Wicked Concepts

Rylyn Williams

## Data wrangling

#read in data tables from survey, 35 by 9 data table  
q1 <- read\_csv("data/q1.csv", show\_col\_types = FALSE)

New names:  
• `` -> `...1`

q2 <- read\_csv("data/q2.csv", show\_col\_types = FALSE)

New names:  
• `` -> `...1`

q3 <- read\_csv("data/q3.csv", show\_col\_types = FALSE)

New names:  
• `` -> `...1`

q4 <- read\_csv("data/q4.csv", show\_col\_types = FALSE)

New names:  
• `` -> `...1`

q5 <- read\_csv("data/q5.csv", show\_col\_types = FALSE )

New names:  
• `` -> `...1`

# remove total row so that it's 35 by 8, including column of concepts  
q1 <- select(q1, -9)  
q2 <- select(q2, -9)  
q3 <- select(q3, -9)  
q4 <- select(q4, -9)  
q5 <- select(q5, -9)  
  
head(q1, 2)

# A tibble: 2 × 8  
 ...1 Imperialism Capitalism Racism Colonizaton Housing Education  
 <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  
1 Imperialism 10 15 12 5 4 5  
2 Capitalism 9 10 12 4 3 5  
# ℹ 1 more variable: Infrastructure <dbl>

head(q2, 2)

# A tibble: 2 × 8  
 ...1 `Criminal Justice` Transportation Economy `Income/wealth`  
 <chr> <dbl> <dbl> <dbl> <dbl>  
1 Imperialism 12 5 11 8  
2 Capitalism 7 6 8 9  
# ℹ 3 more variables: `Corporate wealth` <dbl>, `Fed/Gov't Structure` <dbl>,  
# `Gun Control` <dbl>

head(q3, 2)

# A tibble: 2 × 8  
 ...1 Banking `Voting Access` `Climate Change` `Child Poverty` Nationalizing  
 <chr> <dbl> <dbl> <dbl> <dbl> <dbl>  
1 Imperi… 9 4 8 4 11  
2 Capita… 8 7 5 6 9  
# ℹ 2 more variables: `Digital lit` <dbl>, AI <dbl>

head(q4, 2)

# A tibble: 2 × 8  
 ...1 Immigration Misinformation `Pollution & Plastic` Indigenous Sovereign…¹  
 <chr> <dbl> <dbl> <dbl> <dbl>  
1 Imper… 10 10 6 7  
2 Capit… 7 7 8 5  
# ℹ abbreviated name: ¹​`Indigenous Sovereignty`  
# ℹ 3 more variables: `Weakening of Global West` <dbl>, `Public Trust` <dbl>,  
# `Homeless Rights` <dbl>

head(q5, 2)

# A tibble: 2 × 8  
 ...1 `Welfare Systems` `Civil Rights` `Journalism/media` `Foreign Policy`  
 <chr> <dbl> <dbl> <dbl> <dbl>  
1 Imperial… 8 10 6 10  
2 Capitali… 5 6 4 8  
# ℹ 3 more variables: `Insider Trading` <dbl>, `Generational Wealth` <dbl>,  
# Eurocentrism <dbl>

# Join columns by term column, to create 35 by 35  
concepts <- left\_join(q1,q2,by="...1")  
concepts <- left\_join(concepts,q3,by="...1")  
concepts <- left\_join(concepts,q4,by="...1")  
concepts <- left\_join(concepts,q5,by="...1")  
concepts <- select(concepts, -1)  
rownames(concepts)<-colnames(concepts)  
namesss <- colnames(concepts)

`

concepts <- as.matrix(concepts)  
#replace NA's with Zero as the value is not missing, there is just no tie so there weight it 0  
concepts[is.na(concepts)] <- 0  
# Set diag to false to remove self loops  
cg <- graph\_from\_adjacency\_matrix(concepts)  
#Save the graph as a data frame that shows each ties and their weight.  
cg\_frame <-get.data.frame(cg)

### Create a tnet object out of single counted actor ties, with weights being the count of the tie appearence

#Identify unique vertices for the purpose of   
unique\_vertices <- unique(c(cg\_frame$from, cg\_frame$to))  
valid\_vertices <- unique\_vertices[!duplicated(unique\_vertices) & nchar(unique\_vertices) > 0]  
  
