1. Use the Following Data set of papers to construct a network of authors The data was cleaned using Notepad++, and uploaded to R from there. Jace King introduced me to Notepad++ and refereed me to a few RegEx commands.

   (a) Create the Graph and report the number of edges (with self edges reported separately).

      i. Code:

```
library(igraph)
A = scan("collab_data1", what = 'list') #Upload (cleaned in Notebook++) the comma
separated Authors, each line is an element in a list

#initialize
B = numeric(0);  B2 = numeric(0); C = numeric(0)
C1 = rbind(0,0); S = numeric(0);  S1 = numeric(0)

for(i in 1:(length(A))) B[i] = strsplit(A[i], ',') #Split by comma
for(i in 1:length(B)){
  if (length(B[[i]]) >= 2){
    C = combn(B[[i]],2) #Get all combinations
  } else {
    S = B[[i]] #Keep the single authors for nodes purposes
  }
  C1 = cbind(C, C1)  #add these onto the previous row of combinations
  S1 = c(S, S1)
}
ConnectionsM = t(C1[,1:(ncol(C1)-1)])
ConnectionsD = as.data.frame(ConnectionsM)

NodesM = unique(c(as.vector(ConnectionsM), S1))
NodesD = as.data.frame(NodesM)

g = graph_from_data_frame(d = ConnectionsD, vertices = NodesD, directed = FALSE)
ActorNetwork = simplify(g, remove.loops = FALSE)
WithOutLoops = simplify(ActorNetwork, remove.loops = TRUE)

NodeCount = length(NodesM) #34570
TotCount = ecount(ActorNetwork) #110633
DisinctCount = ecount(WithOutLoops)
SelfLoopCount = TotCount - DisinctCount #389
```

      ii. Output:

```
> TotCount
[1] 110633 (total number of edges)

> SelfLoopCount
[1] 389

> DisinctCount
[1] 110244
```

   (b) Consider the Degree Distribution of Nodes in the Graph where n_j is the number of Nodes with degree j and d is the maximum of n_j

i. For each j from 0 to d, output n_j

   A. Code:

   ```
   nj = degree_distribution(ActorNetwork)*NodeCount
   ```

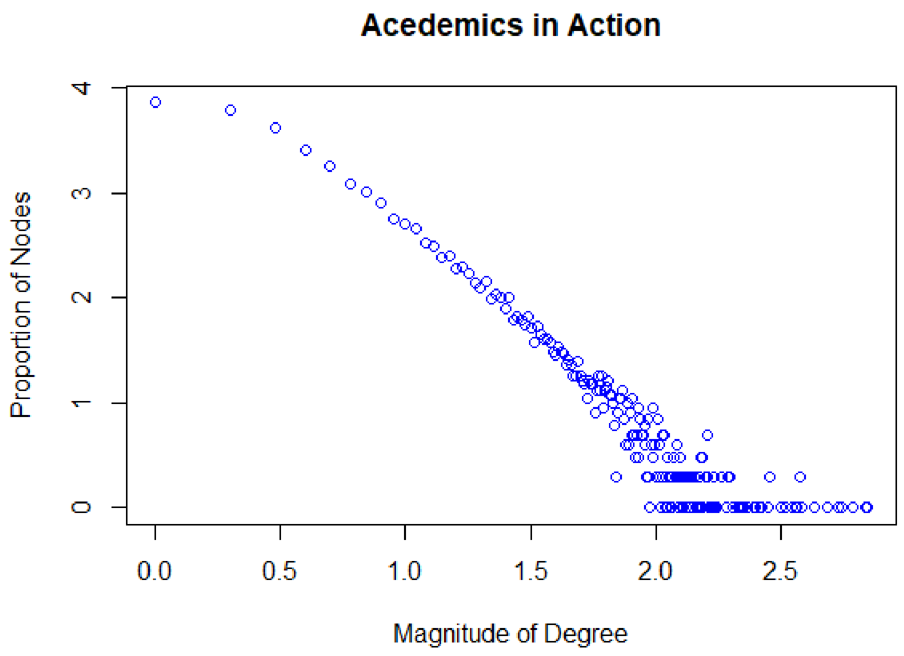
   B. Output: Found under variable nj.

