722] 62.81549 64.70232 68.56557 65.62395 64.01912 61.02900 51.44272
## [729] 60.71051 85.81334 65.87588 55.41700 57.92054 64.48046 68.91751
## [736] 54.89127 66.74623 80.05584 64.18018 65.62289 66.97234 78.32667
## [743] 60.38606 67.79269 56.39082 73.65025 68.70783 77.92816 57.17992
## [750] 69.42685 58.91800 62.03922 58.79631 63.64604 67.70139 58.43272
## [757] 60.52572 62.06360 63.04950 68.14747 52.00061 64.96025 57.22896
## [764] 77.83842 66.88023 68.44027 57.19899 68.17279 63.41769 62.07687
## [771] 66.97914 74.83134 63.09985 63.84496 55.14006 68.26564 61.28347
## [778] 62.16625 62.35903 63.22377 60.03890 65.93997 65.09850 65.09274
## [785] 64.74151 61.66895 63.35723 57.41117 50.50313 70.91441 54.31645
## [792] 51.61254 69.83567 59.18525 58.66087 58.07912 65.27762 58.89276
## [799] 60.75568 59.23653 63.37067 66.09536 63.19758 62.94282 64.59606
## [806] 75.01712 62.63051 62.93524 59.70557 60.02120 69.62872 69.81181
## [813] 65.10913 75.41166 49.16841 76.33998 64.29002 66.14850 59.29347
## [820] 56.67837 66.79685 72.97875 63.52068 67.77030 57.41806 58.26300
## [827] 76.13575 64.46955 64.45738 59.16069 59.37492 76.88707 71.61434
## [834] 57.62404 71.10321 64.52974 65.87144 62.19840 59.14697 68.39092
## [841] 57.36638 58.92206 56.19700 60.85279 79.77108 64.14985 67.83652
## [848] 68.82666 62.79292 62.07381 68.95547 53.89261 66.64599 66.95096
## [855] 65.60938 56.27152 66.18343 67.46293 66.63715 57.99717 60.69205
## [862] 65.92362 68.80460 73.97262 59.38071 62.66127 57.48017 73.34829
## [869] 65.76465 56.04350 68.32720 75.06929 69.95088 64.16747 65.42636
## [876] 68.72884 54.41299 71.35975 72.82316 66.45728 65.19658 54.45538
## [883] 74.47378 69.78547 71.88799 69.13281 65.66940 65.05912 65.07888
## [890] 83.31621 70.12053 66.56569 60.93913 65.97573 63.16888 64.78045
## [897] 65.40537 64.52015 74.00648 61.86218 83.63864 59.10223 69.08119
## [904] 65.00749 76.83741 69.90809 70.09359 63.69182 65.32484 65.24943
## [911] 69.94703 68.66566 61.15157 59.63709 61.29963 65.52775 59.22285
## [918] 63.62071 68.87500 61.61893 63.02722 70.44805 55.86694 66.62102
## [925] 72.58109 72.39647 61.22262 69.13203 62.10161 70.20238 70.01147
## [932] 62.28928 65.12990 73.33089 75.96248 65.71435 52.91603 56.76931
## [939] 68.34843 66.12372 78.78424 67.80939 68.07685 67.32325 62.10315
## [946] 65.88079 69.36436 55.09798 68.35264 61.97934 61.92107 70.38591
## [953] 62.51091 65.96922 56.32055 68.24167 54.31139 66.41309 70.23022
## [960] 55.45582 64.90401 62.90074 58.57006 55.65794 58.18999 57.81387
## [967] 66.41037 56.24620 60.71103 65.23819 66.12488 67.99126 65.36922
## [974] 53.29756 64.49481 61.07559 57.05882 67.58637 56.54562 72.15157
## [981] 53.89384 66.45003 56.08140 71.92557 77.19249 59.15641 70.29368
## [988] 63.75541 73.08403 70.46927 65.40799 63.07230 62.76513 58.13049
## [995] 69.56575 64.81103 70.11246 56.85388 71.30532 67.75189
```

test for the network with size 50

```
Rc2 = c()
seeds = vector(length = 2, mode = "list")
set.seed(12345678)
m = 50
networks2 = vector("list",1000)
for(i in 1:1000){
  seeds[[i]] = .Random.seed
```

```r
  networks2[[i]] = getNodes(m)
}
for(i in 1:1000){
  Rc2[i] = findRc(networks2[[i]])
}
Rc2
```

