Link Prediction: Inferring the Nasdaq 100 network of correlated social media chatter.

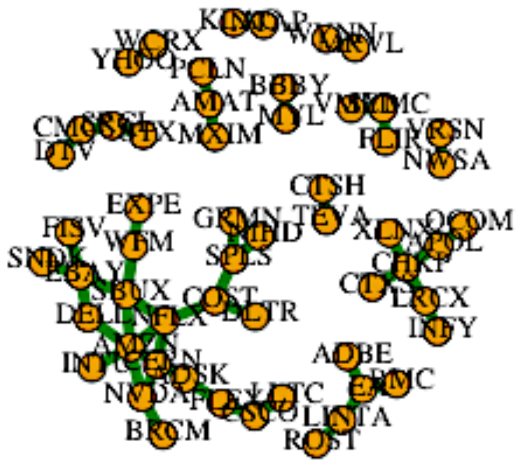
```
#pmayav2

rm(list=ls())
data <-
read.csv("~/Desktop/SocialNetworks/Lab5/AdvancedLab5/wide_twitter_daily_lab5.
csv")
library(tidyverse)

library(igraph)

dim(data)

## [1] 198  93

#heatmap(scale(data[,2:93]), Rowv=NA)

#Calculate the correlation coefficients between each node
mycorr <- cor(data[,2:93])   #92x92
n <- dim(data)[1]

#df of correlations with significant values
mycorrdf <- as.data.frame(as.table(mycorr))
subset(mycorrdf, abs(Freq) > 0.75 & abs(Freq) != 1)

##      Var1 Var2      Freq
## 1444 NVDA BRCM 0.8218245
## 3926 NIHD GRMN 0.7687647
## 3941 SPLS GRMN 0.8145820
## 5655 GRMN NIHD 0.7687647
## 5689 SPLS NIHD 0.7989920
## 5812 BRCM NVDA 0.8218245
## 7035 GRMN SPLS 0.8145820
## 7054 NIHD SPLS 0.7989920
## 8004 YHOO WCRX 0.8108508
## 8459 WCRX YHOO 0.8108508

### METHOD 1 ###
#Partial correlations to predict edges
pcorr.pvals <- matrix(0, dim(mycorr)[1], dim(mycorr)[2])
for(i in seq(1, 92)){
   for(j in seq(1, 92)){
     rowi <- mycorr[i, -c(i, j)]
     rowj <- mycorr[j, -c(i, j)]
     tmp <- (mycorr[i, j] - rowi*rowj)/sqrt((1-rowi^2) * (1-rowj^2))
     tmp.zvals <- (0.5) * log((1+tmp) / (1-tmp))
     tmp.s.zvals <- sqrt(n-4) * tmp.zvals
     tmp.pvals <- 2 * pnorm(abs(tmp.s.zvals), 0, 1, lower.tail=FALSE)
     pcorr.pvals[i, j] <- max(tmp.pvals)
   }
}
pcorr.pvals_df <- as.data.frame(as.table(pcorr.pvals))
#subset(pcorr.pvals_df, abs(Freq) > .99 & abs(Freq) != 1)

#Adjusting for multiple testing
pcorr.pvals.vec <- pcorr.pvals[lower.tri(pcorr.pvals)]
pcorr.pvals.adj <- p.adjust(pcorr.pvals.vec, "BH")
#Benjamini-Hochberg adjustment to control for the false discovery rate

pcorr.edges <- (pcorr.pvals.adj < 0.05)
length(pcorr.pvals.adj[pcorr.edges])

## [1] 48
#edges discovered applying a nominal threshold of 0.05.

