**1. Create random networks**

**(a) Create three undirected random networks with 1000 nodes, and the probability p for drawing an edge between two arbitrary vertices 0.01, 0.05 and 0.1 respectively. Plot the degree distributions.**

|  |
| --- |
| p=0.01 |
| rnd\_G\_1 <- random.graph.game(1000,p=0.01,directed=F)  hist(degree(rnd\_G\_1)) |
|  |

|  |
| --- |
| p=0.05 |
| rnd\_G\_2 <- random.graph.game(1000,p=0.05,directed=F)  hist(degree(rnd\_G\_2)) |
|  |

|  |
| --- |
| p=0.1 |
| rnd\_G\_3 <- random.graph.game(1000,p=0.1,directed=F)  hist(degree(rnd\_G\_3)) |
|  |

**(b) Are these networks connected or disconnected? What are the diameters of these networks?**

**Networks connected or disconnected?**

if edge.connectivity value is 0 -> Disconnected

edge.connectivity(rnd\_G\_1)

|  |  |  |
| --- | --- | --- |
| p | Edge Connectivity | Connection |
| 0.01 | 3 | Connected |
| 0.05 | 27 | Connected |
| 0.1 | 71 | Connected |

Diameter

diameter(rnd\_G\_1)

|  |  |
| --- | --- |
| p | diameter |
| 0.01 | 5 |
| 0.05 | 3 |
| 0.1 | 3 |

**(c) Try to numerically find a value pc (to three significant figures), so that when p < pc the generated random networks are disconnected, and when p > pc the generated random networks are connected.**

graph.density(rnd\_G\_1)

|  |  |  |
| --- | --- | --- |
| p | pc |  |
| 0.01 | 0.01026 | disconnected |
| 0.05 | 0.05011 | disconnected |
| 0.1 | 0.09985 | connected |

**(d) Can you analytically derive the value of pc?**

**2. Create a network with a fat-tailed degree distribution**

**(a) Create an undirected network with 1000 nodes, whose degree distribution is proportional to x-3. Plot the degree distribution. What is the diameter?**

<http://en.wikipedia.org/wiki/Barab%C3%A1si%E2%80%93Albert_model>

Barabasi-Albert model degree distribution is k-3

|  |
| --- |
| Degree Distribution |
| g1 <- barabasi.game(1000,directed=F)  plot(degree.distribution(g1))  lines(degree.distribution(g1)) |
|  |

diameter(g1)

Result = 17

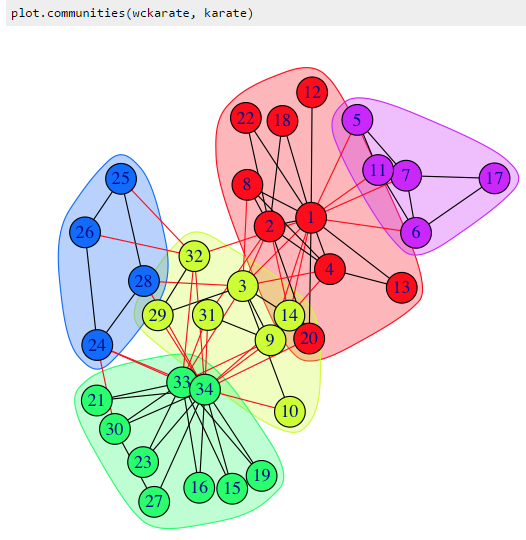
**(b) Is the network connected? Find the giant connected component (GCC) and use fast greedy method to find the community structure. Measure the modularity. Why is the modularity so large?**

|  |
| --- |
|  |
| cl\_g1 <- clusters(g1)  cl\_g1$csize  gccIndex\_g1 <- which.max(cl\_g1$csize)  nonGccNodes\_g1 <- (1:vcount(g1))[cl\_g1$membership != gccIndex\_g1]  gcc\_g1 <- delete.vertices(g1,nonGccNodes\_g1) |
|  |

|  |
| --- |
|  |
| # if we wanted to use the fastgreedy.community agorithm we would do  fc <- fastgreedy.community(g)  com<-community.to.membership(g, fc$merges, steps= which.max(fc$modularity)-1)  V(g)$color <- com$membership+1  g$layout <- layout.fruchterman.reingold  plot(g, vertex.label=NA) |
|  |

modularity(g1,membership(fastgreedy.community(g1)))

[1] 0.9176789



**(c) Try to generate a larger network with 10000 nodes whose degree distribution is proportional to x-3. Compute the modularity. Is the same as the smaller network’s?**

|  |
| --- |
|  |
| g2 <- barabasi.game(10000,directed=F)  modularity(g2,membership(fastgreedy.community(g2))) |
|  |

Modularity : 0.9736591

**(d) You can randomly pick a node I, and then randomly pick a neighbor j of that node. Measure and plot the degree distribution of nodes j that are picked with this process.**

**3. Creates a random graph by simulating its evolution**

**(a) Each time a new vertex is added it creates a number of links to old vertices and the probability that an old vertex is cited depends on its in-degree (preferential attachment) and age. Produce such an undirected network with 1000 nodes. Plot the degree distribution.**

|  |  |  |
| --- | --- | --- |
|  | | |
| g1 <- aging.prefatt.game(1000, pa.exp=1, aging.exp=0, aging.bin=1000,directed=F)  g2 <- aging.prefatt.game(1000, pa.exp=1, aging.exp=-1, aging.bin=1000,directed=F)  g3 <- aging.prefatt.game(1000, pa.exp=1, aging.exp=-3, aging.bin=1000,directed=F) | | |
|  |  |  |

**(b) Use fast greedy method to find the community structure. What is the modularity?**

> modularity(g1,membership(fastgreedy.community(g1)))

[1] 0.908188

> modularity(g2,membership(fastgreedy.community(g2)))

[1] 0.9358132

> modularity(g3,membership(fastgreedy.community(g3)))

[1] 0.9364505

**4. Use the forest fire model to create a directed network**

**(a) This is a growing network model, which resembles how the forest fire spreads by igniting trees close by. Plot the in and out degree distributions.**

|  |  |  |
| --- | --- | --- |
|  | | |
| g\_fire <- forest.fire.game(10000, fw.prob=0.37, bw.factor=0.32/0.37)  plot(degree.distribution(g\_fire,mode="in"))  plot(degree.distribution(g\_fire,mode="out"))  plot(degree.distribution(g\_fire,mode="all")) | | |
|  |  |  |

**(b) Measure the diameter.**

diameter(g\_fire)

[1] 15

**(c) Measure the community structure and modularity.**

fc\_g\_fire <- walktrap.community(g\_fire)

fast greedy applied only undirected graph

modularity(g\_fire,membership(walktrap.community(g\_fire)))

[1] 0.1809519