```
##    [1] 67.97465 53.94221 65.98844 70.34261 63.55861 61.78264 63.09170
##    [8] 64.03471 64.73292 65.21047 73.87765 68.99696 66.96354 66.84837
##   [15] 57.11390 73.12871 67.64326 63.26794 66.86416 70.92538 63.27401
##   [22] 71.42209 66.51836 71.63556 62.69402 65.24153 63.11573 57.05548
##   [29] 74.37610 69.73165 75.44356 59.89089 64.00573 63.63962 70.11032
##   [36] 69.55173 64.16968 67.49262 69.39708 61.38848 69.70316 65.70826
##   [43] 63.22660 70.49973 65.42968 57.63328 61.39314 71.42438 65.65585
##   [50] 67.86468 67.76611 71.24200 69.35044 66.71732 69.80672 66.37624
##   [57] 70.30954 60.23361 69.33738 59.24168 61.00479 65.61188 68.76227
##   [64] 64.15518 66.99202 54.97749 60.21364 69.22544 68.41513 64.61633
##   [71] 62.75879 64.89662 64.12256 60.86582 63.74965 65.15956 62.26226
##   [78] 67.86950 61.68513 64.90860 67.26266 69.89655 67.86115 68.61020
##   [85] 57.12261 66.98580 64.64072 65.62230 68.97535 66.85649 65.02917
##   [92] 66.02181 63.65403 71.63560 67.84368 69.78907 75.65119 70.05773
##   [99] 72.08830 62.54798 68.54256 64.08659 71.28779 70.81007 70.73794
##  [106] 66.67933 60.81108 64.99905 74.12058 72.36763 60.81047 62.28042
##  [113] 65.35860 67.78073 73.25278 66.76161 60.97975 67.49493 68.93050
##  [120] 62.16627 61.56470 60.72974 61.88070 62.70842 63.65665 58.41339
##  [127] 67.35285 69.55873 70.72011 79.96980 67.24474 60.93183 72.31699
##  [134] 65.81292 62.57925 68.29952 67.51321 69.09283 64.30746 70.35584
##  [141] 66.13463 64.20328 61.97532 66.21857 61.77824 63.69260 71.92429
##  [148] 62.31263 63.71468 66.92785 63.81952 66.68671 65.86751 69.52652
##  [155] 60.00294 69.14182 64.33574 68.95587 68.56902 64.20384 66.23144
##  [162] 69.91721 60.95961 63.35060 72.64710 64.93027 60.81544 68.77315
##  [169] 64.71482 59.65044 66.33681 65.98671 68.64988 69.68149 77.97649
##  [176] 68.33597 74.89319 64.35662 64.52448 64.99544 75.76562 76.16286
##  [183] 65.81327 68.20164 60.28492 63.60078 68.27589 71.08451 71.91355
##  [190] 73.91918 66.04302 64.70979 69.48862 59.47092 66.85092 66.86031
##  [197] 70.93599 68.98873 75.34331 61.38620 63.63525 64.95917 71.81078
##  [204] 64.51897 62.32199 65.43029 74.75558 71.55908 67.47698 61.58451
##  [211] 63.71319 72.87940 63.88864 60.85614 68.42937 65.24790 69.85410
##  [218] 59.71143 68.29649 63.51898 67.15783 68.99977 67.85141 64.84718
##  [225] 65.88244 69.35745 68.30898 67.67442 64.18071 67.72494 60.83540
##  [232] 64.38375 63.11715 69.76439 60.25550 62.35154 64.67550 68.67220
##  [239] 58.29228 63.42600 66.45621 60.14196 64.48478 72.66253 68.27908
##  [246] 75.09176 69.06505 64.75835 63.21670 63.28106 72.66838 66.29154
##  [253] 74.95663 66.36977 57.12675 63.69355 68.82204 61.96343 62.43304
##  [260] 66.12247 67.56874 67.63050 70.02983 61.36176 68.73825 64.49875
##  [267] 74.18562 70.09230 71.47899 64.86784 70.38139 61.13105 61.37937
##  [274] 68.47298 63.68182 68.76132 72.33407 66.54069 65.79605 65.37413
##  [281] 69.20732 64.88507 65.80976 66.55692 63.53407 67.47891 66.67882
##  [288] 64.66233 63.07870 73.81032 67.84070 66.90930 65.34833 65.08444
##  [295] 69.74771 63.22243 73.26989 64.79280 71.07999 68.94545 71.25250
##  [302] 62.64900 61.32493 62.50636 61.27190 69.33740 71.06746 63.33207
##  [309] 66.42524 64.20735 65.94054 59.94655 66.97837 61.69716 57.57520
##  [316] 66.96444 63.35078 68.33623 70.89640 62.25150 68.35312 63.75447
##  [323] 62.39638 67.88473 71.85859 59.82057 66.46520 59.54236 66.54880
```