# Create the graph predicted by the statistically significant partial
correlations
pcorr.A <- matrix(0, 92, 92)
pcorr.A[lower.tri(pcorr.A)] <- as.numeric(pcorr.edges)
g.pcorr <- graph.adjacency(pcorr.A, "undirected")
g.pcorr

## IGRAPH 9e1ff29 U--- 92 48 --
## + edges from 9e1ff29:
##  [1]  1--31  3-- 9  3--38  7--59  7--69  9--28  9--47  9--61  9--64  9--72
## [11] 10--20 10--70 12--60 15--31 16--64 19--61 19--64 20--27 20--54 20--90
## [21] 22--30 22--78 23--29 23--61 23--77 24--38 24--53 26--81 28--32 31--52
## [31] 32--37 32--72 32--76 34--88 39--80 43--62 43--77 46--54 50--63 52--71
## [41] 57--89 61--72 62--77 65--85 72--88 78--79 80--83 87--92

#FROM PARTIAL CORRELATION
#Degrees
g_degrees = degree(g.pcorr)
table(g_degrees)

## g_degrees
##  0  1  2  3  4  6
## 36 32 15  4  4  1

#components of graph
clusters = clusters(g.pcorr)
#number of components
clusters$no

## [1] 48

# size of clusters: first row is size, the second row is number of components
of that size
table(clusters$csize)
```

```
##
##  1  2  3  4  5  7 22
## 36  6  2  1  1  1  1
```

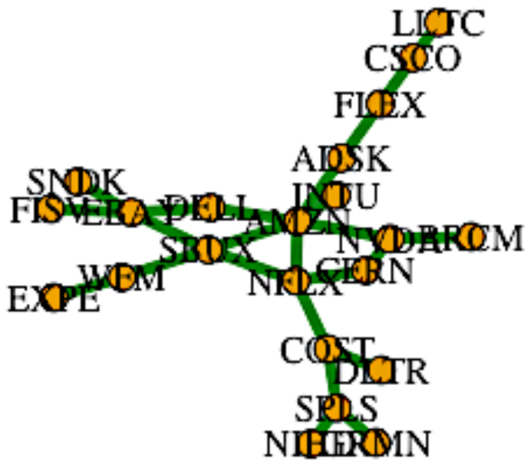
```
#Nodes that belong to giant component
nodes = which(clusters$membership == which.max(clusters$csize))
nodes
```

```
##  [1]  3  9 16 19 23 24 28 29 32 34 37 38 43 47 53 61 62 64 72 76 77 88
```

```
#Plotting partial correlation graphs

#Graph with only nodes with degree > 0
PartialCorrelation <-
read.csv("~/Desktop/SocialNetworks/Lab5/AdvancedLab5/PartialCorrelation_Graph
.csv")
PartialCorrelation <- as.data.frame(PartialCorrelation)
P_EDGES <- PartialCorrelation %>% select(X, X.1)
P_EDGES_1 <- P_EDGES[1:48,]
P_EDGES_1_graph <- graph_from_data_frame(P_EDGES_1, directed = FALSE,
vertices = NULL)
plot(P_EDGES_1_graph, vertex.size=13, edge.color= "forest green",
edge.width=5, vertex.label.color="black", vertex.label.dist=.5,
vertex.label.cex=0.8)
```



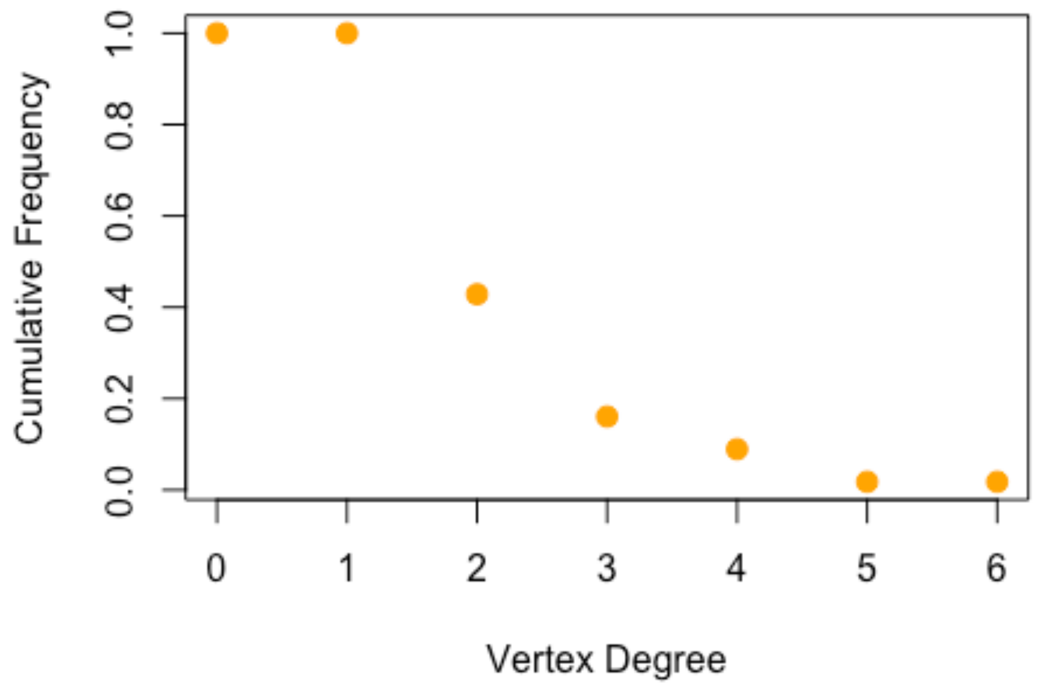
```
#Graph of giant component predicted with partial correlations
giant_component <-
read.csv("~/Desktop/SocialNetworks/Lab5/AdvancedLab5/Giant_component_partial.