# Create an empty graph  
cg\_graph <- graph(edges = numeric(0), directed = FALSE)  
  
# Add vertices to the graph  
cg\_graph <- add\_vertices(cg\_graph, nv = length(valid\_vertices), name = valid\_vertices)  
  
# Count the occurrences of each unique tie  
ties\_count <- table(apply(cg\_frame, 1, function(x) paste(sort(x), collapse = "-")))  
  
# Add subsequent ties of the same kind to the count of the first instance  
unique\_ties <- unique(apply(cg\_frame, 1, function(x) paste(sort(x), collapse = "-")))  
for (tie in unique\_ties) {  
 if (ties\_count[tie] > 1) {  
 first\_instance <- which(apply(cg\_frame, 1, function(x) paste(sort(x), collapse = "-")) == tie)[1]  
 ties\_count[tie] <- ties\_count[tie] + (ties\_count[tie] - 1)  
 ties\_count[tie] <- ties\_count[tie] - 1 # Subtract 1 because we're counting the first instance as unique  
 }  
}  
# Add vertices to the graph  
cg\_graph <- add\_vertices(cg\_graph, nv = length(valid\_vertices), name = valid\_vertices)  
  
# Count the occurrences of each unique tie  
ties\_count <- table(apply(cg\_frame, 1, function(x) paste(sort(x), collapse = "-")))  
  
unique\_ties <- names(ties\_count)  
tie\_parts <- strsplit(unique\_ties, "-")  
from\_vertices <- sapply(tie\_parts, `[`, 1)  
to\_vertices <- sapply(tie\_parts, `[`, 2)  
weights <- as.vector(ties\_count)  
  
# Create a data frame  
cg\_tie\_df <- data.frame(from = from\_vertices, to = to\_vertices, weight = weights)  
  
# Print the data frame  
head(cg\_tie\_df)

from to weight  
1 AI AI 4  
2 AI Banking 6  
3 AI Capitalism 12  
4 AI Child Poverty 6  
5 AI Civil Rights 9  
6 AI Climate Change 12

### creating tnet and statnet object

cg\_tie\_df$from <- as.integer(as.factor(cg\_tie\_df$from))  
cg\_tie\_df$to <- as.integer(as.factor(cg\_tie\_df$to))  
  
# Create the network object  
cg\_tnet <- as.tnet(cg\_tie\_df, type = "weighted one-mode tnet")

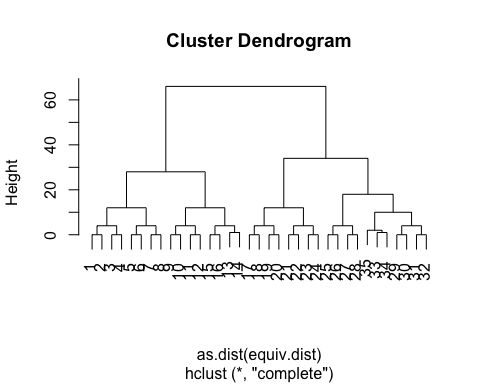
## Node-Level Measures

#Out Degree/ out-strength   
con.outdegree <- degree\_w(cg\_tnet, measure = c("degree", "output"), type="out", alpha = 1)  
#In Degree/ In-strength   
con.indegree <- degree\_w(cg\_tnet, measure = c("degree", "output"), type="in", alpha = 1)  
#closeness  
c\_close <- closeness\_w(cg\_tnet, directed =NULL, gconly = FALSE, alpha = 1)  
#betweeness  
c\_btwn <- betweenness\_w(cg\_tnet, directed =NULL, alpha = 1)  
#constraints  
  
#Rename the columns because the function output names the columns the same regardless of the IN;Out status  
colnames(con.outdegree)[2] <- "Out-Strength"  
colnames(con.outdegree)[3] <- "Out-Degree"  
colnames(con.indegree)[2] <- "In-Strength"  
colnames(con.indegree)[3] <- "In-Degree"  
  
  
#Join the node measures to the same data frame  
con.nodes <-left\_join(as.data.frame(con.outdegree), as.data.frame(con.indegree), by= "node")  
con.nodes <-left\_join(as.data.frame(con.nodes), as.data.frame(c\_close), by= "node")  
con.nodes <-left\_join(con.nodes, as.data.frame(c\_btwn), by= "node")  
  
