Social Network Analysis for American Football Games

Qinhui Xu, Zezhi Zhou, Boer Feng, Piyush Kashyap

3/5/2017

DATA DESCRIPTION

The file football.gml contains the network of American football games between Division IA colleges during regular season Fall 2000, as compiled by M. Girvan and M. Newman. The nodes have values that indicate to which conferences they belong.

1. SETUP

Load the package we need to use.

2. LOAD DATA AND SET UP ATTRIBUTE

Load the football.gml into the working environment and add some attributes to the nodes

```
wd <- "~/Desktop/GWU/SocialNetwork/Project02/football"
setwd(wd)
football<-read.graph("football.gml",format=c("gml"))
clo = closeness(football)
btw = betweenness(football)
ev_obj = evcent(football)
eigen = ev_obj$vector
V(football)$Degree.Centrality = degree(football)
V(football)$Closeness.Centrality = clo
V(football)$Betweenness.Centrality= btw
V(football)$Eigenvalue.Centrality =eigen
V(football)$id <- V(football)
V(football)$university <- V(football)$label
V(football)$label <- V(football)
num_nodes = vcount(football)
```

3. SRTUCTURE OF THE NETWORK

We will show density, average centrality of the social network here

```r
density <- 613/115
print("Density:", density)
```

```
## [1] "Density:"
```

```r
btn = mean(betweenness(football))
clo = mean(closeness(football))
deg = mean(degree(football))
g_undirected <- as.undirected(football, mode='collapse')
ev_obj <- evcent(g_undirected)
eigen <- mean(ev_obj$vector)
mydf <- do.call(rbind, Map(data.frame, Closeness=clo,
                           Degree=deg,
                           Betweenness=btn,
                           Eigen=eigen))
return(mydf)
```

```
##     Closeness   Degree Betweenness     Eigen
## 1 0.003502799 10.66087    85.96522 0.7186437
```

From the result, we can see that the density of the network is about 5.33, and the average closeness of the network is really low, which is only 0.01. Though we know the meaning of all the centrality measures in gengeral, we don't know the specific meaning of these centrality measurements in this sceanrio. After doing some research about the rules of American Football Competitions, we conclude the meaning of these measurements below:

Degree = How many times does one university have competition with other universities?

Betweenness = Measurement of the winning times for one university

Closeness = 1/ average shortest path of one university node

Eigen = Power of influence of your neighbour university node

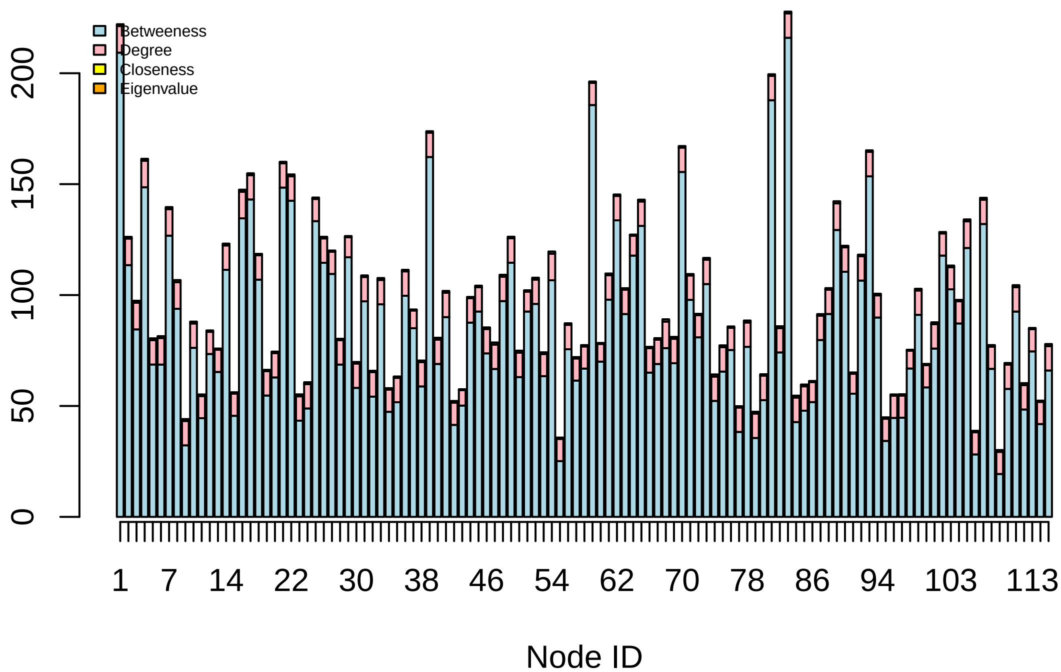
We will explain why the average closeness is so low later.

4. PLOT THE CENTRALITY FOR EACH NODE

```r
g = football
bet <- 0
for (i in 1:num_nodes){
 #print(i)
  bet[i] <- V(g)[i]$Betweenness.Centrality
}
deg <- 0
for (i in 1:num_nodes){
  deg[i] <- V(g)[i]$Degree.Centrality
}
clo <- 0
for (i in 1:num_nodes){
  clo[i] <- V(g)[i]$Closeness.Centrality
```

```
}
eig <- 0
for (i in 1:num_nodes){
  eig[i] <- V(g)[i]$Eigenvalue.Centrality
}
matrix <- rbind(bet,deg,clo,eig)
matrix <- as.matrix(matrix)
xlabbb <-seq(1:num_nodes)
bar <- barplot(matrix,col=c("lightblue","lightpink","yellow","orange"),xlab="Node ID")
axis(1, at=bar, labels=xlabbb)

legend("topleft", c("Betweeness","Degree","Closeness","Eigenvalue"), cex=0.5, bty="n",
       fill = c("lightblue","lightpink","yellow","orange"))
```

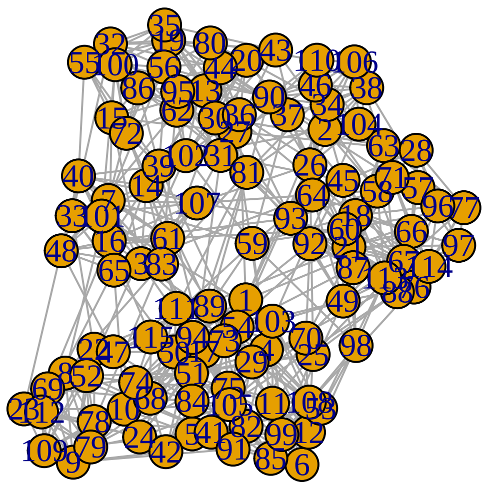


From the barplot, we can see that the betweeness centrality of node 85 is highest.