csv")
giant_component <- as.data.frame(giant_component)
giant_component1 <- giant_component %>% select(name1,name2)
giant_component_graph <- graph_from_data_frame(giant_component1, directed =
FALSE, vertices = NULL)
plot(giant_component_graph, vertex.size=13, edge.color= "forest green",
vertex.label.color="black", edge.width=5)
```



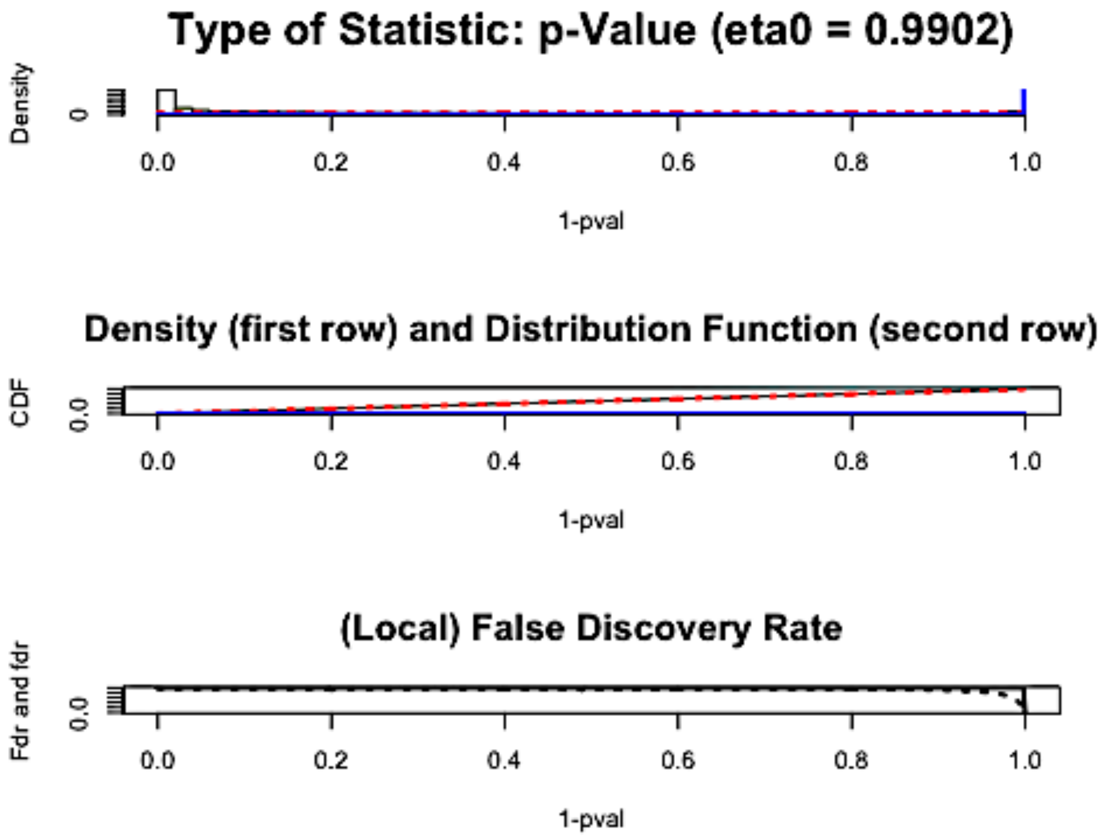
```r
#Degrees of Partial correlation netwkor degree > 0
p_g_degrees <- degree(P_EDGES_1_graph)
table(p_g_degrees)
```

```
## p_g_degrees
##  1  2  3  4  6
## 32 15  4  4  1
```

```r
#Plot the degree distribution of partial correlation's entire network
deg_dist_partial <- degree_distribution(P_EDGES_1_graph, cumulative=T,
mode="all")
plot( x=0:max(p_g_degrees), y=deg_dist_partial, pch=19, cex=1.2,
col="orange",
xlab="Vertex Degree", ylab="Cumulative Frequency")
```

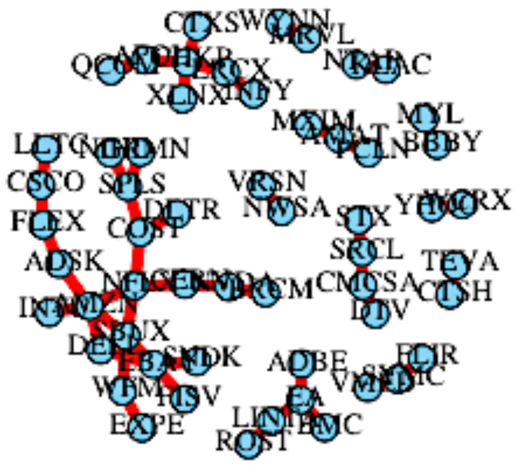


```
### METHOD 2 ###

#FDR tool can also be used to calculate/adjust for false discovery rate (type
I error) under repeated testing and predict new edges based on partial
correlations

#Analogous analysis with FDR
library(fdrtool)
fdr <- fdrtool(pcorr.pvals.vec, statistic="pvalue", plot=TRUE)