ii. Produce a Scatterplot of $(\log j, \log n_j)$

   A. Code:

   ```
   y1 = nj
   x1 = 0:(length(nj)-1)
   plot(log10(x1), log10(y1), col="blue", main = 'Acedemics in Action',
   xlab = 'Magnitude of Degree', ylab = 'Proportion of Nodes' ) #Degree
   ```

   B. Output:



(c) Connected Components

i. Report the number of nodes in the largest connected component, the number of nodes in the graph overall, and their ratio.

A. Code:

```
LarConCom = length(component_distribution(ActorNetwork))
ComponentProportion = LarConCom/NodeCount
```

B. Output:

```
> LarConCom
[1] 27643
> ComponentProportion
[1] 0.799624
```

ii. Let k_j denote the number of connected components of size j. Report j and k_j for each j such that $k_j > 0$.

A. Code:

```
kj = component_distribution(ActorNetwork)[-1]*count_components(ActorNetwork)
```

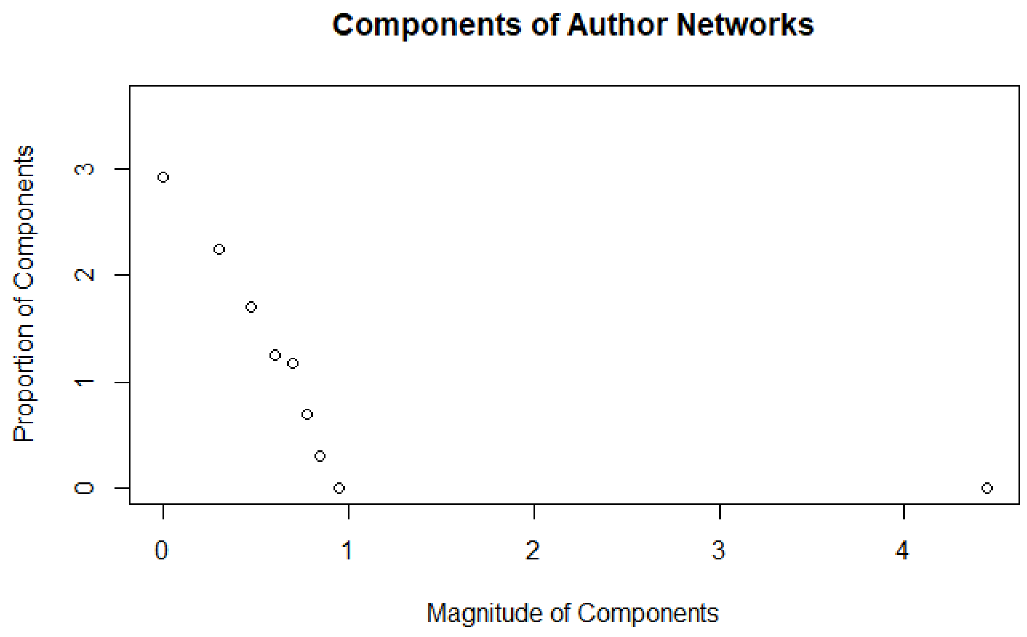
B. Output: Found under Variable kj.

iii. Produce a scatterplot of $(\log j, \log k_j)$

A. Code:

```
plot(log10(0:(length(kj)-1)), log10(kj), xlab = 'Magnitude of
Components', ylab = 'Proportion of Components', main = 'Components
of Author Networks')
```

B. Output:



(d) Node to Node Distances.

   i. Report the maximum distance between Hartmanis and any other author in the largest connected component .

      A. Code:

```
d1 = distances(ActorNetwork, v = "Hartmanis")
MaxDisHart = max(d1[d1 != Inf])
```

      B. Output:

```
> MaxDisHart
[1] 9
```

   ii. Report the maximum distance between Hartmanis and any other author in the largest connected component.