Node 85 is Oklahomo University, which belongs to Big Twelve Conference

5. SHOW THE NETWORK

```
plot(football,layout=layout.fruchterman.reingold)
```

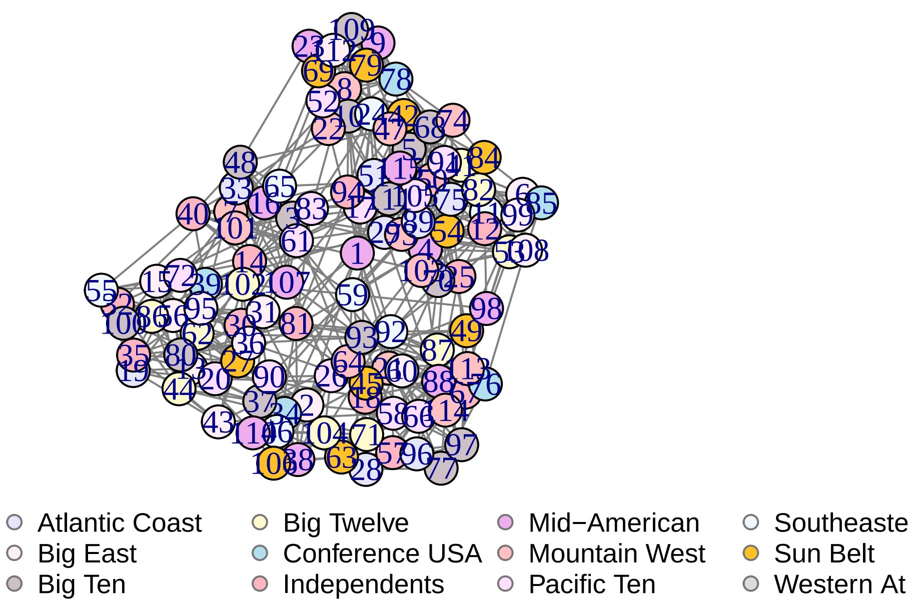


This plots shows the whole netwrok of the football data. Since different node belongs to different conference, we will define the color of vertice based on the value attribute of nodes, which contains the information of conference the node belongs to.

```
colrs <- c("lavender", "lavenderblush", "lavenderblush3","lemonchiffon",
           "lightblue2","lightpink",
           "plum2","rosybrown1","thistle1","aliceblue","goldenrod1","gainsboro")
V(football)$color <- colrs[V(football)$value]
```

```
## Warning in vattrs[[name]][index] <- value: number of items to replace is
## not a multiple of replacement length
```

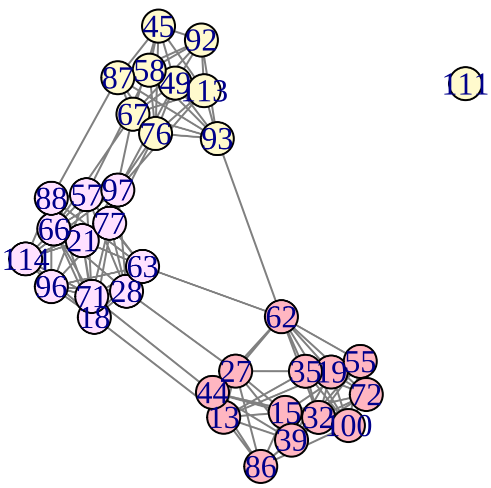
```
plot(football, layout = layout.fruchterman.reingold,
 vertex.color = V(football)$color, edge.color = grey(0.5), edge.arrow.mode = "-")
legend(x=-1.5, y=-1.1, c("Atlantic Coast","Big East", "Big Ten","Big Twelve",
                        "Conference USA","Independents","Mid-American",
                        "Mountain West","Pacific Ten","Southeastern",
                        "Sun Belt","Western Athletic"), pch=21,
   col="#777777", pt.bg=colrs, pt.cex=1, cex=.8, bty="n", ncol=4)
```



From this plot we can have a basic understanding of the conference. But since there are so many nodes in the plot, which makes it a little difficult to see the distribution of the conferences, we will then remove some of the conferences and keep three of them to explain why the closeness of the network is so low.

WHY THE AVERAGE CLOSENESS IS SO LOW?

```
newfootball = football
for (i in c(0,1,2,3,5,7,8,10,11)){
    newfootball <- delete.vertices(newfootball,V(newfootball)[V(newfootball)$value == i])
}
V(newfootball)$color <- colrs[V(newfootball)$value]
plot(newfootball, layout = layout.fruchterman.reingold,edge.color = grey(0.5), edge.arrow.mode = "-")
```



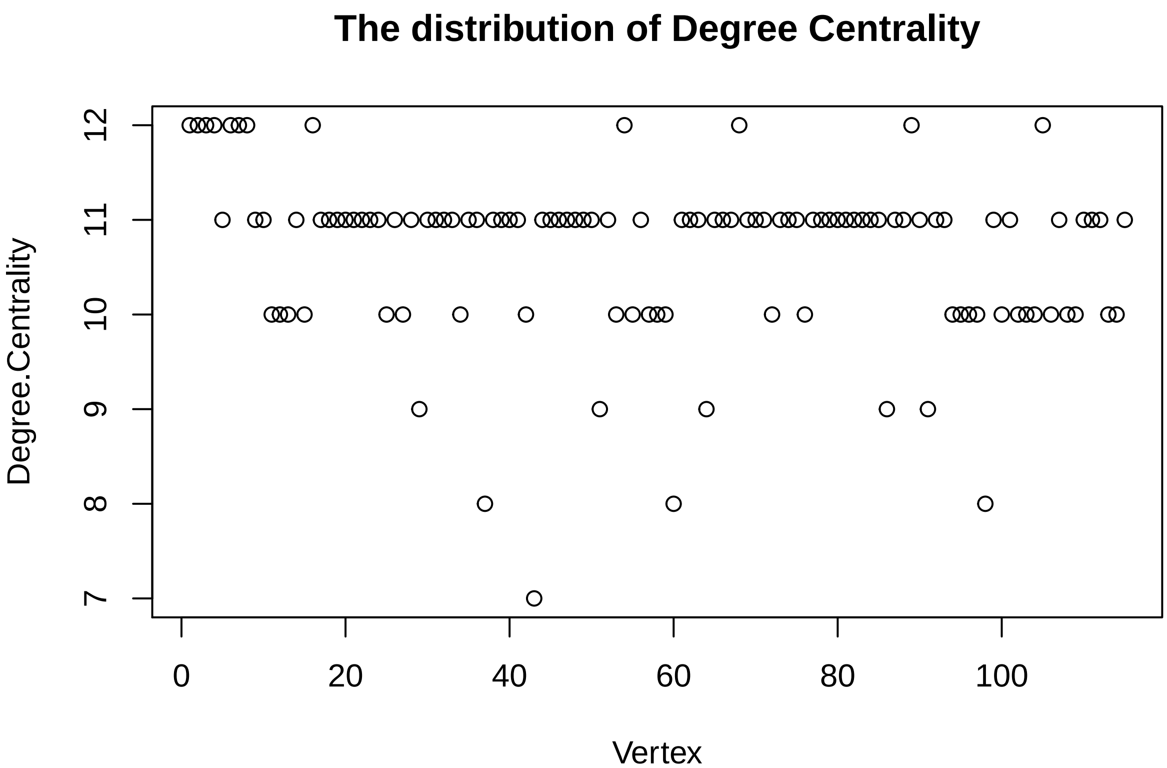
From this plot, we can see that basically nodes in one conference will form an area. The nodes in the center of this area will only connect with the nodes in the outer area. This situation is consistent with the rule for American Football Competition that teams in one conference compete to each other first and then the winner will compete with the teams in other conference.