## Step 1... determine cutoff point
## Step 2... estimate parameters of null distribution and eta0
## Step 3... compute p-values and estimate empirical PDF/CDF
## Step 4... compute q-values and local fdr
## Step 5... prepare for plotting
```

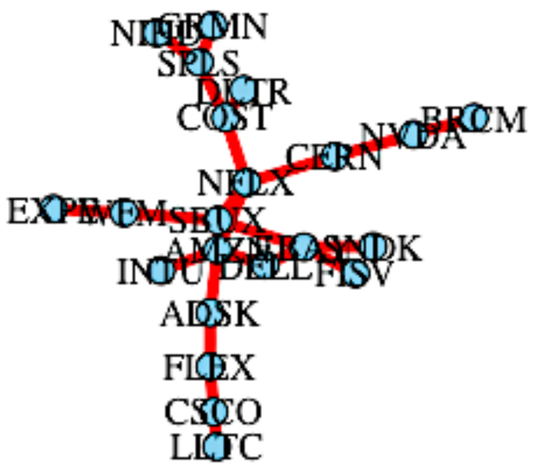


```
pcorr.edges_2 <- (fdr$qval < 0.05)
length(fdr$qval[pcorr.edges_2])

## [1] 47

#47 edges predicted
```

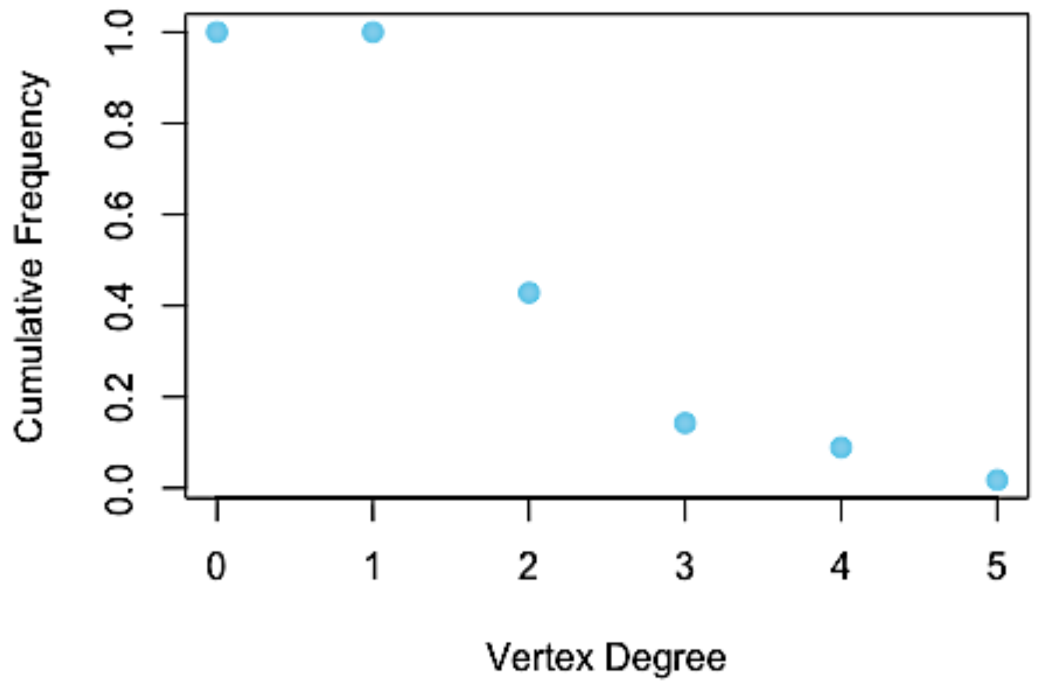
```
#Predicted egdes from FDR
pcorr_fdr <- matrix(0, 92, 92)
pcorr_fdr[lower.tri(pcorr_fdr)] <- as.numeric(pcorr.edges_2)
g.pcorr_fdr <- graph.adjacency(pcorr_fdr, "undirected")
g.pcorr_fdr

## IGRAPH 03f7755 U--- 92 47 --
## + edges from 03f7755:
##  [1]  1--31  3-- 9  3--38  7--59  7--69  9--28  9--47  9--61  9--72 10--20
## [11] 10--70 12--60 15--31 16--64 19--61 19--64 20--27 20--54 20--90 22--30
## [21] 22--78 23--29 23--61 23--77 24--38 24--53 26--81 28--32 31--52 32--37
## [31] 32--72 32--76 34--88 39--80 43--62 43--77 46--54 50--63 52--71 57--89
## [41] 61--72 62--77 65--85 72--88 78--79 80--83 87--92

#with FDR link between nodes 9--64 was not predicted.

#Graphs from DFR
#FDR -> Graph of nodes with degree > 0
fdr_e <- P_EDGES_1[-(9),]
#count(fdr_e) #47
