INFX 576: Problem Set 6 - Groups, Communities and Graph Sets*

Harkar Talwar

Due: Friday, May 11, 2018

Collaborators: Prateek Tripathi, Aakash Agrawal

Instructions:

Before beginning this assignment, please ensure you have access to R and RStudio.

1. Download the `problemset6.Rmd` file from Canvas. You will also need the data from last week's Problem Set 4 in `problemset6_data.Rdata`.
2. Replace the "Insert Your Name Here" text in the `author:` field with your own full name. Any collaborators must be listed on the top of your assignment.
3. Be sure to include well-documented (e.g. commented) code chucks, figures and clearly written text chunk explanations as necessary. Any figures should be clearly labeled and appropriately referenced within the text.
4. Collaboration on problem sets is acceptable, and even encouraged, but each student must turn in an individual write-up in his or her own words and his or her own work. The names of all collaborators must be listed on each assignment. Do not copy-and-paste from other students' responses or code.
5. When you have completed the assignment and have **checked** that your code both runs in the Console and knits correctly when you click `Knit PDF`, rename the R Markdown file to `YourLastName_YourFirstName_ps6.Rmd`, knit a PDF and submit the PDF file on Canvas.

Setup:

In this problem set you will need, at minimum, the following R packages.

```
# Load standard libraries
library(statnet)
# Load the required data
load("problemset6_data.Rdata")
```

Problem 1: Cohesive Subgroups

In this problem we use data collected by Krackhardt (1987), `kfr` capturing self-reported friendship ties among 21 managers in a high-tech firm. This data is directed and unvalued, it is possible for i to nominate j as a friend without reciprocation.

(a) Cliques

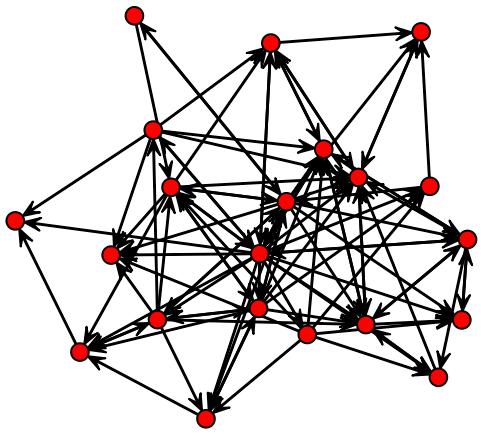
Using the `clique.census` command, perform the following analyses on `kfr`:

- Obtain a length-tabulation of clique membership by vertex.
- Obtain the combined clique co-membership matrix.

*Problems originally written by C.T. Butts (2009)

- Use the clique co-membership matrix to obtain a cohesion-based blockmodel of `kfr`. You may find the commands `hclust` `cutree` and `blockmodel` helpful here. Show the dendrogram (with cutoff value), block image matrix, and block image.

```
# Plot the network
gplot(kfr, main = "Krackhardt: Manager Friendship Network (kfr)")
```



Length-tabulation of clique membership by vertex

```
census = clique.census(kfr)  # Perform a clique census
census$clique.count          # Print clique membership by vertex
```

```
##   Agg v1 v2 v3 v4 v5 v6 v7 v8 v9 v10 v11 v12 v13 v14 v15 v16 v17 v18 v19
## 1   4  0  0  0  0  0  0  1  0  1   1   0   0   0   0   0   0   0   0   0
## 2   9  2  3  1  1  0  1  0  1  0   0   1   0   1   1   1   1   1   1   1
## 3   6  1  0  0  2  2  0  0  0  0   0   3   3   0   0   1   0   3   0   2
##   v20 v21
## 1   1   0
## 2   0   1
## 3   0   1
```

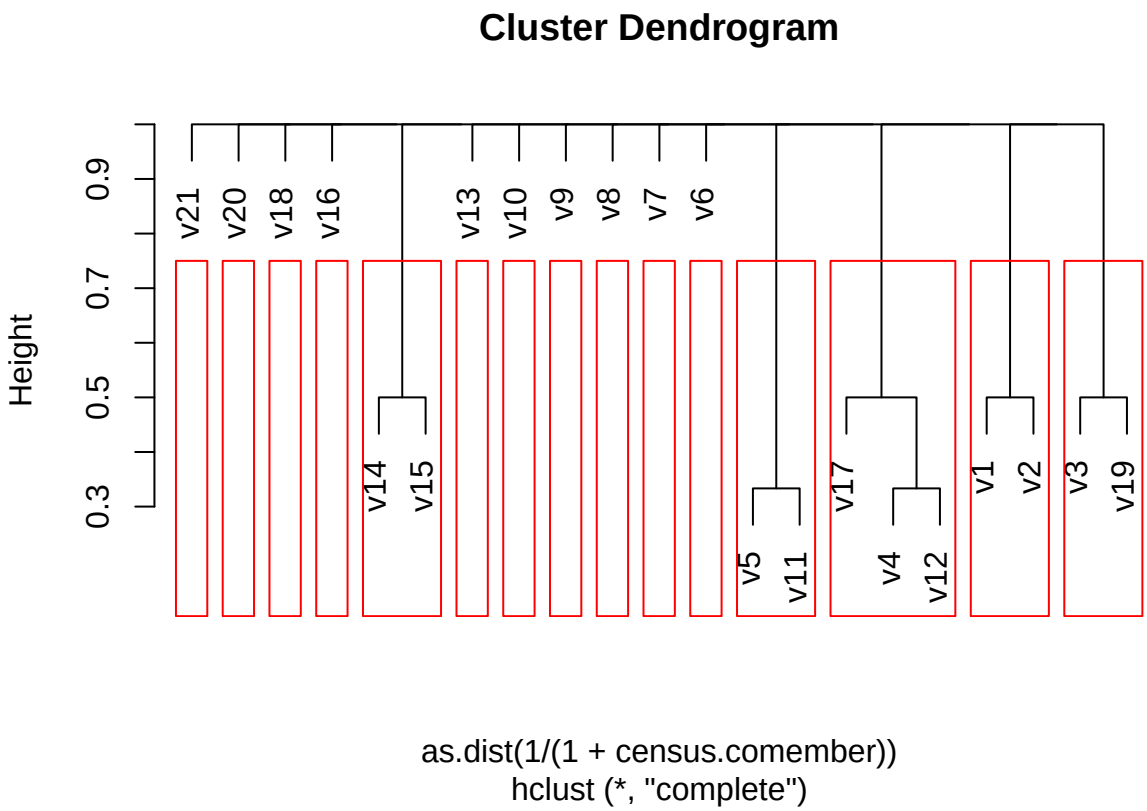
Combined clique co-membership matrix

```
# Obtain a clique comembership matrix
census.comember = clique.census(kfr,
                                clique.comembership = "sum")$clique.comemb
census.comember
```

```
##     v1 v2 v3 v4 v5 v6 v7 v8 v9 v10 v11 v12 v13 v14 v15 v16 v17 v18 v19 v20
## v1   3  1  0  1  0  0  0  0  0   0   0   1   0   0   0   1   0   0   0   0
## v2   1  3  0  0  0  0  0  0  0   0   0   0   0   0   0   0   0   1   0   0
## v3   0  0  1  0  0  0  0  0  0   0   0   0   0   0   0   0   0   0   1   0
## v4   1  0  0  3  0  0  0  1  0   0   0   2   0   0   0   0   1   0   0   0
## v5   0  0  0  0  2  0  0  0  0   0   2   0   0   0   0   0   1   0   1   0
## v6   0  0  0  0  0  1  0  0  0   0   0   0   0   0   0   0   1   0   0   0
## v7   0  0  0  0  0  0  1  0  0   0   0   0   0   0   0   0   0   0   0   0
## v8   0  0  0  1  0  0  0  1  0   0   0   0   0   0   0   0   0   0   0   0
## v9   0  0  0  0  0  0  0  0  1   0   0   0   0   0   0   0   0   0   0   0
## v10  0  0  0  0  0  0  0  0  0   1   0   0   0   0   0   0   0   0   0   0
## v11  0  0  0  0  2  0  0  0  0   0   4   0   1   0   1   0   1   0   2   0
## v12  1  0  0  2  0  0  0  0  0   0   0   3   0   0   0   0   2   0   0   0
## v13  0  0  0  0  0  0  0  0  0   0   1   0   1   0   0   0   0   0   0   0
## v14  0  0  0  0  0  0  0  0  0   0   0   0   0   1   1   0   0   0   0   0
## v15  0  0  0  0  0  0  0  0  0   0   1   0   0   1   2   0   0   0   1   0
## v16  1  0  0  0  0  0  0  0  0   0   0   0   0   0   0   1   0   0   0   0
## v17  0  0  0  1  1  1  0  0  0   0   1   2   0   0   0   0   4   0   0   0
## v18  0  1  0  0  0  0  0  0  0   0   0   0   0   0   0   0   0   1   0   0
## v19  0  0  1  0  1  0  0  0  0   0   2   0   0   0   1   0   0   0   3   0
## v20  0  0  0  0  0  0  0  0  0   0   0   0   0   0   0   0   0   0   0   1
## v21  0  1  0  0  0  0  0  0  0   0   0   1   0   0   0   0   1   0   0   0
##     v21
## v1    0
## v2    1
## v3    0
## v4    0
## v5    0
## v6    0
## v7    0
## v8    0
## v9    0
## v10   0
## v11   0
## v12   1
## v13   0
## v14   0
## v15   0
## v16   0
## v17   1
## v18   0
## v19   0
## v20   0
## v21   2
```