We can take the node 19 in the pink area as an example. If node 19 wants to connect with node 49, it must connect to node 62 at first, then connect to 93 and 58. The logic is same for every node in the network. Therefore, the shortest average path for every node gets higher, then the closeness will get smaller and smaller.

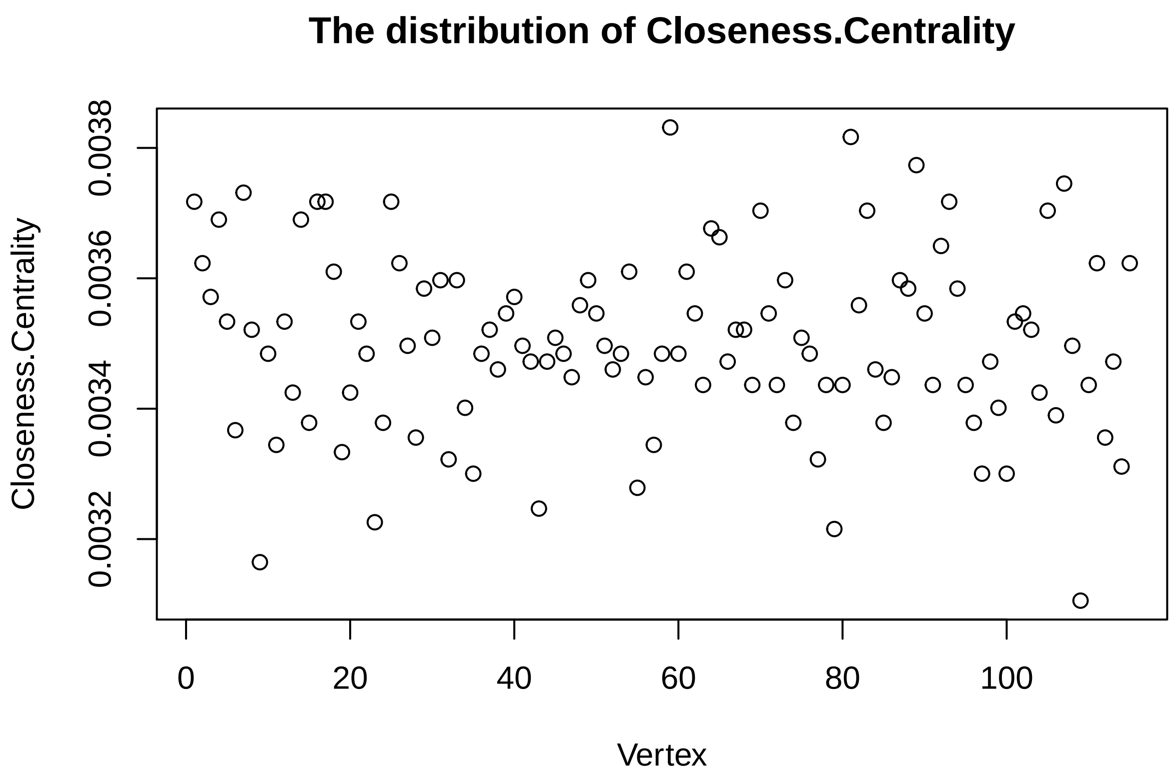
WHAT IS THE DISTRIBUTION OF DIFFERENT CENTRALITY?

```
g=football
closeness_g = closeness(g)
btw_g = betweenness(g)
ev_obj_g = evcent(g)
eigen_g = ev_obj_g$vector
V(g)$Degree.Centrality = degree(g)
V(g)$Closeness.Centrality = closeness_g
```

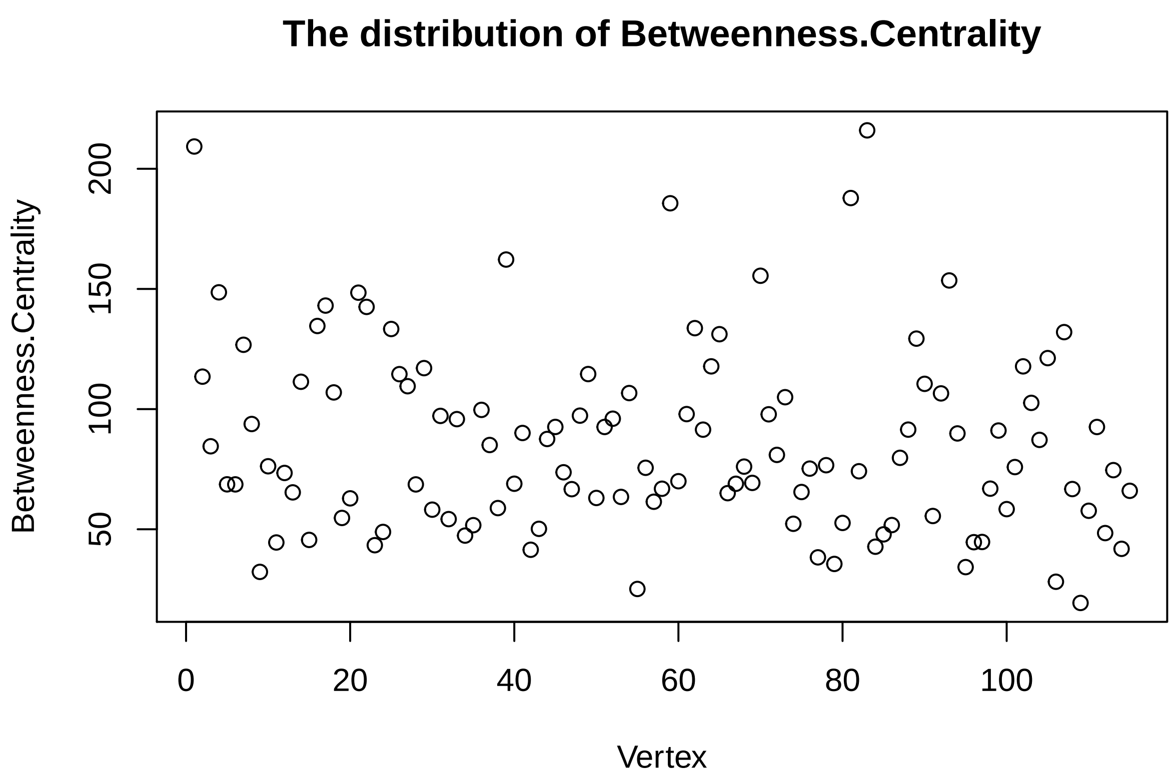
```r
V(g)$Betweenness.Centrality= btw_g
V(g)$Eigenvalue.Centrality =eigen_g
v_size = get.vertex.attribute(g,"Degree.Centrality")
plot(v_size, xlab="Vertex", ylab="Degree.Centrality",
                                main="The distribution of Degree Centrality")
```