```
## [330] 72.31033 60.20165 63.84365 70.98626 67.03939 58.54299 69.17300
## [337] 73.67763 74.17961 65.84412 69.07549 63.12924 69.59405 71.68533
## [344] 65.91860 64.04756 70.32632 68.42189 67.07073 68.66588 64.63217
## [351] 61.99755 66.86466 67.23444 74.21522 61.88032 64.06736 71.34556
## [358] 70.96853 66.11351 63.70492 68.92791 57.06318 71.55384 70.78233
## [365] 65.96776 60.29849 64.30806 67.52307 64.82283 68.31633 68.36937
## [372] 66.94656 69.98792 67.01834 63.35195 69.85641 67.57606 67.30788
## [379] 73.07845 64.74838 63.84063 69.22957 73.95907 67.45730 71.05295
## [386] 57.95636 65.68462 66.57352 61.46277 64.43952 59.96312 65.67820
## [393] 66.35739 72.90652 71.94195 68.59827 63.90112 63.98233 66.00798
## [400] 67.24335 66.74742 66.93375 60.11291 69.28950 59.55957 66.91301
## [407] 65.10516 62.60179 58.65205 64.42221 71.45863 78.41078 61.81774
## [414] 64.46254 65.62461 67.43994 72.83066 71.10092 68.39221 61.30909
## [421] 62.33242 67.85731 69.11978 64.48055 65.36918 67.42588 69.41277
## [428] 67.39221 65.44453 68.14674 67.05228 65.64875 65.90250 61.12445
## [435] 62.78445 69.51852 61.24410 67.37754 65.35703 67.03955 59.80593
## [442] 64.70790 59.92532 67.44976 61.50278 67.30971 67.59115 69.35839
## [449] 67.18829 65.37155 66.70785 69.08018 63.51348 67.57415 63.88028
## [456] 69.79654 60.73039 76.30591 69.11800 82.58083 57.41447 78.71781
## [463] 63.95894 68.16049 63.71669 68.16130 66.30971 66.20580 65.10939
## [470] 63.00105 59.25165 72.06718 65.26470 72.90237 65.29996 61.00361
## [477] 74.80997 70.34619 78.05705 63.05765 64.20457 64.96950 66.43512
## [484] 61.97330 65.84570 72.35359 67.00449 61.65083 67.25009 65.32301
## [491] 62.45286 63.52567 72.33335 65.53129 70.40551 79.72261 77.64063
## [498] 66.40247 60.77471 66.13226 65.98661 65.73583 61.81123 62.99284
## [505] 66.52425 64.37938 60.61586 69.91245 68.85307 64.30033 59.40444
## [512] 65.06305 62.17057 67.17960 71.67699 75.46307 68.85567 75.51547
## [519] 77.23412 63.64189 63.61133 62.20118 61.14461 65.10636 68.57644
## [526] 69.25698 70.56028 56.62511 61.60394 65.77280 68.79255 64.46518
## [533] 64.13726 70.57420 62.72681 68.60789 67.87853 66.72968 69.50442
## [540] 73.61181 70.30513 58.75829 67.32076 70.12398 66.12660 68.56522
## [547] 69.19585 64.96702 64.80109 70.03818 68.75697 74.70056 77.36353
## [554] 53.16008 64.49981 64.69282 63.34795 57.11625 64.05842 65.12479
## [561] 58.99654 60.83565 74.91697 62.82643 70.63363 72.63957 62.22066
## [568] 74.99297 75.01825 70.41104 64.62412 66.71870 63.89893 60.23627
## [575] 65.67762 62.39585 63.90317 66.26201 68.23508 67.31500 65.02528
## [582] 66.26743 68.94115 67.83074 68.21717 65.34771 59.34120 57.56373
## [589] 72.71502 68.55127 67.82864 70.12390 62.79402 68.39709 60.11929
## [596] 71.09758 60.26956 66.36449 74.30890 66.29982 67.94040 64.64280
## [603] 67.89107 58.82092 59.34098 71.26296 71.75789 68.96948 67.57954
## [610] 65.58159 65.49289 67.69895 64.76562 70.57805 70.07566 74.02906
## [617] 68.53343 68.60100 62.69350 64.11535 64.64707 68.18285 71.00036
## [624] 63.88872 75.29336 72.34196 66.55461 72.95019 69.80006 65.15655
## [631] 70.68845 69.98168 65.33711 65.92163 67.13837 76.13837 67.87898
## [638] 65.67475 64.36202 62.11970 59.19467 58.87678 61.26925 70.13222
## [645] 63.20391 59.32410 69.59508 66.11963 57.47540 63.59705 65.93653
## [652] 68.14127 63.02187 66.07165 73.03361 64.11676 73.27815 65.66572
## [659] 64.19539 65.93070 61.50957 60.74917 65.88199 60.73106 58.22003
## [666] 69.77177 68.36199 61.91776 64.62407 65.60974 72.57938 67.74631
## [673] 73.61002 67.47020 63.89805 63.39795 61.76649 67.89100 62.26949
## [680] 60.78228 72.43814 65.66838 67.65593 64.22495 67.80945 77.11892
## [687] 61.20110 75.42466 67.08168 67.29458 68.72317 72.82901 64.67000
## [694] 65.54068 63.03748 69.92291 64.51187 72.84497 61.50653 70.11522
## [701] 68.58157 63.99309 63.07531 59.90756 65.39043 65.03204 78.18464
```

```
## [708] 61.48956 65.75998 67.54892 61.11570 67.12995 64.94863 69.48718
## [715] 65.82595 63.87451 71.94974 68.01270 65.87553 62.67890 64.17537
## [722] 70.31861 72.70515 63.39689 62.85541 66.84728 57.91498 77.04269
## [729] 62.37133 67.73606 69.21038 63.18203 62.43903 62.48144 71.78739
## [736] 73.61387 68.06121 64.95946 66.68995 69.28555 68.17049 66.63380
## [743] 61.44396 64.36410 70.17941 66.70677 68.32561 68.84319 66.80229
## [750] 63.08362 69.84646 63.03466 64.90127 60.66986 67.65934 67.81607
## [757] 68.18217 57.93546 67.44757 63.30860 66.34947 67.39617 60.31041
## [764] 75.65491 64.46009 63.13636 66.00922 67.32085 65.20805 66.56606
## [771] 71.98860 65.43564 63.86769 63.23145 65.96062 74.24790 68.25511
## [778] 63.91988 66.70126 73.40900 71.12283 67.08245 65.95766 66.18041
## [785] 63.57536 74.60513 74.25450 67.39638 64.79725 70.05042 64.83009
## [792] 67.26340 64.06582 67.26032 60.88677 59.18401 71.61504 66.56587
## [799] 65.27329 66.05334 71.23062 61.57349 70.44912 57.71097 68.05382
## [806] 70.36145 63.84100 62.08125 58.63671 69.61508 73.26107 63.30828
## [813] 71.70644 66.78055 73.73910 63.72747 71.43387 59.38940 68.67102
## [820] 67.66601 67.68208 72.13028 65.82373 67.97633 61.69639 73.69590
## [827] 71.52983 67.35400 71.51479 70.29739 61.81768 70.74919 68.84398
## [834] 67.67096 67.80653 72.72309 66.60921 61.83303 66.73219 62.98482
## [841] 68.94865 71.06799 66.68730 66.01889 65.82186 66.07790 63.69857
## [848] 65.51675 71.95755 63.68366 58.75742 67.88882 59.91925 63.62838
## [855] 74.25505 67.55613 67.08072 60.80469 67.67781 62.66349 68.10180
## [862] 71.58860 65.26926 61.59450 69.93999 74.23334 65.98636 70.71833
## [869] 72.64329 75.49339 62.70942 63.71798 71.88768 64.31527 68.74355
## [876] 67.18980 70.49276 67.24699 70.78799 62.12816 65.73672 69.86003
## [883] 58.62620 66.91127 65.17429 67.77561 68.74322 65.43608 66.33441
## [890] 64.47311 62.90301 63.67598 65.16910 66.33903 67.11485 61.16211
## [897] 70.20565 63.50358 60.59787 56.64177 74.52377 69.20126 70.35063
## [904] 60.53481 59.08550 64.23687 63.05771 58.35372 71.69479 63.84184
## [911] 63.18953 74.41766 72.81036 73.88946 68.77600 73.47677 66.70601
## [918] 68.79016 73.34400 72.88905 74.86476 76.63822 66.85181 59.48790
## [925] 69.76076 72.16410 61.36500 67.23390 67.07602 72.85564 66.32386
## [932] 70.41072 60.47500 66.00550 62.62447 70.37292 64.51662 64.77397
## [939] 68.30532 66.99954 68.40388 69.10039 68.20330 60.75913 66.44545
## [946] 66.92534 67.20526 62.49837 72.12879 65.60387 66.07127 66.35304
## [953] 68.36038 62.69866 68.66268 64.60245 69.09867 58.50308 63.66548
## [960] 66.43194 66.20387 69.08615 64.81167 75.75844 60.73122 74.51828
## [967] 67.25095 68.64319 62.08016 67.23911 60.68325 63.75601 65.24506
## [974] 65.36624 63.61891 57.48322 59.98681 68.80308 63.66738 68.15772
## [981] 70.72230 63.68983 59.22288 66.77399 67.02262 62.19288 64.35343
## [988] 62.57194 71.01002 70.60219 66.29651 65.22503 71.24115 71.27330
## [995] 57.87332 75.74332 69.51179 74.46143 65.56948 66.45213
```

plot for comparing Rc with different size