fdr_e_graph <- graph_from_data_frame(fdr_e, directed = FALSE, vertices =
NULL)
plot(fdr_e_graph, vertex.size=13, vertex.color='sky blue',edge.color= "red",
edge.width=5, vertex.label.color="black", vertex.label.dist=.5,
vertex.label.cex=0.8)
```



```
#Graph of giant component predicted with FDR
giant_component_FDR <- giant_component1[-(6),]
giant_component_FDRgraph <- graph_from_data_frame(giant_component_FDR,
directed = FALSE, vertices = NULL)
plot(giant_component_FDRgraph, vertex.size=13, vertex.color='sky blue',
edge.color= "red", vertex.label.color="black", edge.width=5)
```



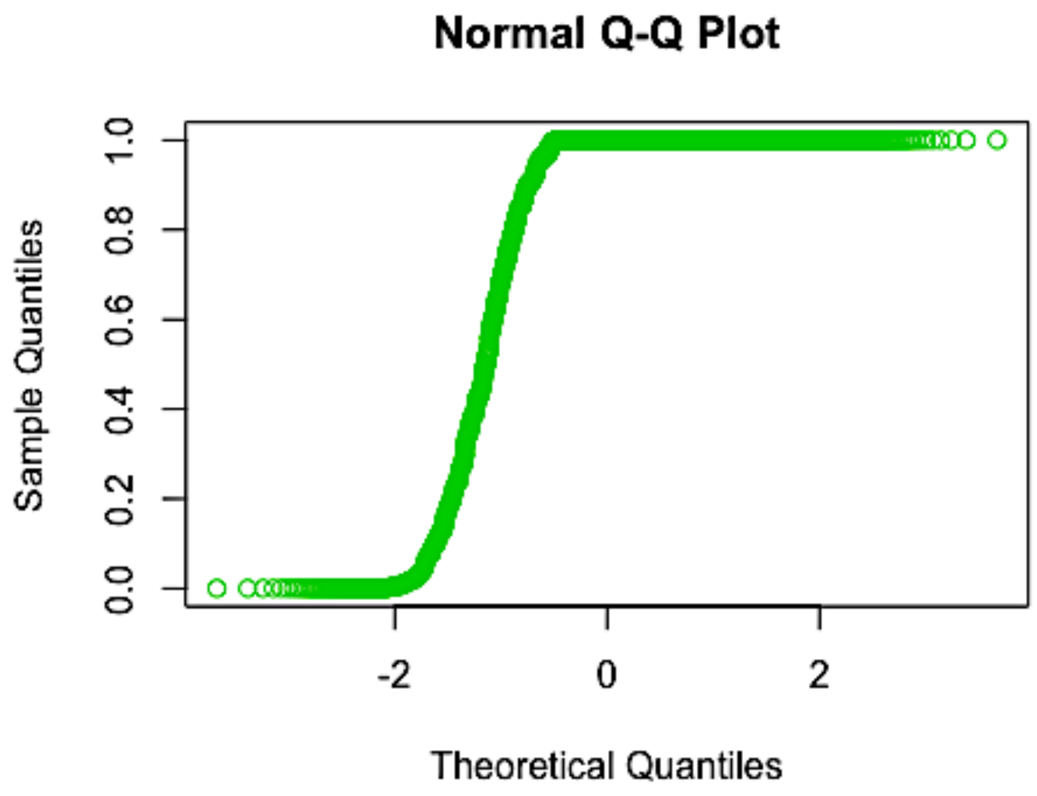
```
#FDR-> Degrees
g_degrees_fdr = degree(fdr_e_graph)
table(g_degrees_fdr)

## g_degrees_fdr
##  1  2  3  4  5
## 32 16  3  4  1

deg_dist_fdr <- degree_distribution(fdr_e_graph, cumulative=T, mode="all")
plot( x=0:max(g_degrees_fdr), y=deg_dist_fdr, pch=19, cex=1.2, col="sky
blue",
xlab="Vertex Degree", ylab="Cumulative Frequency")
```



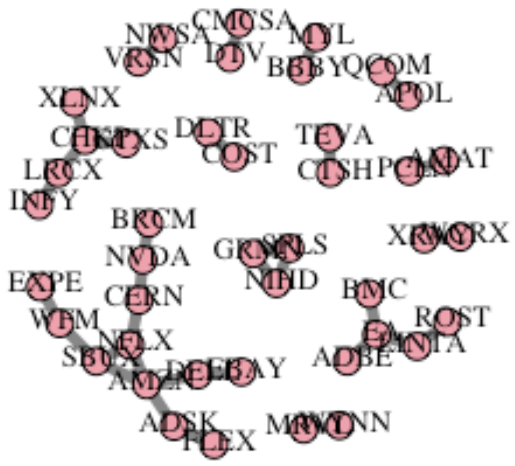
```
#components of graph
clusters_fdr = clusters(fdr_e_graph)
```

```
#number of components
clusters_fdr$no

## [1] 12

# size of clusters: first row is size, the second row is number of components
of that size