  
#temporary rename of node column to "name" to join the evigenor centrality for each nodes to the dataset and then "node" was replaced as the variable name for the nodes  
colnames(con.nodes)[1] <- "name"  
cg.stat <- as.network.matrix(cg\_tnet)   
set.vertex.attribute(cg.stat, "name",namesss)   
con.nodes <- left\_join(con.nodes, get.eigen(cg.stat), by = "name")  
colnames(con.nodes)[1] <- "node"  
  
  
head(con.nodes, 15)

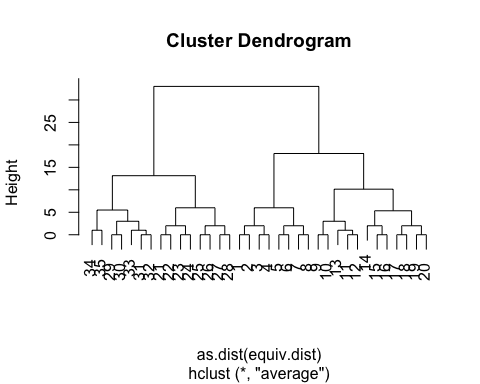
node Out-Strength Out-Degree In-Strength In-Degree closeness n.closeness  
1 1 34 277 0 0 28.35788 0.8340552  
2 2 33 365 1 6 35.96357 1.0577521  
3 3 32 462 2 28 45.11392 1.3268801  
4 4 31 339 3 28 33.69244 0.9909542  
5 5 30 331 4 52 32.79148 0.9644554  
6 6 29 276 5 51 27.54074 0.8100219  
7 7 28 245 6 60 24.57148 0.7226907  
8 8 27 314 7 99 30.99681 0.9116709  
9 9 26 319 8 94 31.62296 0.9300871  
10 10 25 230 9 87 23.55165 0.6926956  
11 11 24 322 10 148 31.62439 0.9301290  
12 12 23 200 11 108 20.01304 0.5886190  
13 13 22 218 12 117 21.67877 0.6376110  
14 14 20 205 13 167 20.95682 0.6163771  
15 15 20 193 14 143 19.20735 0.5649219  
 betweenness eigen eigen.rc eigen.dc  
1 0.0 NaN NaN NaN  
2 0.0 NaN NaN NaN  
3 5.0 NaN NaN NaN  
4 0.0 NaN NaN NaN  
5 0.0 NaN NaN NaN  
6 0.0 NaN NaN NaN  
7 0.0 NaN NaN NaN  
8 4.5 NaN NaN NaN  
9 2.0 NaN NaN NaN  
10 0.0 NaN NaN NaN  
11 11.5 NaN NaN NaN  
12 0.0 NaN NaN NaN  
13 0.0 NaN NaN NaN  
14 5.5 NaN NaN NaN  
15 0.0 NaN NaN NaN

## Strucutual Equivalnce

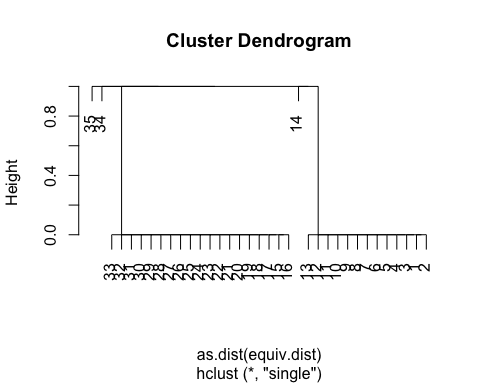
#STRUCTUAL Equivalence   
cg.se <-equiv.clust(cg.stat,  
 equiv.fun = "sedist",  
 method = "hamming",   
 mode = "graph")  
plot(cg.se,labels = cg.se$glabels)