```r
plotRc = function(Rc1, Rc2 = NULL, bw = 2,
           title = 'Compare Rc with different size')
{
    Rcs = data.frame(
      Rc12 = c(Rc1, Rc2),
      strat = rep(c(1,2),
                  times = c(length(Rc1), length(Rc2)))
      )
    require(ggplot2)
    ggplot(data = Rcs) +
```

```r
      geom_density(aes(x = Rc12, 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), name = 'Density', labels = c('Rc1','Rc2'))
}
```

comparison Rc and RcTwo with different size

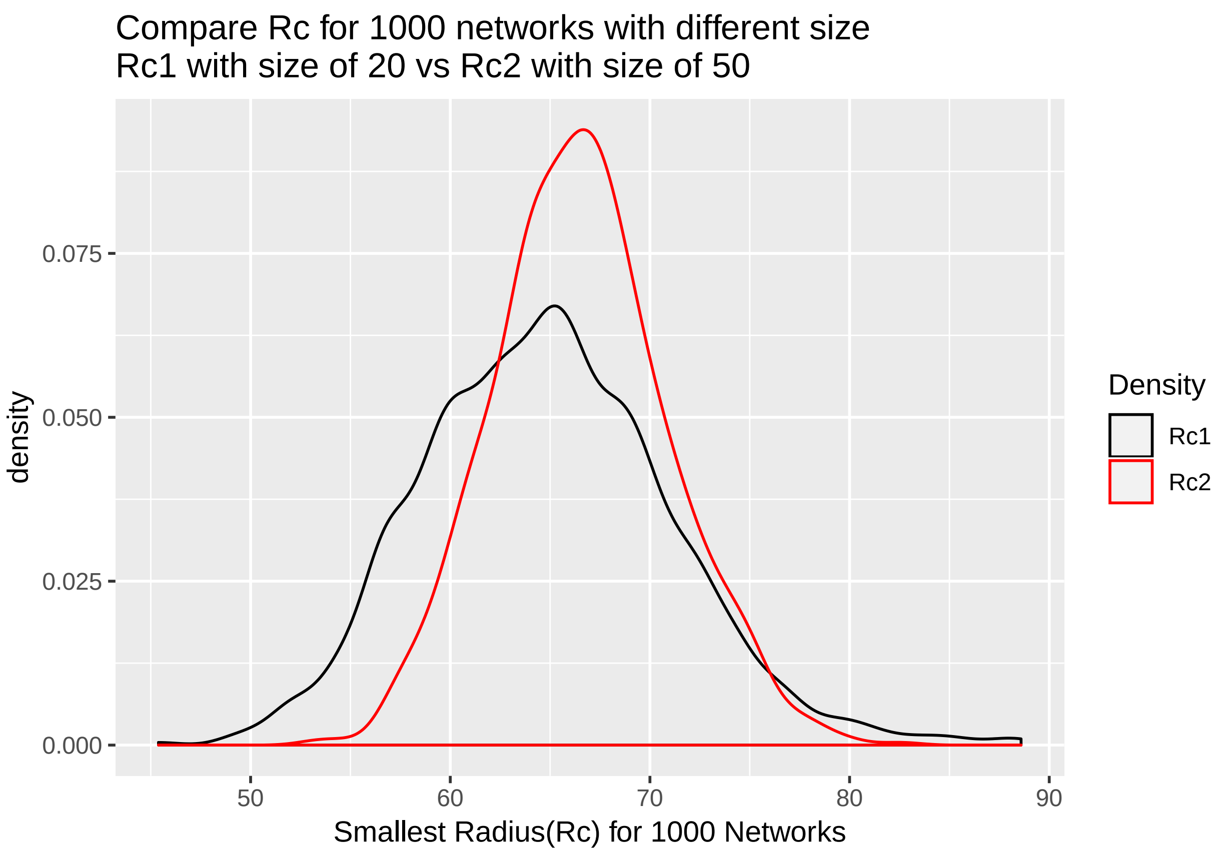
```r
summary(Rc)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   45.39   60.30   64.75   64.81   68.79   88.59
```