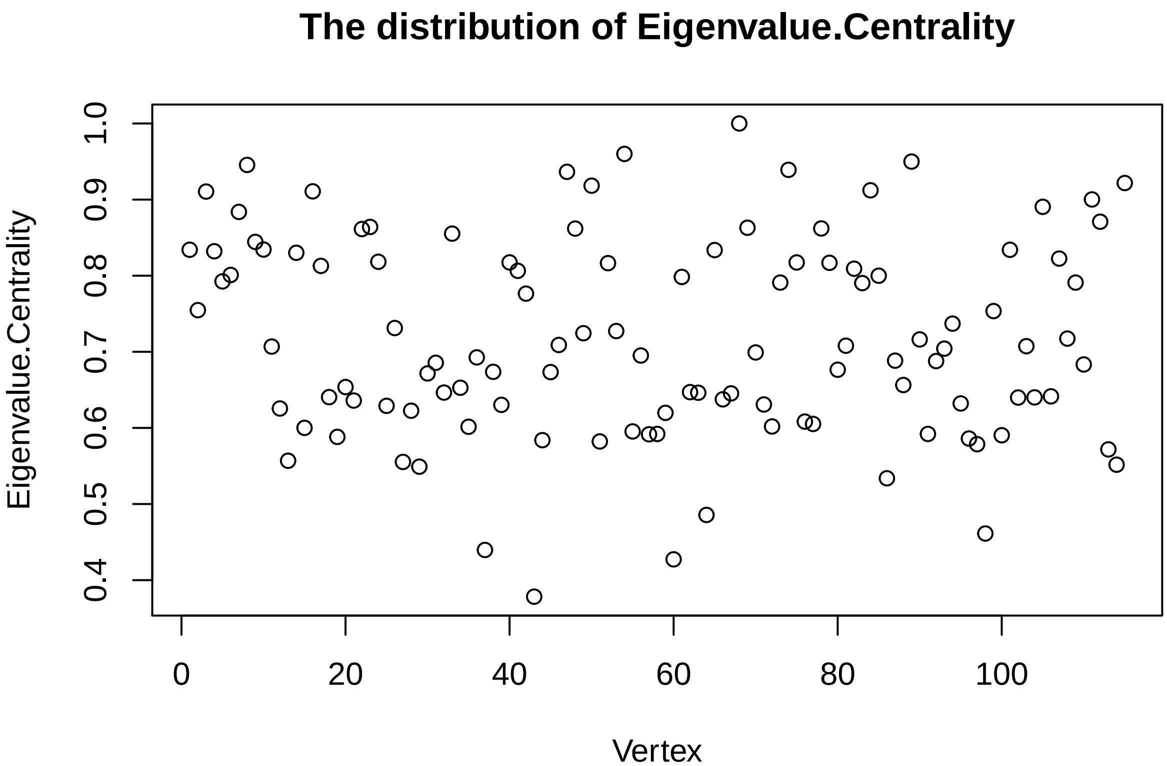

```r
v_size = get.vertex.attribute(g,"Closeness.Centrality")
plot(v_size, xlab="Vertex", ylab="Closeness.Centrality",
                                main="The distribution of Closeness.Centrality")
```




```
v_size = get.vertex.attribute(g,"Betweenness.Centrality")
plot(v_size, xlab="Vertex", ylab="Betweenness.Centrality",
                                main="The distribution of Betweenness.Centrality")
```



```r
v_size = get.vertex.attribute(g,"Eigenvalue.Centrality")
plot(v_size, xlab="Vertex", ylab="Eigenvalue.Centrality",
                                main="The distribution of Eigenvalue.Centrality")
```

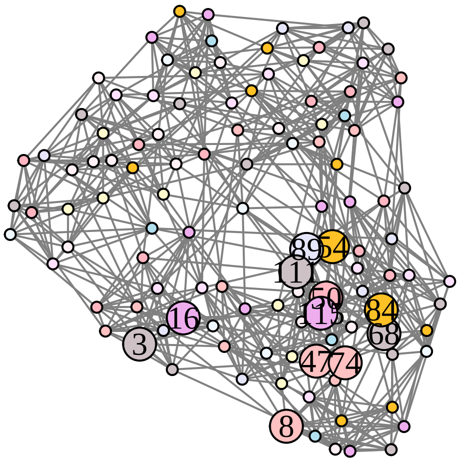


Here we can see the distribution of these four centralities.

WHAT DOES EGIEN CENTRALITY MEAN?

As we mentioned before, eigen centraliy means the power of the neighbor nodes. Now, we will have a more detailed look at the eigen centrality. From the plot of distribution of Egien Centrality avove, we can find that only a few nodes have eigen centrality above 0.9. So, we will make the nodes with eigen centrality higher than 0.9 look bigger.