Clustering based on clique comembership

```
 # Perform clustering by comembership
hc = hclust(as.dist(1/(1+census.comember)))
plot(hc)                # Create a dendrogram
rect.hclust(hc, h=0.7)  # Draw a cutoff point
```

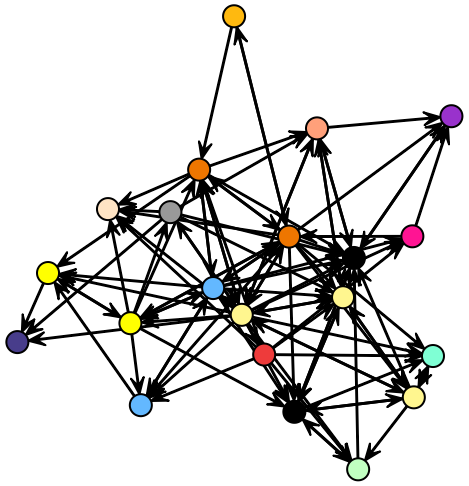



```
ct = cutree(hc, h=0.7)  # Cut the tree into clusters
```

```
# Update the palette to have 15 colors for different clusters
palette(c("black", "steelblue1", "khaki1", "darkorange2",
          "gray60", "darkslateblue", "aquamarine", "bisque",
          "brown2", "darkgoldenrod1", "yellow", "darkseagreen1",
          "darkorchid", "deeppink", "lightsalmon"))
# Plot the network, with vertices colored by cluster
gplot(kfr, vertex.col = ct, vertex.cex = 1.25,
      main = "Krackhardt Friendship Ties (kfr)
      \n (Colored by Comembership Clusters)")
```

Krackhardt Friendship Ties (kfr)

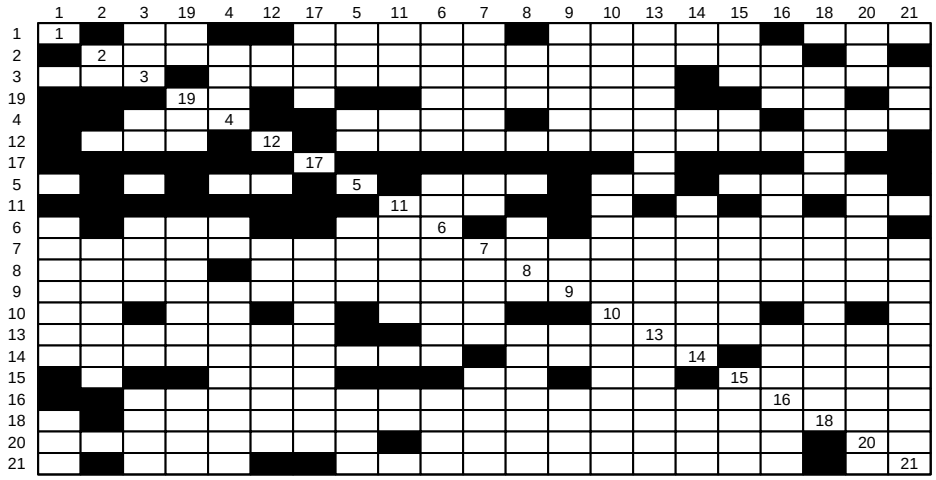
(Colored by Comembership Clusters)



```
palette("default")
```

```
# Create a sociomatrix for the vertices, ordered by comembership cluster
plot.sociomatrix(kfr[order(ct), order(ct)],
                 labels=list(order(ct),order(ct)),
                 main = "Krackhardt(kfr): Block Image Matrix", cex.lab = 0.5)
```