```r
summary(Rc2)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   53.16   63.67   66.43   66.55   69.21   82.58
```

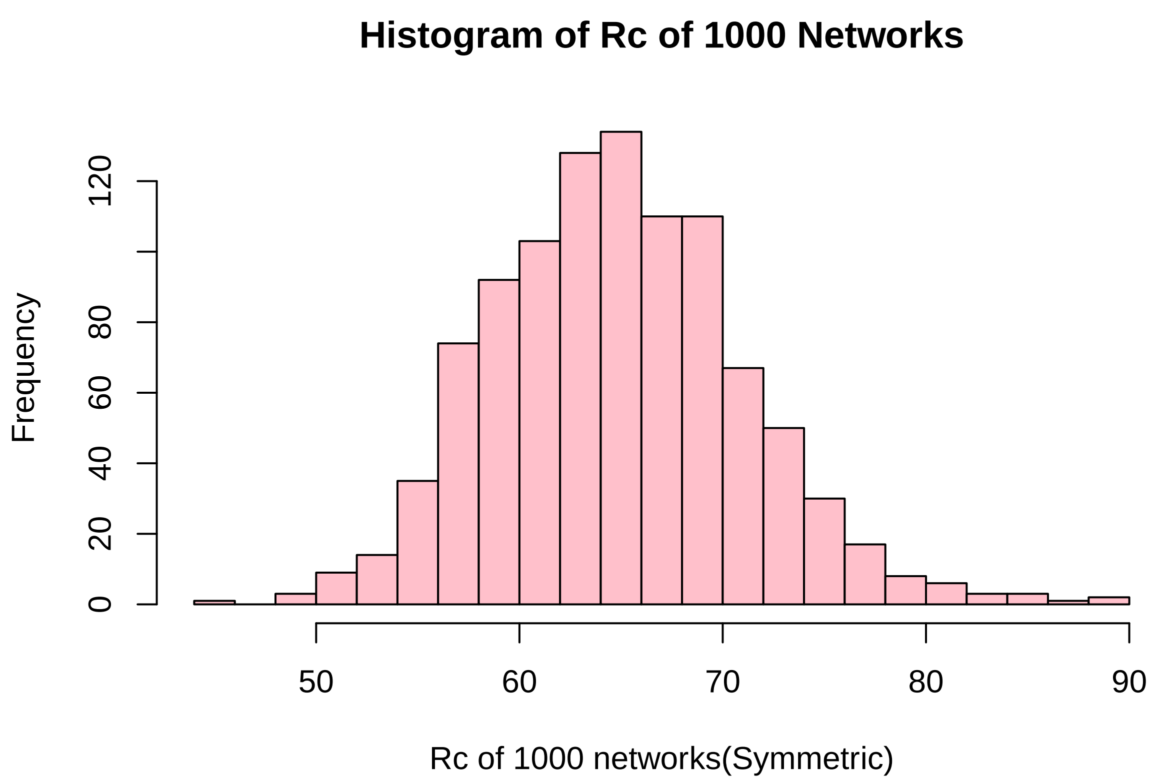
```r
plotRc(Rc, Rc2, bw = 1,
          title = "Compare Rc for 1000 networks with different size\nRc1 with size of 20 vs Rc2 with si
```




```r
summary(Rc)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   45.39   60.30   64.75   64.81   68.79   88.59
```

```r
Histogram = hist(Rc, breaks = 20, col = "pink", xlab = "Rc of 1000 networks(Symmetric)", main = "Histog
```




```
print("The the distribution of Rc of 1000 Networks is 'Symmetric'.")
```

```
## [1] "The the distribution of Rc of 1000 Networks is 'Symmetric'."
```

```
Histogram
```

```
## $breaks
##  [1] 44 46 48 50 52 54 56 58 60 62 64 66 68 70 72 74 76 78 80 82 84 86 88
## [24] 90
##
## $counts
##  [1]   1   0   3   9  14  35  74  92 103 128 134 110 110  67  50  30  17
## [18]   8   6   3   3   1   2
##
## $density
##  [1] 0.0005 0.0000 0.0015 0.0045 0.0070 0.0175 0.0370 0.0460 0.0515 0.0640
## [11] 0.0670 0.0550 0.0550 0.0335 0.0250 0.0150 0.0085 0.0040 0.0030 0.0015
## [21] 0.0015 0.0005 0.0010
##
## $mids
##  [1] 45 47 49 51 53 55 57 59 61 63 65 67 69 71 73 75 77 79 81 83 85 87 89
##
## $xname
## [1] "Rc"
##
## $equidist
## [1] TRUE
##
## attr(,"class")
## [1] "histogram"
```

find the 4 Rc values that coincide with (or are close to) the minimum, maximum, median, and mean Rc value

```
Min = min(Rc)
Median = median(Rc)
Mean = mean(Rc)
Max = max(Rc)