      A. Code:

```
d = distances(ActorNetwork, v = "Bornberg-Bauer")
MaxDisBorn = max(d[d != Inf])
```

      B. Output:

```
> MaxDisBorn
[1] 12
```

   iii. Give an upper and lower bound on the diameter of the largest connected component:
   The largest connected component will have a diameter of between 12-18. The lower bound is 12 since Harthamis is distance 12 from their furthest node. The max is 18 since any node can travel to Bornberg-Bauer (at most distance 9) and another at most distance 9 to reach their destination.

(e) Analyze an additional property of the network:

   i. Code:

```
ClustCoeff = transitivity(ActorNetwork, type = 'average')
```

   ii. Output:

```
> ClustCoeff
[1] 0.5919084
```

   iii. Interpretation: The author network has an average clustering coefficient of 0.5919084, meaning a little over half of the average authors coauthors have authored together (try saying that 10 times fast). Their are a few factors leading into a coefficient that is so high. For one, authors in similar fields will often expand upon each others work. Also, academics interact with the same group of people through conferences and other academic affairs, thus creating a clique of coauthorship.

2. Analyze the Properties of a Random and Small World Network and compare them to the author network

   (a) Creating the Graphs:

      i. Rand Network:

```
Rand = function(n,m){
  if(m > ((n^2+n)/2)){stop( "You can only have up to (n^2+n)/2 unique edges") }
  Nodes = as.data.frame(1:n)
  rand = matrix(sample(1:n, 2*m, replace = TRUE), m, 2)
  d = unique(t(apply(rand,1,sort)))
  while (nrow(d) < (m+1)){
    c = sort(cbind(sample(1:n, 1),sample(1:n, 1)))
    c = matrix(c, 1, 2)
    if(nrow(unique(rbind(d, c))) == nrow(rbind(d, c))){
      d = rbind(d, c)
    }
  }
  d = d[2:nrow(d),]
  Edges = as.data.frame(d)
  g = graph_from_data_frame(d = Edges, vertices = Nodes, directed = FALSE)
}
```

      ii. Interpretation: This had only 1 self loop when constructed with the same number of nodes and edges as the author network. This is expected as there are much more as the number of total possible edges, which is $\frac{(n+1)*n}{2}$ is much larger than the number of edges, so a their is a very small probability a given connection will happen.

      iii. Small World:

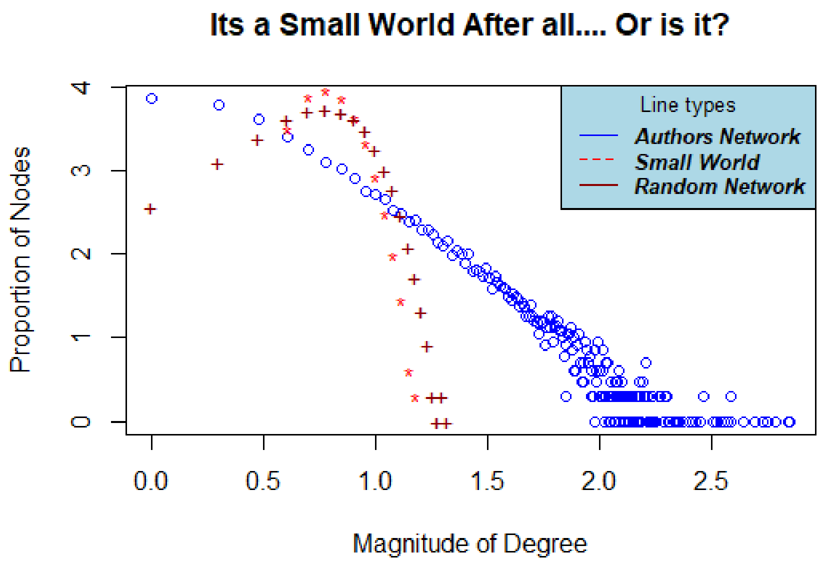
```
  Smol = function(n,m){
  if(m<2*n){stop( "m must be larger than 2*n") }
  nd = as.data.frame((1:n))
  circ = cbind(1:n, c(2:n, 1))
  web = cbind(1:n, c(3:n, 1, 2))
  rand = matrix(sample(1:n, 2*(m-2*n), replace = TRUE), m-2*n, 2)
  d = unique(t(apply(rbind(circ, web, rand),1,sort)))
  while (nrow(d) < m){
    c = sort(cbind(sample((1:n), 1), sample((1:n), 1)))
    c = matrix(c , 1, 2)
    if (nrow(unique(rbind(d, c))) == nrow(rbind(d, c))){d = rbind(d, c)}
  }
  e = as.data.frame(d)
  g = graph_from_data_frame(d = e, vertices = nd, directed = FALSE)
}
```