table(clusters_fdr$csize)

##
##  2  3  4  5  7 22
##  6  2  1  1  1  1

#Nodes that belong to giant component
nodes = which(clusters_fdr$membership == which.max(clusters_fdr$csize))
nodes

## ADSK AMZN BRCM CERN COST CSCO DELL EBAY EXPE GRMN NFLX NIHD SBUX FLEX INTU
##    2    4    8    9   12   13   15   17   18   20   25   26   28   32   35
## NVDA DLTR SPLS LLTC FISV SNDK  WFM
##   38   43   44   45   47   48   49

#Find overlap between the graph predicted by partial correlations with the
FDR graph
(graph.intersection(P_EDGES_1_graph, fdr_e_graph, byname=FALSE))

## IGRAPH f5000a7 U--- 56 43 --
## + attr: name_1 (v/c), name_2 (v/c)
## + edges from f5000a7:
##  [1] 31--56 30--55 29--54 28--49 27--53 26--44 25--28 24--52 23--51 22--50
## [11] 21--40 20--44 20--26 19--30 18--49 17--48 17--47 17--28 16--23 15--17
## [21] 14--46 13--45 13--32 12--44 12--43 12--25 11--42 11--29 10--41 10--40
## [31] 10--39  9--25  7--16  5--10  4--35  4--28  4--25  4--15  3--34  3--33
## [41]  2--32  2-- 4  1--16

### METHOD 3 ###

#Compute the Fisher’s transformation to approximate the BIVARIATE
distributions and to determine the confidence intervals that are used to
obtain p-values.