Min
```

```
## [1] 45.38933
```

```
Median
```

```
## [1] 64.74742
```

```
Mean
```

```
## [1] 64.80816
```

```
Max
```

```
## [1] 88.58908
```

```
i = which(abs(Rc-Min)<=0.005)
j = which(abs(Rc-Median)<=0.005)
k = which(abs(Rc-Mean)<=0.005)
l = which(abs(Rc-Max)<=0.005)
if(length(i)>1 | length(j)>1 | length(k)>1 | length(l)>1){
  i = i[1]
  j = j[1]
  k = k[1]
  l = l[1]
}
CloseMin = Rc[i]
CloseMedian = Rc[j]
CloseMean = Rc[k]
CloseMax = Rc[l]

c(i, CloseMin)
```

```
## [1] 67.00000 45.38933
```

```
c(j, CloseMedian)
```

```
## [1] 262.00000  64.75066
```

```
c(k, CloseMean)
```

```
## [1] 996.00000  64.81103
```

```
c(l, CloseMax)
```

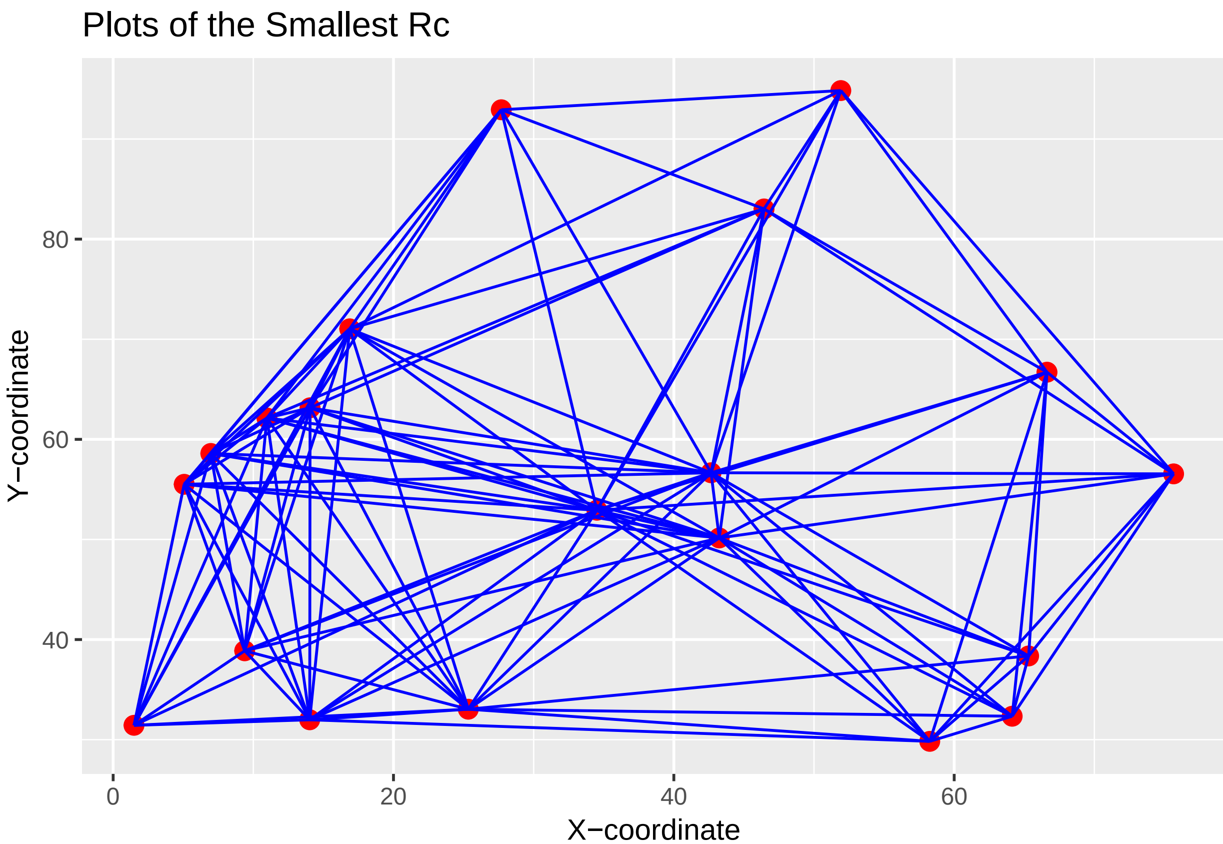
```
## [1] 600.00000  88.58908
```

test for Min Median Mean Max Rc with size of twenty of the network

```
mindf = data.frame(networks[i])
comb = combn(nrow(mindf), 2)
connected = comb[,which(as.numeric(dist(networks[[i]]))<=Rc[i])]
connections = data.frame(
  from = mindf[connected[1, ], 1:2],
  to = mindf[connected[2, ], 1:2]
```

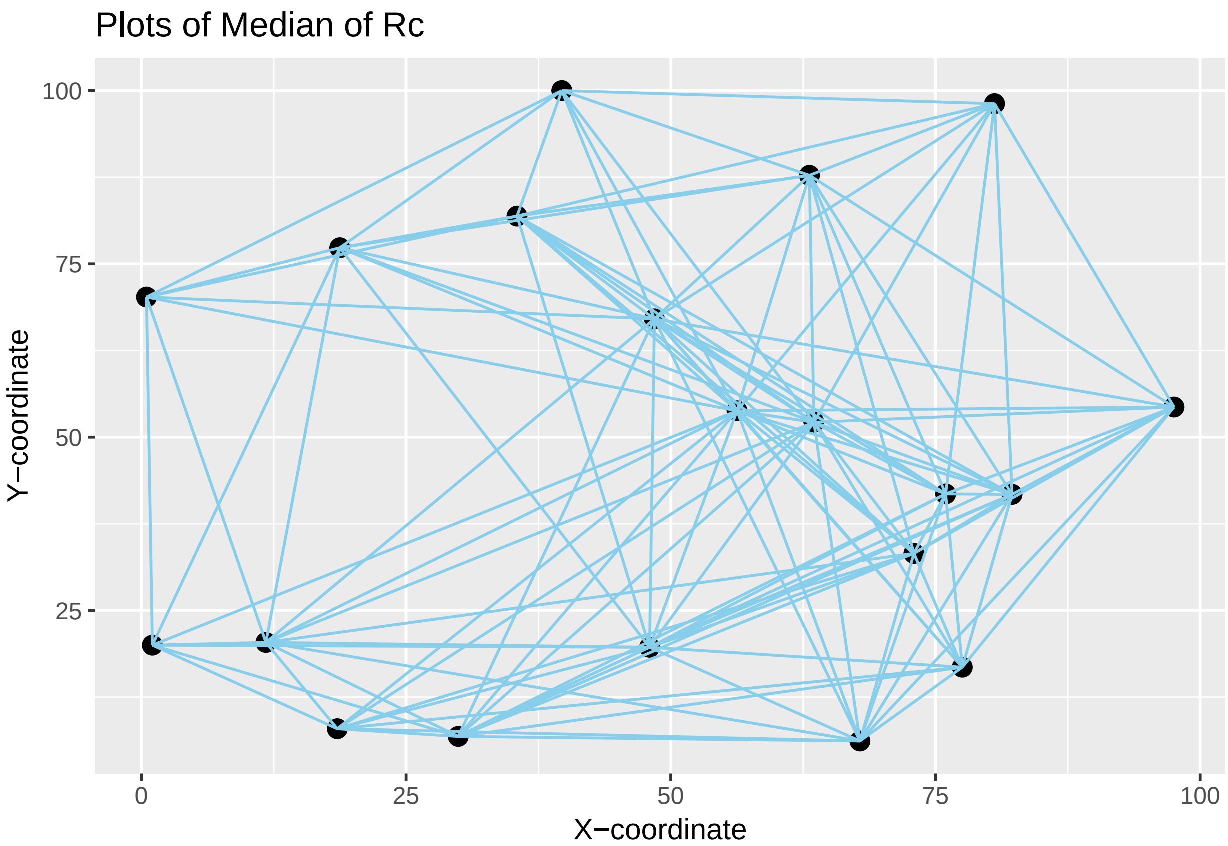
```
)
names(connections) = c("x1", "y1", "x2", "y2")