Krackhardt(kfr): Block Image Matrix

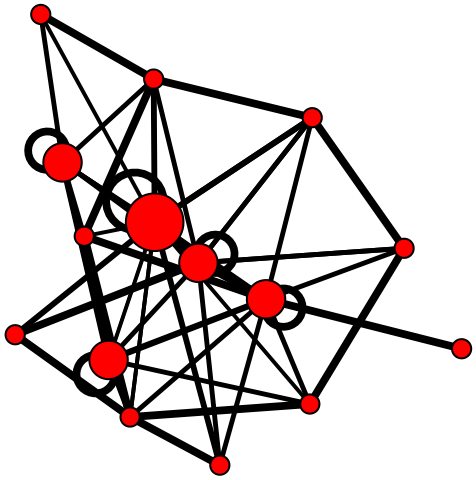



```
# Create a blockmodel based on the comembership clustering
bm<-blockmodel(kfr, ct)
bm
```

```
##
## Network Blockmodel:
##
## Block membership:
##
##  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21
##  1  1  2  3  4  5  6  7  8  9  4  3 10 11 11 12  3 13  2 14 15
##
## Reduced form blockmodel:
##
##   1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21
##            Block 1   Block 2   Block 3   Block 4   Block 5   Block 6
## Block 1  1.0000000 0.0000000 0.3333333 0.0000000 0.0000000 0.0000000
## Block 2  0.5000000 1.0000000 0.1666667 0.5000000 0.0000000 0.0000000
## Block 3  0.8333333 0.3333333 1.0000000 0.3333333 0.3333333 0.3333333
## Block 4  0.7500000 0.7500000 0.6666667 1.0000000 0.0000000 0.0000000
## Block 5  0.5000000 0.0000000 0.6666667 0.0000000       NaN 1.0000000
## Block 6  0.0000000 0.0000000 0.0000000 0.0000000 0.0000000       NaN
## Block 7  0.0000000 0.0000000 0.3333333 0.0000000 0.0000000 0.0000000
## Block 8  0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000
## Block 9  0.0000000 0.5000000 0.3333333 0.5000000 0.0000000 0.0000000
## Block 10 0.0000000 0.0000000 0.0000000 1.0000000 0.0000000 0.0000000
## Block 11 0.2500000 0.5000000 0.0000000 0.5000000 0.5000000 0.5000000
```

```
## Block 12 1.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000
## Block 13 0.5000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000
## Block 14 0.0000000 0.0000000 0.0000000 0.5000000 0.0000000 0.0000000
## Block 15 0.5000000 0.0000000 0.6666667 0.0000000 0.0000000 0.0000000
##            Block 7   Block 8   Block 9 Block 10  Block 11  Block 12
## Block 1  0.5000000 0.0000000 0.0000000      0.0 0.0000000 0.5000000
## Block 2  0.0000000 0.0000000 0.0000000      0.0 0.7500000 0.0000000
## Block 3  0.6666667 0.3333333 0.3333333      0.0 0.3333333 0.6666667
## Block 4  0.5000000 1.0000000 0.0000000      0.5 0.5000000 0.0000000
## Block 5  0.0000000 1.0000000 0.0000000      0.0 0.0000000 0.0000000
## Block 6  0.0000000 0.0000000 0.0000000      0.0 0.0000000 0.0000000
## Block 7        NaN 0.0000000 0.0000000      0.0 0.0000000 0.0000000
## Block 8  0.0000000       NaN 0.0000000      0.0 0.0000000 0.0000000
## Block 9  1.0000000 1.0000000       NaN      0.0 0.0000000 1.0000000
## Block 10 0.0000000 0.0000000 0.0000000      NaN 0.0000000 0.0000000
## Block 11 0.0000000 0.5000000 0.0000000      0.0 1.0000000 0.0000000
## Block 12 0.0000000 0.0000000 0.0000000      0.0 0.0000000       NaN
## Block 13 0.0000000 0.0000000 0.0000000      0.0 0.0000000 0.0000000
## Block 14 0.0000000 0.0000000 0.0000000      0.0 0.0000000 0.0000000
## Block 15 0.0000000 0.0000000 0.0000000      0.0 0.0000000 0.0000000
##          Block 13  Block 14  Block 15
## Block 1       0.5 0.0000000 0.5000000
## Block 2       0.0 0.5000000 0.0000000
## Block 3       0.0 0.3333333 0.6666667
## Block 4       0.5 0.0000000 0.5000000
## Block 5       0.0 0.0000000 1.0000000
## Block 6       0.0 0.0000000 0.0000000
## Block 7       0.0 0.0000000 0.0000000
## Block 8       0.0 0.0000000 0.0000000
## Block 9       0.0 1.0000000 0.0000000
## Block 10      0.0 0.0000000 0.0000000
## Block 11      0.0 0.0000000 0.0000000
## Block 12      0.0 0.0000000 0.0000000
## Block 13      NaN 0.0000000 0.0000000
## Block 14      1.0       NaN 0.0000000
## Block 15      1.0 0.0000000       NaN
```

```r
# Plot the blockmodel of clusters, with node size indicating the size of the cluster
# And edge width indicating the density
gplot(bm$block.model, vertex.cex = table(ct),
      edge.lwd = 6*bm$block.model, usearrows=FALSE,
      diag=TRUE, main = "kfr Blockmodel Based on Clique Comembership")
```



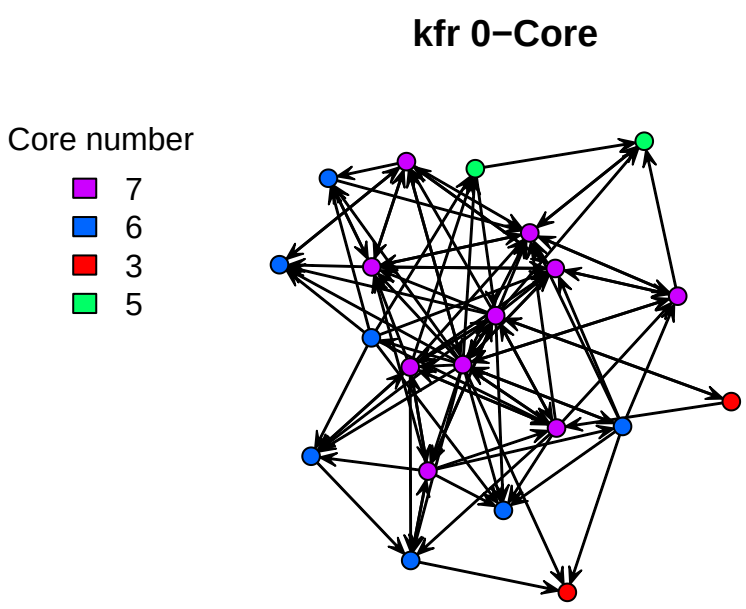
(b) K-Cores

Use the `kcores` command to calculate the total degree k-cores of `kfr`. Visualize the network, indicating by size, shape, or color the core number for each vertex.

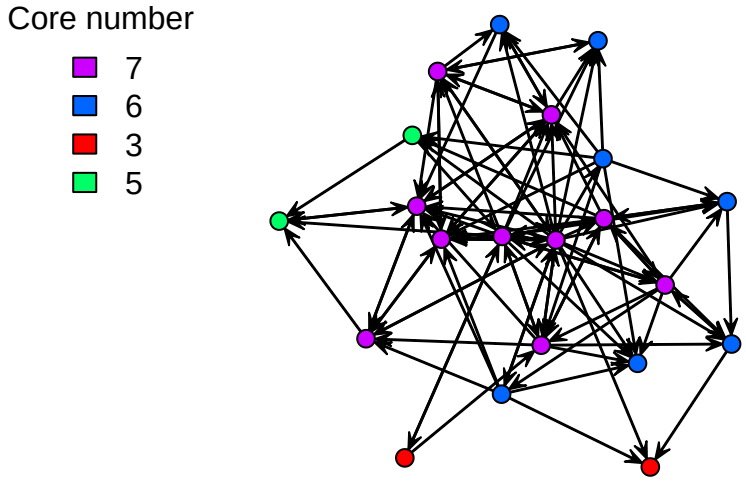
```
# Compute k-cores
kc = kcores(kfr)

# Function to plot nodes where cores >= core.number
plot.kcores = function(core.number) {
  index = core.number # index to filter nodes
  gplot(kfr[kc>=index,kc>=index],
        vertex.col= rainbow(max(kc[kc>=index])-2)[kc[kc>=index]-2],
        main = paste("kfr ", core.number, "-Core", sep = ""))
  legend.vals = kc[kc>=index]
  legend("topleft",
         fill=unique(rainbow(max(kc[kc>=index])-2)[kc[kc>=index]-2]),
         legend=unique(legend.vals), bty="n", title = "Core number")
}
```