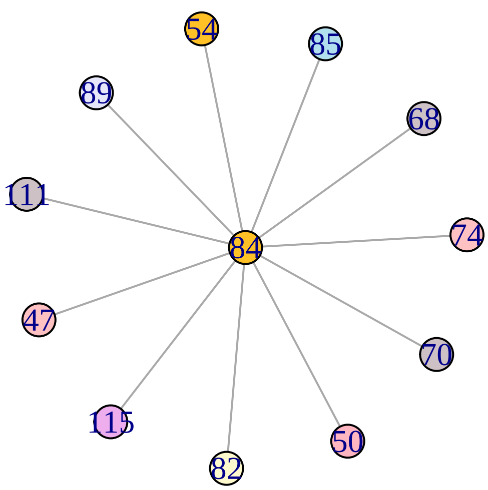
```r
bound = 0.9
V(g)$size = 5
V(g)[Eigenvalue.Centrality>=bound]$size = 15
V(g)$label = NA
V(g)[Eigenvalue.Centrality>=bound]$label = V(g)[Eigenvalue.Centrality>=bound]$id
plot(g, layout = layout.fruchterman.reingold,vertex.color = V(g)$color,
       vertex.label = V(g)$label, vertex.label.color="black",edge.color = grey(0.5),
       edge.arrow.mode = "-")
```



The nodes in bigger size are the nodes with eigen centrality higher than 0.9. We will take node 84 as an example.

Therefore, we need to see a subnetwork that only contains node 84 and the nodes connected to node 84.

```
e = E(football)[from(84)]
sub = subgraph.edges(football, as_ids(e))
plot(sub)
```



Here, we can see that node 84 is connected to node 85, so we think maybe the influence power is related to the winning times which is betweeness centrality.

Therefore, we will explore the relationship between the eigen centrality and the average betweeness of the neighbor nodes.

First, we will define a new attribute:avg_bet for V(football)

```
for (i in 1:115){
  e = E(football)[from(i)]
  sub = subgraph.edges(football, as_ids(e))
  node_num = vcount(sub)-1
  sum_bet = 0
```

```
  list = V(sub)$id
  for (node in list){
    bet = V(football)[node]$Betweenness.Centrality
    sum_bet = bet+sum_bet
  }
  sum_bet = sum_bet - V(football)[i]$Betweenness.Centrality
  avg_bet= sum_bet/node_num
  V(football)[i]$avg_bet = avg_bet
}
V(football)$avg_bet
```

```
##   [1]  80.87474  86.71277 105.96873  89.19635  99.13833  72.19918  99.19464
##   [8]  79.37477  70.79980  86.23675  79.28835 100.38129  86.63935  95.67982
##  [15]  85.99830  91.51181  97.72408  90.02512  75.43011  74.08667  80.03423
##  [22]  71.82297  71.61699  84.78724  90.94055  86.26905  86.52581  81.01006
##  [29]  90.96375 100.04265  87.25059  69.80940 103.43502  91.62326  69.63947
##  [36]  91.40870  91.66933  86.37902  77.23699 115.91147  86.21018  95.55978
##  [43]  78.03970  82.54560  83.62388  83.01471  83.35304  96.30402  86.25850
##  [50]  80.96500  92.53768  69.80183  90.05886  80.08289  78.59325  76.28932
##  [57]  74.09442  92.18870 103.51309  87.79730  98.37157  78.58433  81.20505
##  [64]  85.53833  96.72592  96.11908  91.21940  82.86168  74.04457  87.42271
##  [71]  80.91221  78.85058  85.65841  79.91215  93.04367  90.42611  73.92284
##  [78]  79.14638  67.05522  84.79846  78.94713  96.75169  83.83482  84.55501
##  [85]  78.34158  99.71166 102.82509  85.33989  90.36692  76.74987 103.43745
##  [92] 103.82886  98.76593 121.91475 101.93691  81.22378  81.13047  95.11386
##  [99]  74.19365  68.41069 111.58239  87.54649  83.79434  80.47131  90.78123
## [106]  94.25878 107.32932  82.84209  75.04925  81.94234  94.12114  73.83286
## [113]  85.64744  82.43550  97.34687
```

Now, we can see the average betweenness centraility for all the neighbor nodes of one node.

Then, we will do a linear regression for the avg_bet and eigen for every node.

```
avg_bet = V(football)$avg_bet
node_eigen = V(football)$Eigenvalue.Centrality
slr <- lm(node_eigen~avg_bet)
summary(slr)
```

```
##
## Call:
## lm(formula = node_eigen ~ avg_bet)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.33466 -0.09416 -0.01671  0.10343  0.28396
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 0.663089   0.101680   6.521 2.03e-09 ***
## avg_bet     0.000639   0.001161   0.550    0.583
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1308 on 113 degrees of freedom
## Multiple R-squared:  0.002673,   Adjusted R-squared:  -0.006153
## F-statistic: 0.3029 on 1 and 113 DF,  p-value: 0.5832
```

We can see though p-value is around 0.5, so we cannot reasonably sure that there is relationship between avg_bet and node_eigen.

So, we will define a new attribute: avg_deg for V(football)

```
for (i in 1:115){
  e = E(football)[from(i)]
  sub = subgraph.edges(football, as_ids(e))
  node_num = vcount(sub)-1
  sum_deg = 0
  list = V(sub)$id
  for (node in list){
    bet = V(football)[node]$Degree.Centrality
    sum_deg = bet+sum_deg
  }
  sum_deg = sum_deg - V(football)[i]$Degree.Centrality
  avg_deg= sum_deg/node_num
  V(football)[i]$avg_deg = avg_deg
}
V(football)$avg_deg
```

```
##   [1] 10.750000 10.666667 11.166667 10.750000 10.818182 10.333333 10.916667
##   [8] 11.000000 10.818182 10.909091 10.800000 10.200000 10.300000 11.090909
##  [15] 10.500000 11.000000 11.090909 10.545455 10.000000 10.454545 10.454545
##  [22] 11.000000 11.000000 10.818182 10.300000 10.818182 10.300000 10.545455
##  [29] 10.222222 10.454545 10.636364 10.545455 11.181818 11.000000 10.090909
##  [36] 10.909091 10.375000 10.454545 10.181818 11.090909 10.818182 11.200000
##  [43] 10.428571  9.909091 10.727273 10.818182 11.272727 11.272727 10.818182
##  [50] 11.272727 10.333333 10.636364 11.100000 11.083333 10.700000 10.818182
##  [57] 10.700000 10.400000 10.100000  9.750000 10.818182 10.636364 10.727273
##  [64]  9.666667 11.000000 10.636364 10.454545 11.166667 11.000000 10.272727
##  [71] 10.636364 10.700000 10.818182 11.363636 11.000000 10.700000 10.272727
##  [78] 11.000000 10.636364 10.818182 10.545455 11.000000 10.909091 11.272727
##  [85] 10.818182 10.666667 10.636364 10.636364 11.000000 10.818182 10.555556
##  [92] 10.727273 10.818182 11.100000 10.900000 10.800000 10.700000 10.125000
##  [99] 10.454545 10.500000 11.090909 10.900000 10.700000 10.800000 10.916667
## [106] 10.900000 11.181818 10.900000 11.100000 10.636364 11.181818 11.000000
## [113] 10.100000 10.300000 11.363636
```

Now, we can see the average betweenness centraility for all the neighbor nodes of one node.