#If the pair of variables (Xi,Xj) has a bivariate normal distribution, the
density of p_hat(ij) under H0 : p(ij) = 0 is known to be well approximated by
that of a Gaussian random variable with mean zero and variance 1/(n − 3), for
sufficiently large n.

# Fisher's transformation of the overall correlations
z2 <- 0.5 * log((1 + mycorr) / (1 - mycorr))

#calculate p-values by comparing to the appropriate normal distribution
z2.vec <- z2[upper.tri(z2)]
n <- dim(data)[1]
```

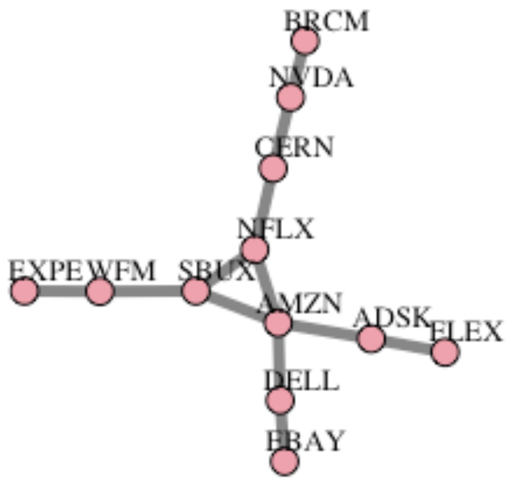
```r
corr.pvals2 <- 2 * pnorm(abs(z2.vec), 0, sqrt(1 / (n-3)), lower.tail=FALSE)
#In assessing these p-values, however, it is necessary to account for the
fact that we are conducting 4186 tests at the same time
length(corr.pvals2)

## [1] 4186

#'p.adjust' may be used to calculate p-values adjusted for multiple testing
#used Benjamin-Hochberg adjustment to control for the false discovery rate
corr.pvals2.adj <- p.adjust(corr.pvals2, "BH")

# Number of edges predicted: using statistical significance at the p < 0.05
threshold
length(corr.pvals2.adj[corr.pvals2.adj < 0.05])

## [1] 180

####

#there is the fact that a large number of tests are to be conducted simulta-
neously (i.e., for all Nv(Nv -1)/2 potential edges), which implies that the
problem of multiple testing must be addressed.

#normal QQ plot of the values in corr.pvals2.adj
qqnorm(corr.pvals2.adj, col = 3)
```



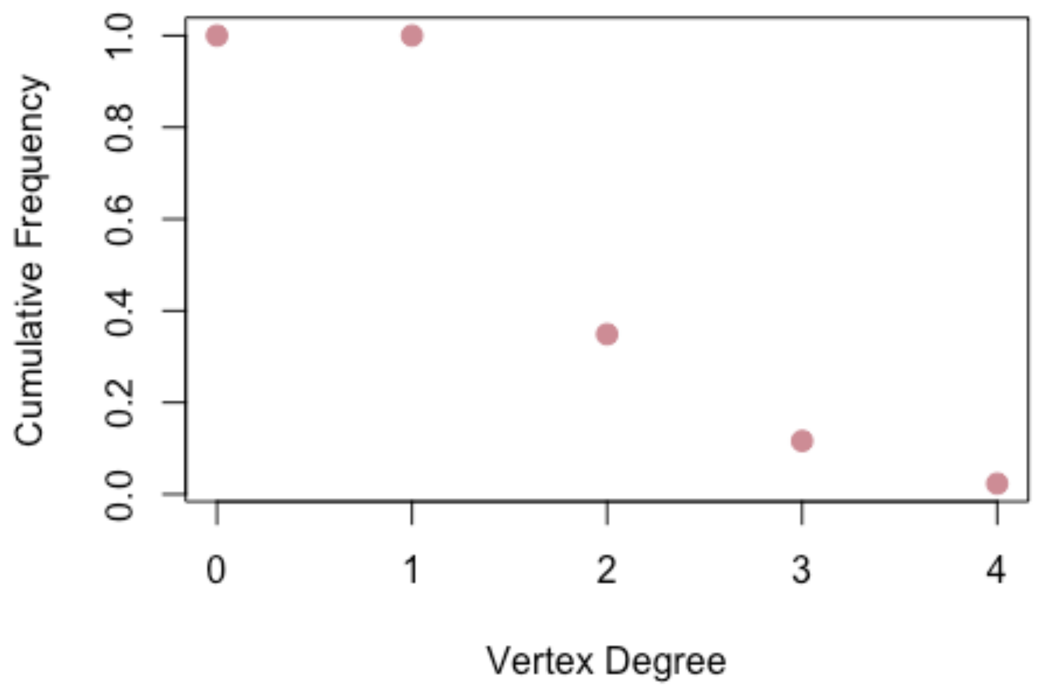
```
### METHOD 4 ###

#Prediction of links using partial correlation and P<0.01
pcorr.edges_P01 <- (pcorr.pvals.adj < 0.01)
length(pcorr.pvals.adj[pcorr.edges_P01])

## [1] 32

# Create the graph
pcorr.P01 <- matrix(0, 92, 92)
pcorr.P01[lower.tri(pcorr.P01)] <- as.numeric(pcorr.edges_P01)
g.pcorr_P01 <- graph.adjacency(pcorr.P01, "undirected")
g.pcorr_P01

## IGRAPH 832591b U--- 92 32 --
## + edges from 832591b:
##  [1]  1--31  3-- 9  3--38  7--69  9--28  9--61  9--72 10--70 12--60 15--31
## [11] 16--64 19--61 19--64 20--27 20--54 20--90 22--30 23--29 26--81 28--32
## [21] 31--52 34--88 43--62 43--77 46--54 52--71 57--89 61--72 62--77 65--85
## [31] 72--88 87--92

#P<0.01