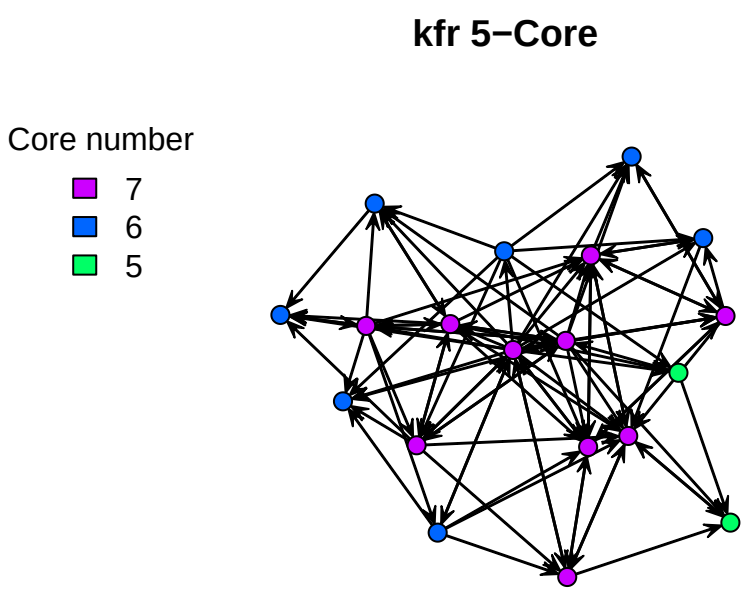
```
plot.kcores(0)
```



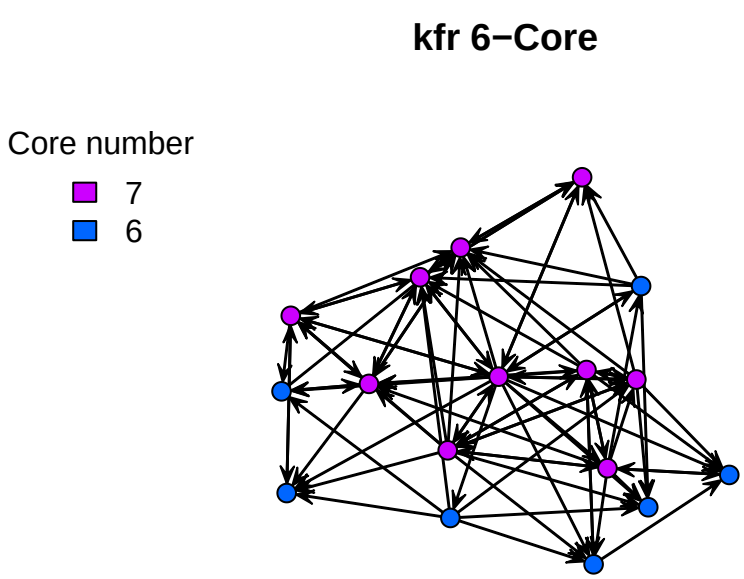

```
plot.kcores(3)
```



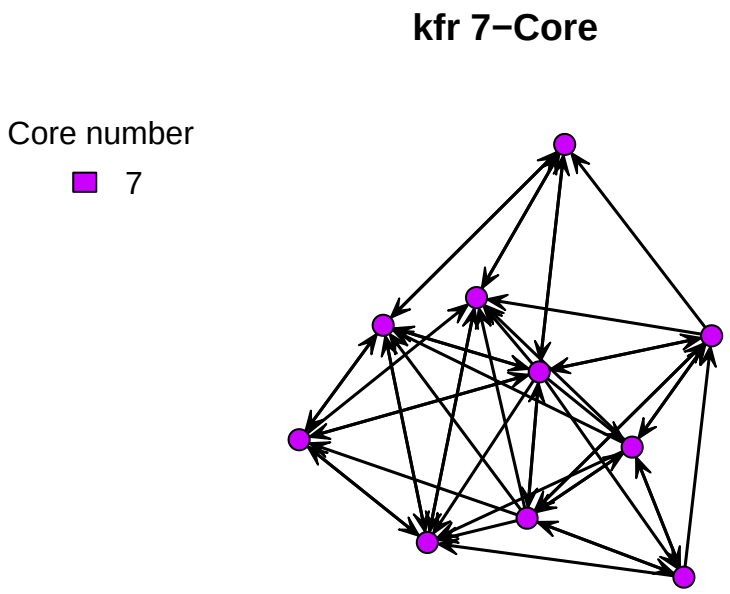

```
plot.kcores(5)
```




```
plot.kcores(6)
```




```
plot.kcores(7)
```



(c) Discussion

Based on the anlaysis in parts (a) and (b), how would you summarize the structure of this network; in particular, how many distinct dense clusters do there appear to be?

Analysis based on Clique Comembership

- The analysis in part a) showed that there are 5 distinct dense clusters (out of the 15 identified) based on clique comembership in the kfr network. These are:
  1. [v5, v11]
  2. [v4, v12, v17]
  3. [v14, v15]
  4. [v1, v2]
  5. [v3, v19]

  The clusters are shown in the Dendrogram above, with 0.7 as the cut-off.
- Further, the sociomatrix of the vertices (ordered by clique comembership) shows that managers that are part of the same cluster have incoming and outgoing friendship ties with a lot of common managers, thus clique comembership seems to be a good indicator of cohesive subgroups in the kfr network.
- The reduced form blockmodel matrix shows that density of within block ties is 1 for all of the 5 clusters of managers, which we would expect based on clique comembership. Further, higher inter-block friendships are also observed for ties between these 5 clusters. For instance, the density of friendship

ties from Block 3 to Block 1 is 0.83.

Analysis based on k-cores

- Our analysis in part b) above shows that there are 4 clusters based on core numbers in the kfr network. These are:
  1. 3-shell, 2. 5-shell, 3. 6-shell, 4. 7-shell
- The 7-shell has the highest density with 10 managers belonging to it, followed by the 6 shell (7 vertices), and the 5-shell, 3-shell with 2 managers each.
- Thus we see that each of the managers has friendship ties to at-least 3 other managers, and almost half of them have 7 ties.

Problem 2: Graph Correlation

Last week, we saw network data from the famous Bernard, Killworth, and Sailer (BKS) studies. These studies examined the issue of accuracy in self-reported network data. Each study involved a group of subjects, each of whom was asked to rank-order all other group members by frequency of interaction. The self-reported interaction frequency was referred to as the "cognitive" ntework by BKS (i.e. the network as understood by the subjects themeselves). During the study period, behavioral information on interaction within the same groups was also collected via trained observers. The network of observed pairwise interaction frequencies was referred to as the "behavioral" network. Accuracy was assessed by comparing the "cognitive" and "behavioral" networks. The BKS studies were controversial and launched a much larger literature on the accuracy of network measurement.

(a) Comparing Networks

For each of the data objects `bkfrat`, `bkham`, `bkoff` and `bktec` (each itself a list containing the cognitive and behavioral network from a BKS study) perform a QAP test of the correlation between the self-report and the observed structure. Show in each case the test results, including a plot of the QAP replicates.