Then, we will do a multiple regression for the avg_bet, avg_deg and eigen for every node.

```
avg_bet = V(football)$avg_bet
avg_deg = V(football)$avg_deg
node_eigen = V(football)$Eigenvalue.Centrality
mr <- lm(node_eigen~avg_bet+avg_deg)
summary(mr)
```

```
##
## Call:
## lm(formula = node_eigen ~ avg_bet + avg_deg)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.26498 -0.05158  0.00283  0.05224  0.17792
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -2.3774784  0.2342875  -10.15   <2e-16 ***
## avg_bet     -0.0014365  0.0007366   -1.95   0.0537 .
## avg_deg      0.3004113  0.0222930   13.48   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.08113 on 112 degrees of freedom
## Multiple R-squared:  0.6195, Adjusted R-squared:  0.6127
## F-statistic: 91.19 on 2 and 112 DF,  p-value: < 2.2e-16
```

Therefore, we can be reasonably sure that eigen centrality is something related to average degree and average betweenness of all the neighbor nodes. However, we can see that the p-value for avg_bet is higher than 0.05. So, we will use eta square to see how much proportion of the variance can be uniquely explained by every independent variable.

```
require(heplots)
```

```
## Loading required package: heplots
```

```
## Loading required package: car
```

```
etasq(mr,anova=TRUE,partial=FALSE)
```

```
## Anova Table (Type II tests)
##
## Response: node_eigen
##             eta^2  Sum Sq  Df  F value  Pr(>F)
## avg_bet   0.01279 0.02503   1   3.8025 0.05368 .
```

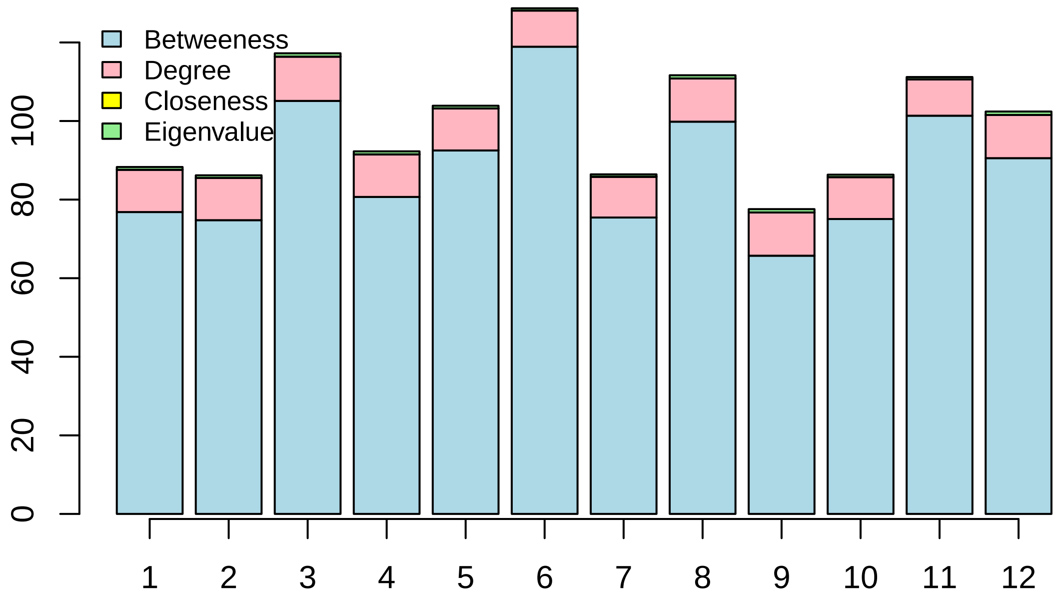
```
## avg_deg   0.61061 1.19523   1 181.5913 < 2e-16 ***
## Residuals         0.73718 112
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

From the result above, we can conclude that there is only 0.1279 of the variance can be explained by avg__bet.

So, average degree of the neighbor nodes is the one that really matters for the eigen centrality.

WHICH CONFERENCE IS MORE ACTIVE?

```r
g = football
bet <- 0
for (i in 0:11){
  bet[i+1] <- sum(V(g)[value==i]$Betweenness.Centrality)/length(V(g)[value==i]$Betweenness.Centrality)
}
deg <- 0
for (i in 0:11){
  deg[i+1] <- sum(V(g)[value==i]$Degree.Centrality)/length(V(g)[value==i]$Degree.Centrality)
}
clo <- 0
for (i in 0:11){
  clo[i+1] <- sum(V(g)[value==i]$Closeness.Centrality)/length(V(g)[value==i]$Closeness.Centrality)
}
eig <- 0
for (i in 0:11){
  eig[i+1] <- sum(V(g)[value==i]$Eigenvalue.Centrality)/length(V(g)[value==i]$Eigenvalue.Centrality)
}
  matrix <- rbind(bet,deg,clo,eig)
  matrix <- as.matrix(matrix)
  xlabbb <-seq(0:11)
  bar = barplot(matrix,col=c("lightblue","lightpink","yellow","lightgreen"))
  axis(1, at=bar, labels=xlabbb)
  legend("topleft", c("Betweeness","Degree","Closeness","Eigenvalue"), cex=0.8, bty="n", fill = c("ligh
```



Here, we can see that conference five (Independents) has the highest Betweeness. Considering the competition rule of American Football, we think this conference win many of other conferences, and then compete with other conferences. Also, the winner of this conference maybe came to the last round of the competition. Following this logic, we take a look at the betweeness of every university node in the conference five.