(b) Plot the log-log degree distribution of the three aforementioned networks.

i. Code:

```
y1 = nj
x1 = 0:(length(nj)-1)

y2 = degree_distribution(S)*NodeCount
x2 = 0:(length(y2)-1)

y3 = degree_distribution(R)*NodeCount
x3 = 0:(length(y3)-1)

plot(log10(x1), log10(y1), col="blue", main = 'Its a Small World
After all.... Or is it?', ylab = 'Proportion of Nodes' , xlab =
'Magnitude of Degree') #Degree Distribution of Author Network

points(log10(x2), log10(y2), col="red", pch="*")

points(log10(x3), log10(y3), col="dark red",pch="+")

legend(x = "topright", legend=c("Authors Network", "Small World",
"Random Network"),col=c("blue", "red", "dark Red"), lty=1:2,
cex=0.8, title="Line types", text.font=4, bg='lightblue')
```

ii. Output:



iii. Interpretation: The Author network shows that the larger the degree, the less amount of nodes will take that degree. This relationship is logarithmic linear (which is exponential decay relationship) but with more spread for larger degree. Both the Small world and Random networks show a parabolic relationship with most nodes taking on around log(degree) = 1 (meaning degree 10). Both of these networks do not have nodes with as large of a degree.

This is most likely because acedemic papers will often have a large number of contributers as many hands make lighter work. This is not capture in the randomness of the other two graphs. Also, the small world graph forces each node to have at least four connections, so their aren't low degree nodes like in the real world example.

(c) Report the size of the largest connected component of the Random Network and Small World Network.

i. Code:

```
LarConComRand = length(component_distribution(Rand(NodeCount, DistictLoopCount)))
LarConComSmol = length(component_distribution(Smol(NodeCount, DistictLoopCount)))
```

ii. Output:

```
> LarConComRand
[1] 34513
> LarConComSmol
[1] 34571
```

(d) Node to Node Distances

i. Code:

```
d2 = distances(S, v = "6")
MaxDisS = max(d2[d2 != Inf])

d3 = distances(R, v = "6")
MaxDisR = max(d3[d3 != Inf])
```

ii. Output:

```
> MaxDisS
[1] 9
> MaxDisR
[1] 9
```

(e) My own analysis.

i. Code:

```
ClustCoeffS = transitivity(S, type = 'average')
ClustCoeffR = transitivity(R, type = 'average')
```

ii. Output:

```
> ClustCoeffS
[1] 0.2138458
> ClustCoeffR
[1] 0.000165493
```

iii. Interpretation: We see that the clustering coefficient of the random and small world graphs are much lower than the author network. For the small world graph, this coefficient is small, in part, for the same reason that their are so few self loops. The neighbor of a given node has a much larger proportion of connections to choose from, namely $(n+1)*n/2$, than the average number of neighbors available (nameley about 6). The small world graph, however, forces each neighbor to be neighbors with at least one other neighbor. Also, since their are around 35,000 nodes to 100,000 connections, 70,000 are already made and about 30,000 remain. The likely hood that these connections would be between neighbors of a given node is unlikely. It is slightly more likely, however, that the node would receive another connection from the 30,000 remaining.