#Plot nodes with degree > 0
PartialCorrelation_P01 <-
read.csv("~/Desktop/SocialNetworks/Lab5/AdvancedLab5/P_01_graph.csv")
PartialCorrelation_P01 <- as.data.frame(PartialCorrelation_P01)
edges_p01 <- PartialCorrelation_P01[c('name1', 'name2')]
g.edges_p01 <- graph_from_data_frame(edges_p01, directed = FALSE, vertices =
NULL)
plot(g.edges_p01, vertex.size=13, vertex.color= 'lightpink2', edge.color=
"gray53", edge.width=5, vertex.label.color="black", vertex.label.dist=.5,
vertex.label.cex=0.8)
```



```
#Graph of giant component predicted with partial correlations
giant_component_p01 <-
read.csv("~/Desktop/SocialNetworks/Lab5/AdvancedLab5/p_01_giantcomponent.csv"
)
giant_component_p01 <- as.data.frame(giant_component_p01)
giant_component_p01_ <- giant_component_p01 %>% select(name1,name2)
g.giant_component_p01_ <- graph_from_data_frame(giant_component_p01_,
directed = FALSE, vertices = NULL)
plot(g.giant_component_p01_, vertex.size=13, vertex.color= 'lightpink2',
edge.color= "gray53", vertex.label.color="black", edge.width=5,
vertex.label.dist=2, vertex.label.cex=0.8)
```



```
#Degrees of Partial correlation netwkork with degree > 0
p_g_degrees_P01 <- degree(g.edges_p01)
table(p_g_degrees_P01)

## p_g_degrees_P01
##  1  2  3  4
## 28 10  4  1

#Plot the degree distribution of partial correlation's entire network
deg_dist_partial_01 <- degree_distribution(g.edges_p01, cumulative=T,
mode="all")

plot( x=0:max(p_g_degrees_P01), y=deg_dist_partial_01, pch=19, cex=1.2,
col="lightpink3",
xlab="Vertex Degree", ylab="Cumulative Frequency")
```



```
#components of graph
clusters_p01 = clusters(g.edges_p01)
#number of components
clusters_p01$no   #13

## [1] 13

# size of clusters: first row is size, the second row is number of components
of that size
table(clusters_p01$csize)

##
##  2  3  5 12
##  9  1  2  1

#Nodes that belong to giant component
nodes_p01 = which(clusters_p01$membership == which.max(clusters_p01$csize))
nodes_p01

## ADSK AMZN BRCM CERN DELL EXPE NFLX SBUX FLEX NVDA EBAY  WFM
##    2    4    8    9   14   16   21   24   26   30   37   38
```