```
V(football)[value==5]$id
```

```
## [1] 37 43 81 83 91
```

```
V(football)[value==5]$Betweenness.Centrality
```

```
## [1]  85.06001  50.17220 187.82634 215.98577  55.52058
```

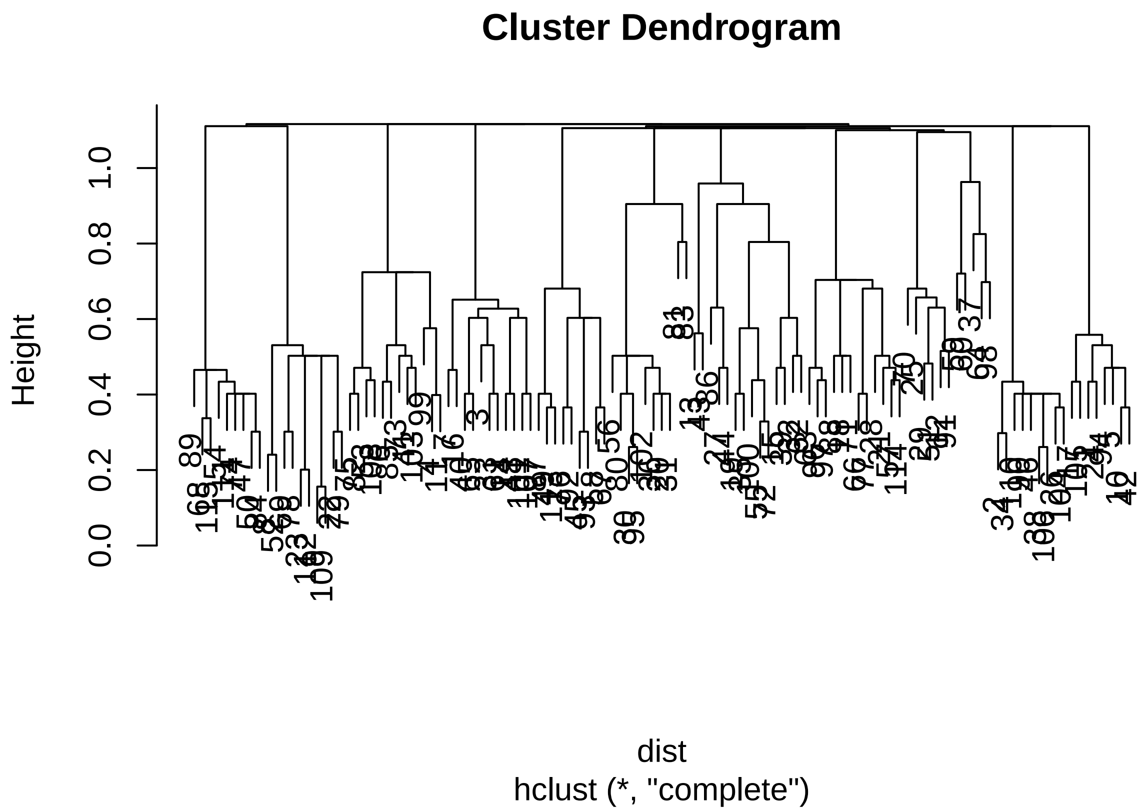
We can see that node 83 has a really high betweeness (top 5 nodes with high Betweenness.Centrality). We think this node should be the winner in this conference.

WHAT DOES BLOCK MEAN IN OUR MODEL

First, let us take a look at our cluster, community and block model.

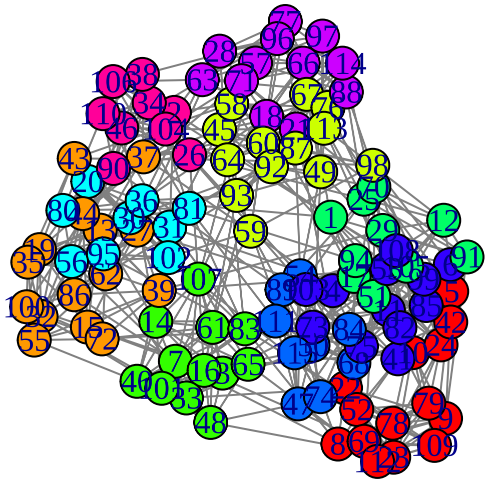
```
  matrix_row_to_col <- get.adjacency(football)
  matrix_row_to_col <-as.matrix(matrix_row_to_col)
  matrix_col_to_row <- t(matrix_row_to_col)
  matrix <- rbind(matrix_row_to_col,matrix_col_to_row  )
  cors <- cor(matrix_row_to_col)
```

```
dissimilarity <- 1 - cors
dist <- as.dist(dissimilarity)
hclust <- hclust(dist)
plot(hclust)
```



This is what cluster looks like.

```
com = walktrap.community(football, steps=10)
V(football)$sg = com$membership
V(football)$color = NA
V(football)$color = rainbow(max(V(football)$sg))[V(football)$sg]
plot(football, layout = layout.fruchterman.reingold, vertex.color =
       V(football)$color, edge.color = grey(0.5), edge.arrow.mode = "-")
```

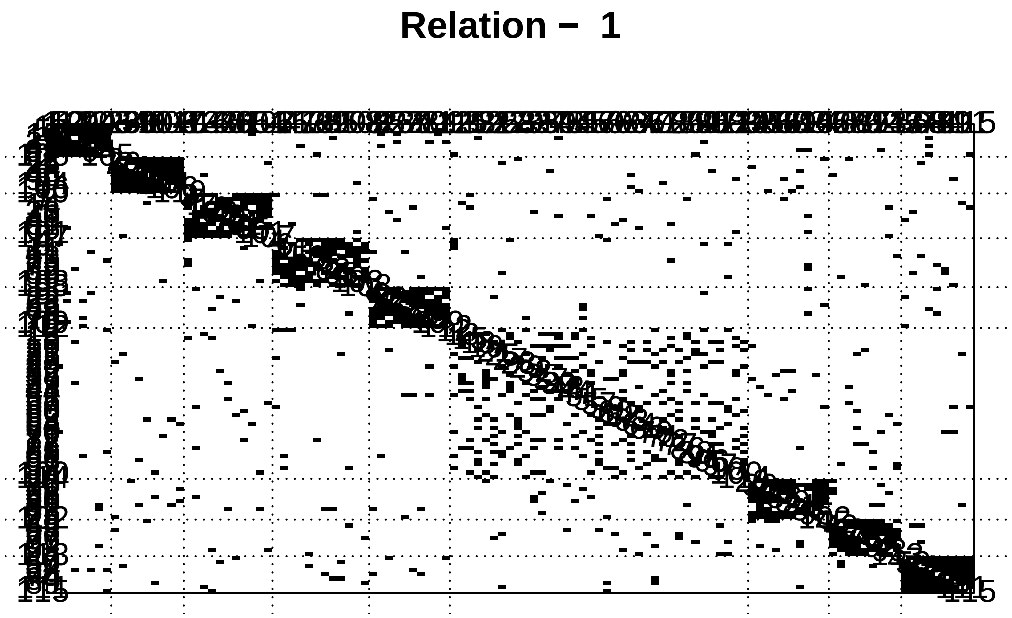


From the community plot, we can see that there are nine blocks

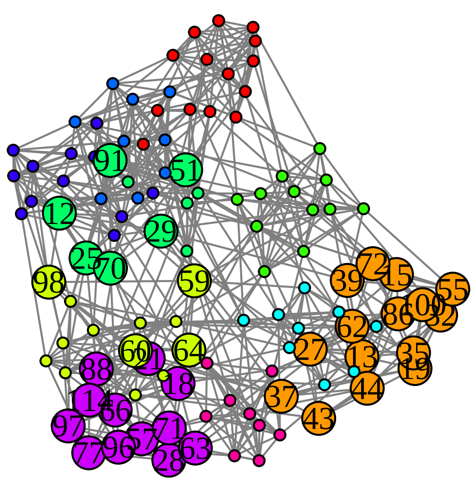
```
matrix_row_to_col <- get.adjacency(football)
matrix_row_to_col <-as.matrix(matrix_row_to_col)
matrix_col_to_row <- t(matrix_row_to_col)
matrix <- rbind(matrix_row_to_col,matrix_col_to_row  )
cors <- cor(matrix_row_to_col)
dissimilarity <- 1 - cors
dist <- as.dist(dissimilarity)
hclust <- hclust(dist)
num_clusters = 9
clusters <- cutree(hclust, k = num_clusters)
cluster_cor_mat <- clusterCorr(cors,clusters)
gcor(cluster_cor_mat, cors)
```