plot1 = ggplot(mindf,aes(x=mindf$x,y=mindf$y)) +
  geom_point(col = "red", size = 3) +
  geom_segment(data = connections, aes(x=x1, y=y1, xend=x2, yend=y2), col="blue") +
  labs(x = "X-coordinate", y = "Y-coordinate", title = "Plots of the Smallest Rc")
plot1
```



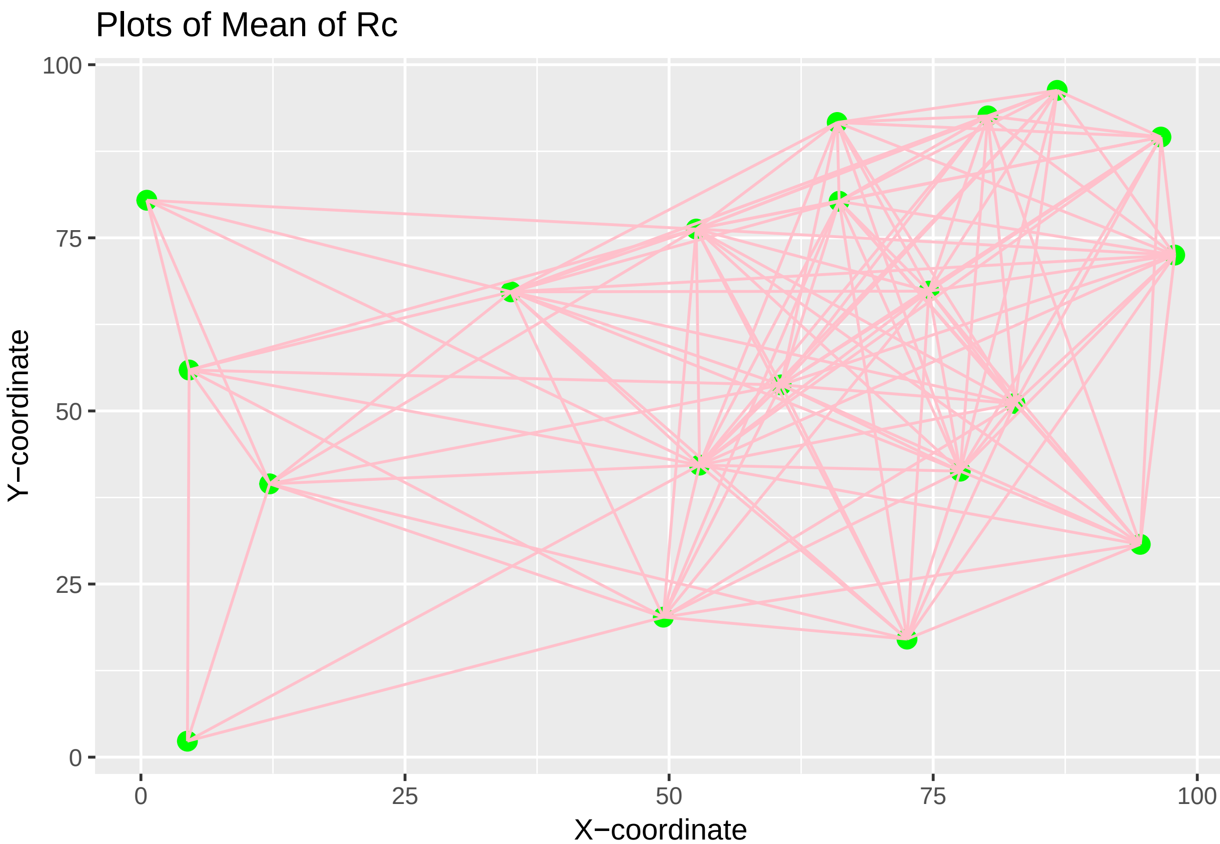