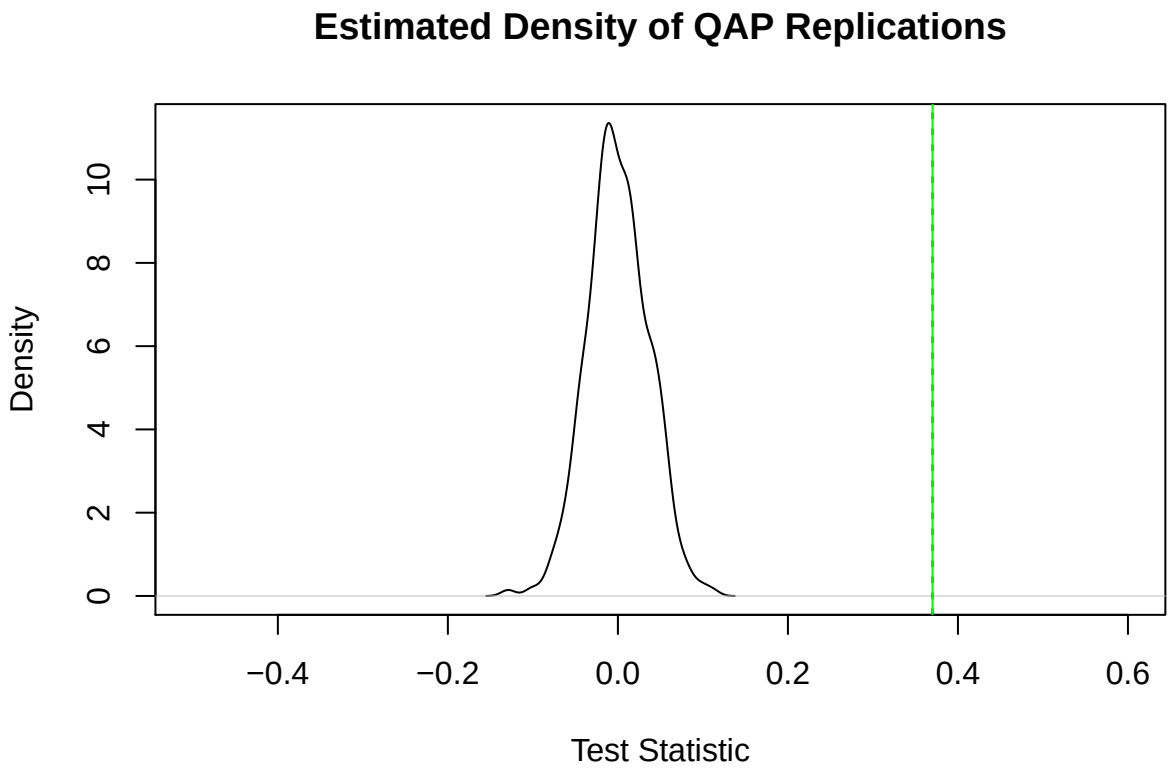
```
# Function to perform QAP test
perform.qap.test = function(network.obj) {
  qt<-qaptest(list(network.obj$Behavioral,network.obj$Cognitive),gcor,g1=1,g2=2)
  print(summary(qt))                          # Examine the results
  plot(qt, xlim = c(-0.5, 0.6))               # Plot the QAP distribution
  abline(v = qt$testval, col="green")         # Plot the observed value
}
```

QAP test for bkfrat

```
perform.qap.test(bkfrat)
```

```
##
## QAP Test Results
##
## Estimated p-values:
##  p(f(perm) >= f(d)): 0
##  p(f(perm) <= f(d)): 1
##
## Test Diagnostics:
##  Test Value (f(d)): 0.3700903
##  Replications: 1000
##  Distribution Summary:
##      Min:     -0.1310971
```

```
##      1stQ:    -0.02304961
##      Med:     -0.001642394
##      Mean:    0.0001592585
##      3rdQ:    0.02377282
##      Max:     0.1137281
```

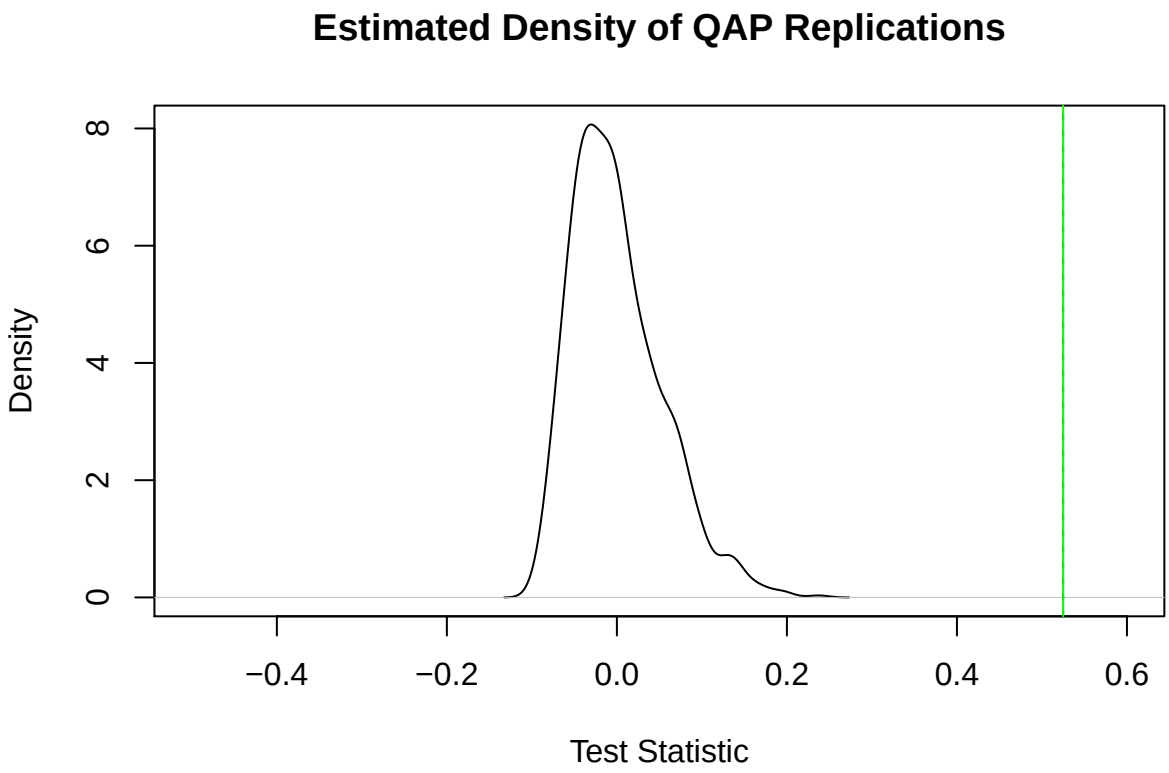


QAP test for bkham

```
perform.qap.test(bkham)
```

```
##
## QAP Test Results
##
## Estimated p-values:
##  p(f(perm) >= f(d)): 0
##  p(f(perm) <= f(d)): 1
##
## Test Diagnostics:
##  Test Value (f(d)): 0.5249309
##  Replications: 1000
##  Distribution Summary:
##      Min:     -0.09655226
##      1stQ:    -0.03941231
##      Med:     -0.007870707
##      Mean:    0.0005466503
##      3rdQ:    0.03152848
```