```
## [1] 0.7978018
```

```
mean <- mean(matrix_row_to_col)
blockmodel <- blockmodel(matrix_row_to_col, clusters)
plot(blockmodel,main="blockmodel")
```




```r
V(football)$block = clusters
V(football)$size = 5
V(football)[V(football)$block == 6]$size = 15
V(football)$label = NA
V(football)[V(football)$block == 6]$label = V(football)[V(football)$block == 6]$id
plot(football, layout = layout.fruchterman.reingold, vertex.size = V(football)$size,
     vertex.color = V(football)$color,vertex.label = V(football)$label,
     vertex.label.color="black",edge.color = grey(0.5),edge.arrow.mode = "-")
```



Here we can see the block model plot of our social network.

Form the plot of block model, we can see that there is sparse interaction in block 6. Based on the definition of block, we can know that nodes in one block have similar position and role in the whole network. Therefore, we assume that the block 6 contains nodes which lose in the first round competition in every conference. To explore this, we visualize the nodes in block 6. We can see that the nodes in block 6 are distributed across all the conferences, which can prove our assumption to some extent.

WHICH CNOFERENCE HAS TEAMS THAT COMPETE AMONG ONE ANOTHER?

```r
football <- data.matrix(football)
adj = get.adjacency(football)
adj = as.matrix(adj)
value <- V(football)$value
value <- data.frame(value)
label <- V(football)$label
label <- data.frame(label)
df <- cbind(value,label)
foot <- set_vertex_attr(football, "value", index = V(football), df[,1])
foot <- set_vertex_attr(football, "label", index = V(football), df[,2])
foot <- asNetwork(foot)
ergm_fb <- ergm(foot ~ edges)
```

```
## Evaluating log-likelihood at the estimate.
```

```r
summary(ergm_fb)
```

```
##
## ==========================
## Summary of model fit
## ==========================
##
## Formula:   foot ~ edges
##
## Iterations:  6 out of 20
##
## Monte Carlo MLE Results:
##       Estimate Std. Error MCMC % p-value
## edges -2.27144    0.04242      0  <1e-04 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##      Null Deviance: 9087  on 6555  degrees of freedom
##  Residual Deviance: 4072  on 6554  degrees of freedom
##
## AIC: 4074    BIC: 4081    (Smaller is better.)
```

```r
ergm_fb2 <- ergm(foot ~ edges + nodematch("value"))
```

```
## Evaluating log-likelihood at the estimate.
```

```r
summary(ergm_fb2)
```

```
##
## ==========================
## Summary of model fit
```

```
## ==========================
##
## Formula:   foot ~ edges + nodematch("value")
##
## Iterations:  6 out of 20
##
## Monte Carlo MLE Results:
##                  Estimate Std. Error MCMC % p-value
## edges           -3.27878    0.06883      0  <1e-04 ***
## nodematch.value  4.39532    0.12259      0  <1e-04 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##      Null Deviance: 9087  on 6555  degrees of freedom
##  Residual Deviance: 2467  on 6553  degrees of freedom
##
## AIC: 2471    BIC: 2484    (Smaller is better.)
```

```r
ergm_fb3 <- ergm(foot ~ edges + nodematch("value",diff=T))
```

```
## Evaluating log-likelihood at the estimate.
```

```r
summary(ergm_fb3)
```

```
##
## ==========================
## Summary of model fit
## ==========================
##
## Formula:   foot ~ edges + nodematch("value", diff = T)
##
## Iterations:  13 out of 20
##
## Monte Carlo MLE Results:
##                      Estimate Std. Error MCMC % p-value
## edges                -3.27878    0.06883      0  <1e-04 ***
## nodematch.value.0    20.59079  580.70519      0   0.972
## nodematch.value.1    20.34938  583.58969      0   0.972
## nodematch.value.2     4.66507    0.34406      0  <1e-04 ***
## nodematch.value.3     4.25961    0.28483      0  <1e-04 ***
## nodematch.value.4     4.07371    0.32928      0  <1e-04 ***
## nodematch.value.5     1.08156    1.05634      0   0.306
## nodematch.value.6     3.85860    0.24587      0  <1e-04 ***
## nodematch.value.7    20.34938  583.58969      0   0.972
## nodematch.value.8     5.35822    0.47931      0  <1e-04 ***
## nodematch.value.9     4.25961    0.28483      0  <1e-04 ***
## nodematch.value.10    3.18347    0.44232      0  <1e-04 ***
## nodematch.value.11    3.97193    0.32363      0  <1e-04 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##      Null Deviance: 9087  on 6555  degrees of freedom
##  Residual Deviance: 2374  on 6542  degrees of freedom
```

```
##
## AIC: 2400    BIC: 2488    (Smaller is better.)
```

```r
exp(-2.27144)/(1+exp(-2.27144))
```

```
## [1] 0.09351607
```

First, we do a ergm model to explore the relationship between the probability based on edge formation. We can see that we get a negative edge parameter(p<50%) since the network is rather sparse. The edge parameter (-2.27144) here is the log of the edge odds. The corresponding probability is 0.09351607: exp(-2.27144)/(1+exp(-2.27144)).

To understand that which conference has teams that play more among one another we have to look at the individual coefficient of node match values in following summary fit chart.

We can see that conference eight has the significant p value and highest coefficient, i.e. 5.36, among remaining nodes.

This indicates that team under conference eight has competed more withing among one another in comparison to teams outside its conference.