```
mediandf = data.frame(networks[j])
comb2 = combn(nrow(mediandf), 2)
connected2 = comb2[,which(as.numeric(dist(networks[[j]]))<=Rc[j])]
connections2 = data.frame(
  from = mediandf[connected2[1, ], 1:2],
  to = mediandf[connected2[2, ], 1:2]
)
names(connections2) = c("x1", "y1", "x2", "y2")

plot2 = ggplot(mediandf,aes(x=mediandf$x,y=mediandf$y)) +
  geom_point(col = "black", size = 3) +
  geom_segment(data = connections2, aes(x=x1, y=y1, xend=x2, yend=y2), col="skyblue") +
  labs(x = "X-coordinate", y = "Y-coordinate", title = "Plots of Median of Rc")
plot2
```



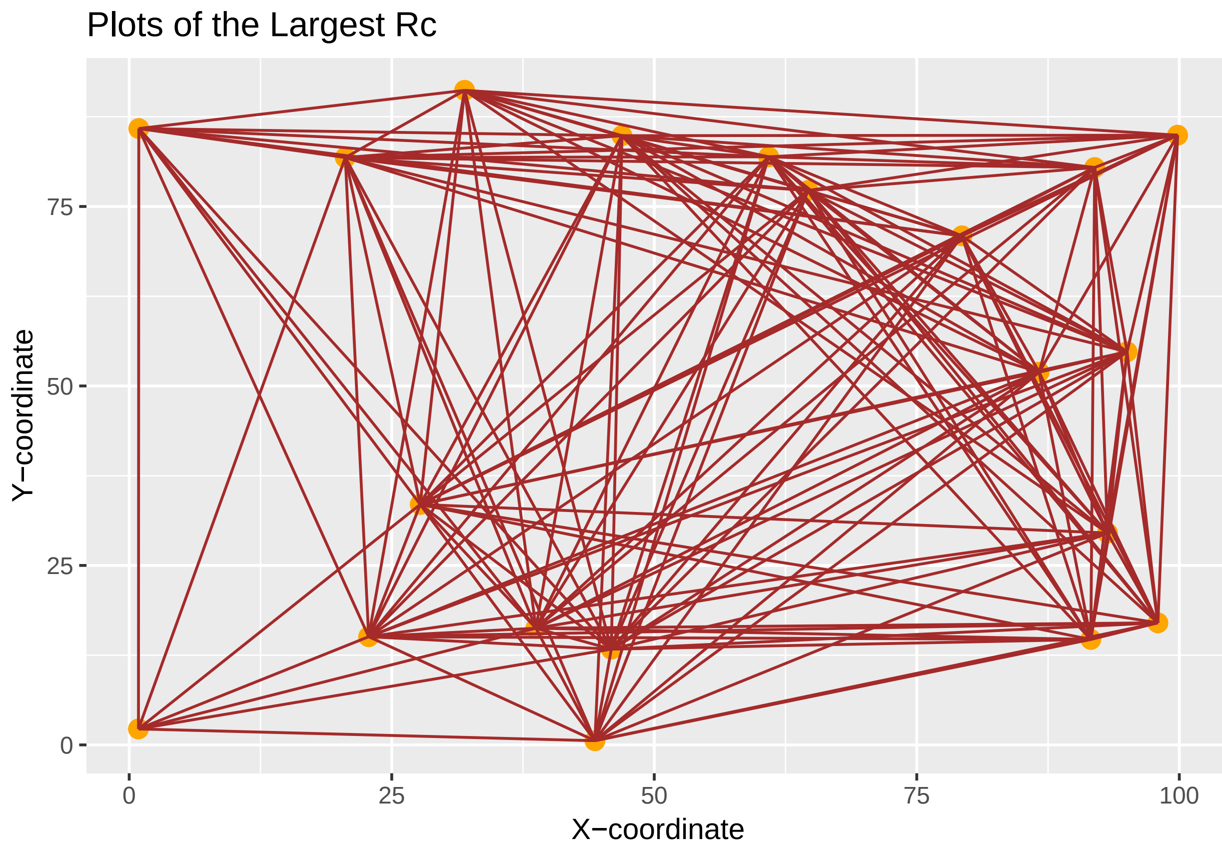

```
meandf = data.frame(networks[k])
comb3 = combn(nrow(meandf), 2)
connected3 = comb[,which(as.numeric(dist(networks[[k]]))<=Rc[k])]
connections3 = data.frame(
  from = meandf[connected3[1, ], 1:2],
  to = meandf[connected3[2, ], 1:2]
)
names(connections3) = c("x1", "y1", "x2", "y2")

plot3 = ggplot(meandf,aes(x=meandf$x,y=meandf$y)) +
  geom_point(col = "green", size = 3) +
  geom_segment(data = connections3, aes(x=x1, y=y1, xend=x2, yend=y2), col="pink") +
  labs(x = "X-coordinate", y = "Y-coordinate", title = "Plots of Mean of Rc")
plot3
```




```
maxdf = data.frame(networks[l])
comb4 = combn(nrow(maxdf), 2)
connected4 = comb[,which(as.numeric(dist(networks[[l]]))<=Rc[l])]
connections4 = data.frame(
  from = maxdf[connected4[1, ], 1:2],
  to = maxdf[connected4[2, ], 1:2]
)
names(connections4) = c("x1", "y1", "x2", "y2")

plot4 = ggplot(maxdf,aes(x=maxdf$x,y=maxdf$y)) +
  geom_point(col = "orange", size = 3) +
  geom_segment(data = connections4, aes(x=x1, y=y1, xend=x2, yend=y2), col="brown") +
  labs(x = "X-coordinate", y = "Y-coordinate", title = "Plots of the Largest Rc")
plot4
```


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