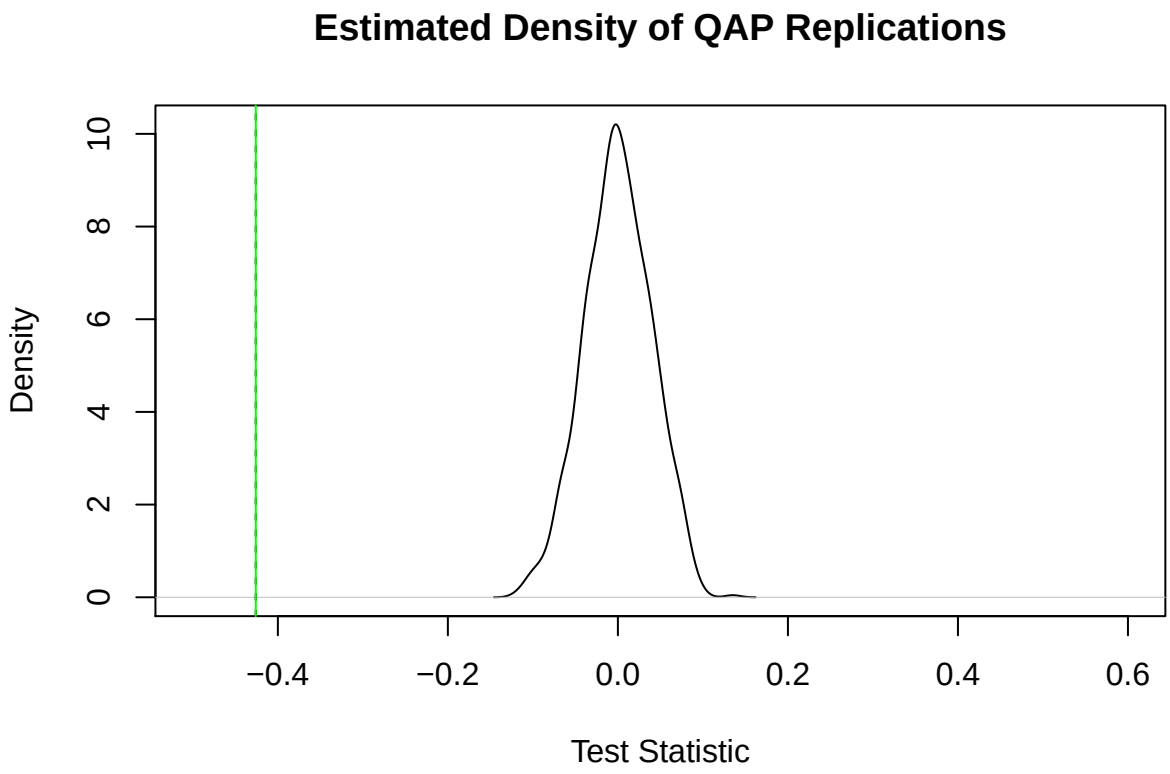
```
##      Max:     0.237161
```



QAP test for bktec

```
perform.qap.test(bktec)
```

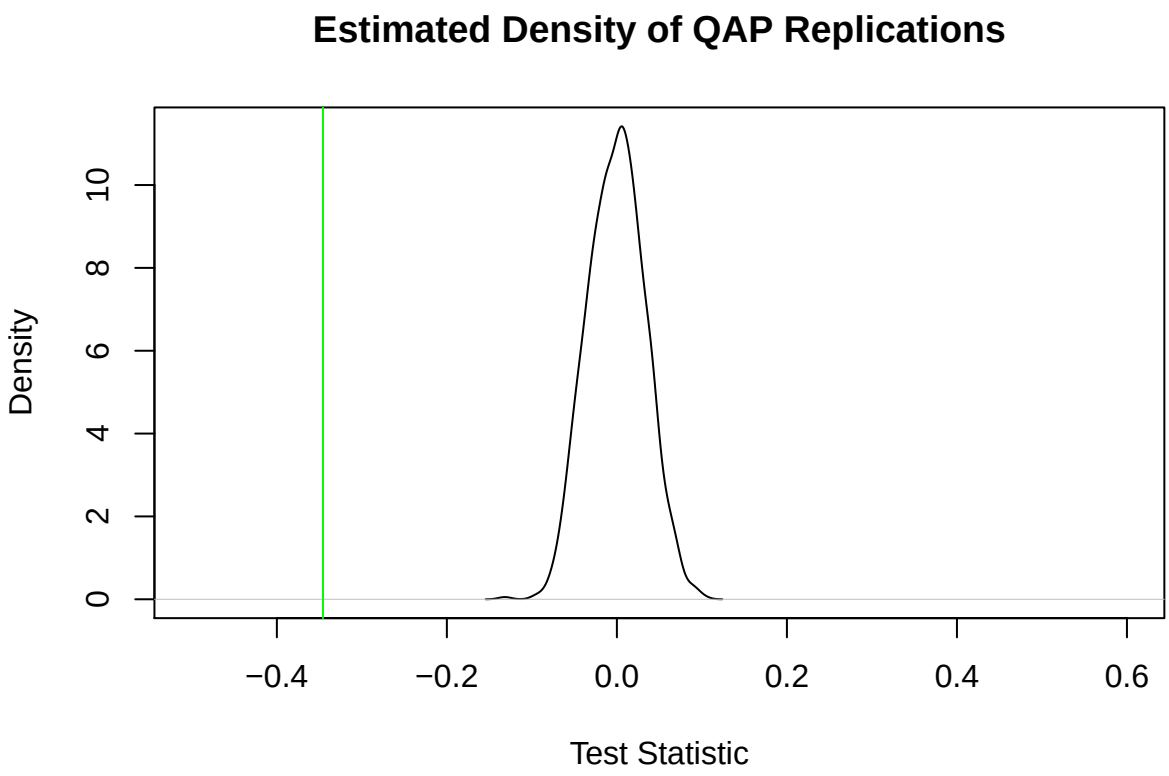
```
##
## QAP Test Results
##
## Estimated p-values:
##  p(f(perm) >= f(d)): 1
##  p(f(perm) <= f(d)): 0
##
## Test Diagnostics:
##  Test Value (f(d)): -0.4260065
##  Replications: 1000
##  Distribution Summary:
##      Min:     -0.1189926
##      1stQ:    -0.02744383
##      Med:     -0.0002071213
##      Mean:    0.0001709796
##      3rdQ:    0.02807572
##      Max:     0.1351669
```



QAP test for bkoff

```
perform.qap.test(bkoff)
```

```
##
## QAP Test Results
##
## Estimated p-values:
##  p(f(perm) >= f(d)): 1
##  p(f(perm) <= f(d)): 0
##
## Test Diagnostics:
##  Test Value (f(d)): -0.3457147
##  Replications: 1000
##  Distribution Summary:
##      Min:     -0.1317618
##      1stQ:    -0.02436421
##      Med:     0.0008941191
##      Mean:    -0.0005016919
##      3rdQ:    0.02291869
##      Max:     0.1013554
```



(b) Discussion

Use the results from part (a) to provide your own assessment of the extent to which the data does or does not show general agreement between observation and informant report.

For these tests, our null hypothesis is:
H_o: The graph correlation between the observed and self reported networks is 0.

- For the network bkfrat and bkham, the observed values of graph correlation are positive (0.37 and 0.52 respectively). The p-values for the QAP tests on these networks are smaller than our significance level of 0.05. This means that the probability of observing such values of correlation, given the null hypothesis is almost 0. Therefore, we reject our null hypothesis. Thus, the self reported interactions among the fraternity members and the radio operators are somewhat similar to the observed interactions among them.
- For the other two networks bktec and bkoff, the correlation values are negative (-0.42 and -0.34 respectively). The p-values are almost 0 and therefore indicate that we can reject our null hypothesis. Thus, for the small business and the research group networks, the observed interactions among members are quite different from the self-reported ones.

(c) Observation and Networks

What reliability or validity issues might arise in the BKS studies if the observed report data is taken to be the true "behavioral" network?

- Taking observed data to be the true behavioral network can lead to validity issues. The observed data should be compared to the self-reported interactions among individuals.

- Differences between the self-reported and observed data can highlight areas, where further analysis might be necessary to determine the accurate network structure. The QAP test above clearly shows how there are differences between the observed and the self-reported interactions for bkoff and bktec networks.
- Lastly, in the BKS studies[1], the observants are said to be unobtrusive, however it is uncertain, to what extent this can be true, in order for them to accurately capture interactions. Any obtrusive observation can influence the actors' behavior and create further validity concerns.

[1] Bernard H, Killworth P and Sailer L. (1982). Informant accuracy in social network data V. Social Science Research, 11, 30-66.

Problem 3: Multivariate Analysis of Network Sets

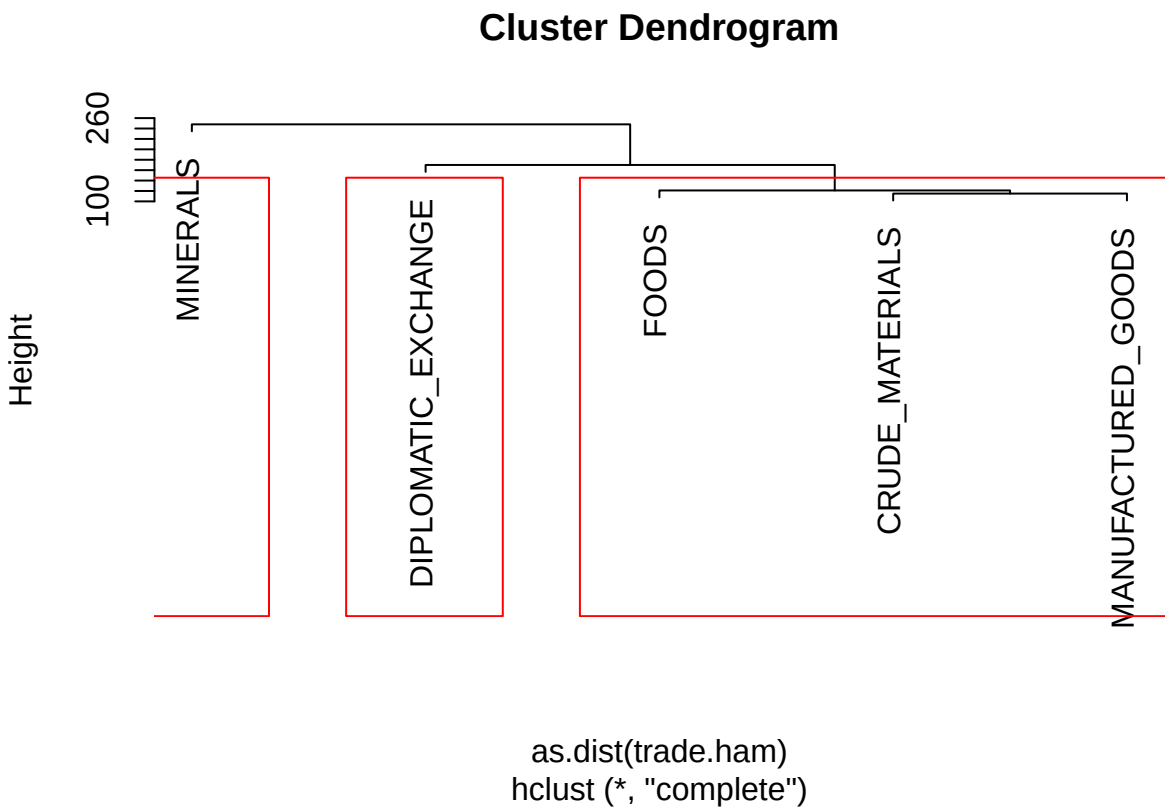
For this problem we will use data on international trade, called `trade` in the data for this problem set. This data captures trade in various types of products/materials among countries. You will want to explore the data before answering these questions, to ensure you understand what is present.

(a) Clustering

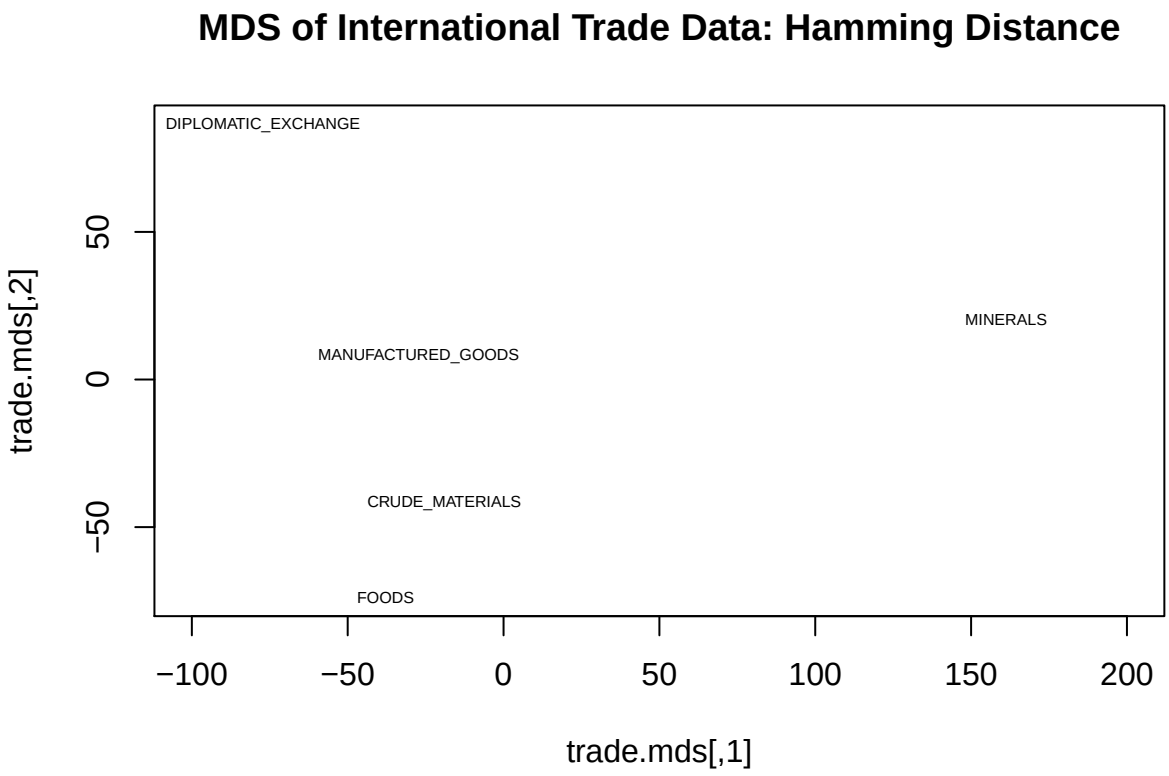
Show a hierarchical clustering of the trade networks, based on the Hamming distance. Compare this with a two-dimensional MDS solution on the same data.

```
# Compute the hamming distance for the trade networks
trade.ham = hdist(trade)
```

```
# Perform clustering based on hamming distance
tradehc<-hclust(as.dist(trade.ham))
# Plot the dendrogram
plot(tradehc,labels=rownames(trade))
# Cut the plot into 3 clusters
rect.hclust(tradehc, k=3)
```




```
# Perform multi-dimensional scaling
trade.mds = cmdscale(trade.ham)
# Plot the results
plot(trade.mds, type = "n", xlim = c(-100, 200),
     main = "MDS of International Trade Data: Hamming Distance")
text(trade.mds, labels = rownames(trade), cex = 0.5)
```



Observations

- The hierarchical clustering of the trade networks shows 3 distinct clusters in the figure above. The two-dimensional MDS solution also gives similar results, where networks with similar adjacency patterns are proximate.

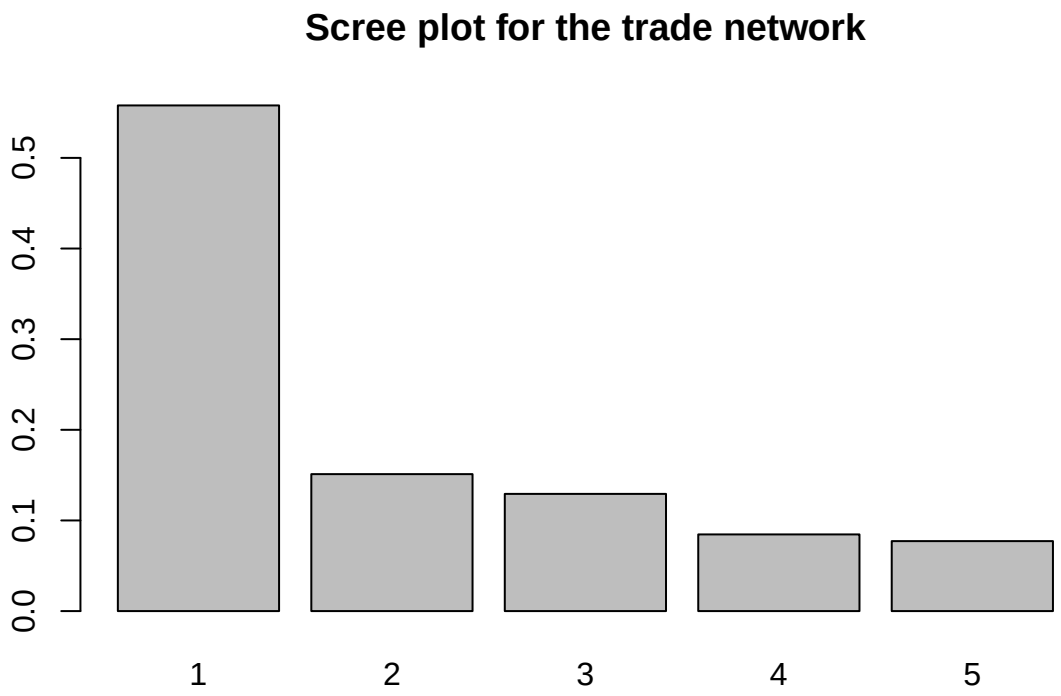
(b) PCA

Conduct a PCA on the trade networks. How many dimensions are needed to account for the bulk of the variation in these networks? Try using a scree plot to help with this question. Plot the loadings on the first two components; what does this suggest about the undering relationships among the trade networks?

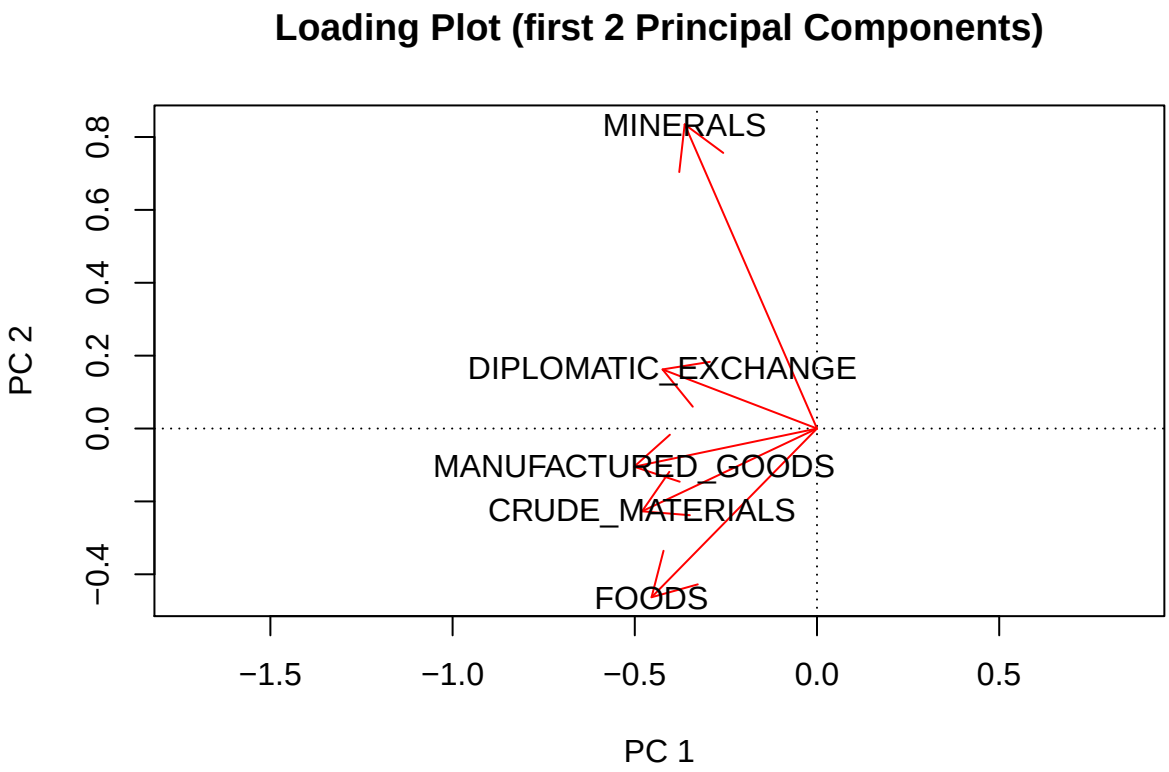
```
trade.cor = gcor(trade)  # compute the graph correlation matrix
trade.cor
```

```
##           1         2         3         4         5
## 1 1.0000000 0.3725626 0.2877321 0.3922966 0.3380220
## 2 0.3725626 1.0000000 0.5670013 0.5775224 0.4165298
## 3 0.2877321 0.5670013 1.0000000 0.5554769 0.3700570
## 4 0.3922966 0.5775224 0.5554769 1.0000000 0.5333617
## 5 0.3380220 0.4165298 0.3700570 0.5333617 1.0000000
```

```
trade.eig = eigen(trade.cor)  # Eigen decomposition of the correlation matrix
trade.evals = trade.eig$values # Extract Eigen values
# Scree plot to explain the % of variance explained by each Principal Component
barplot(trade.evals/sum(trade.evals), names.arg = 1:length(trade.evals),
        main = "Scree plot for the trade network")
```




```
# Plot the first two principal components
plot(trade.eig$vectors[,1:2],type="n",asp=1,xlab="PC 1",
     ylab="PC 2", main = "Loading Plot (first 2 Principal Components)")
abline(h=0,v=0,lty=3)
arrows(0,0,trade.eig$vectors[,1],trade.eig$vectors[,2],col=2)
text(trade.eig$vectors[,1:2],label=rownames(trade))
```



Observations

- The scree plot indicates that about 55% of the variation in these networks can be explained by the first principal component. Further, the bulk of the variation (~70%) can be explained by 3 dimensions (first 3 principal components).
- The loading plot for the first two principal components is shown above. We see that all of the trade networks are negatively correlated with the first principal component.
- Further, the MINERALS trade network shows a high positive correlation with the second principal component, and the FOODS trade network shows a high negative correlation with the second PC.
- It is seen that the MINERALS network varies quite differently as opposed to other networks. The other clusters of networks that have similar patterns of adjacency are [DIPLOMATIC_EXCHANGE] and [MANUFACTURED_GOODS, CRUDE_MATERIALS, FOODS]

(c) Discussion

Compare your PCA results to those obtained using MDS. In what ways are they similar? Different?

- Both the MDS analysis as well as the Eigen Vector Decomposition are similar in that they reveal three clusters in the International trade network, the same as those which we had first identified in the Cluster Dendrogram based on Hamming Distance:-
  1. DIPLOMATIC_EXCHANGE
  2. MINERALS
  3. MANUFACTURED_GOODS, CRUDE_MATERIALS, FOODS
- These clusters depict the shared tendency of certain trade network graphs to appear together, with the exchanges/interactions between countries being similar within the clusters. Therefore, with our

dimensionality reduction analysis we can say that analyzing one network from each of these clusters can help us understand the interactions between countries in all 5 networks.