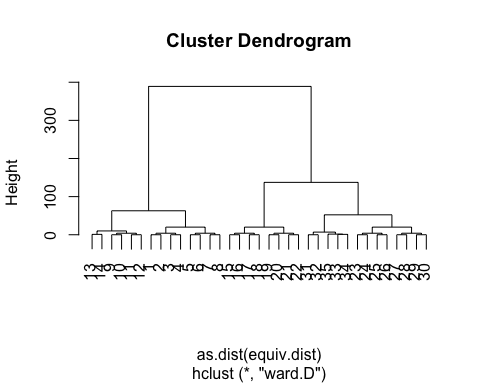
#Average Cluster Method  
cg.ase <- equiv.clust(cg.stat,  
 equiv.fun = "sedist",  
 cluster.method = "average",  
 method = "hamming",   
 mode = "graph")  
plot(cg.ase, cg.ase$glabels)



#Single Cluster Method  
cg.sse<- equiv.clust(cg.stat,  
 equiv.fun = "sedist",  
 cluster.method = "single",  
 method = "hamming",   
 mode = "graph")  
plot(cg.sse,labels = cg.sse$glabels)



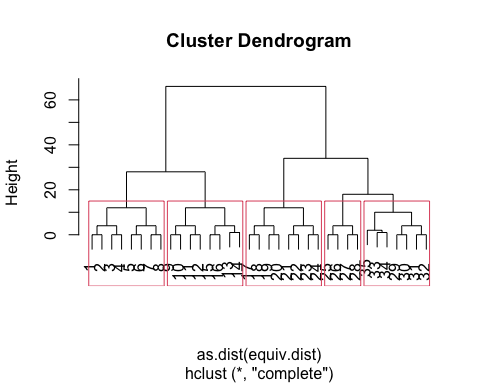
# Ward.D method  
cg.wse<- equiv.clust(cg.stat,  
 equiv.fun = "sedist",  
 cluster.method = "ward.D",  
 method = "hamming",   
 mode = "graph")  
plot(cg.wse,labels = cg.wse$glabels)



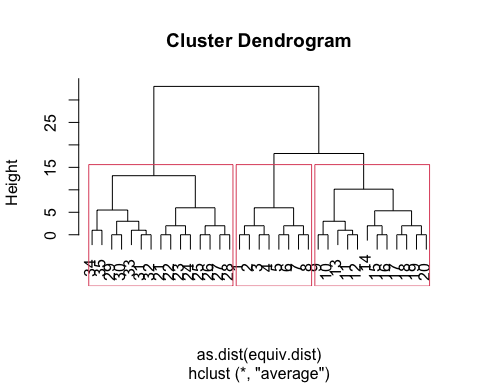
## Partitioning

### Height equal to 15

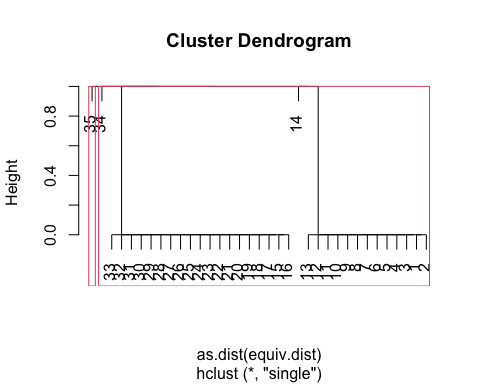
#Partitioning regular clustering - 15  
plot(cg.se,labels = cg.se$glabels)  
rect.hclust(cg.se$cluster, h = 15)



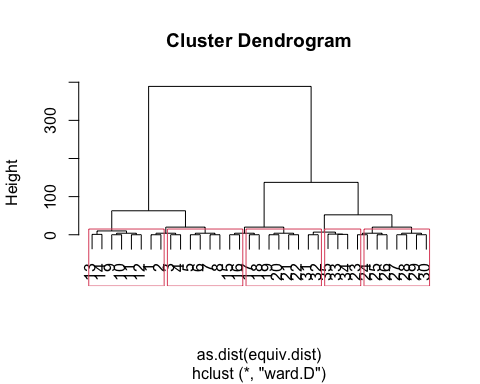
#Partitioning Average Cluster Method - 15  
plot(cg.ase,labels = cg.ase$glabels)  
rect.hclust(cg.ase$cluster, h = 15)



#Partitioning Single Cluster Method - 15  
plot(cg.sse,labels = cg.sse$glabels)  
rect.hclust(cg.sse$cluster, h = 15)

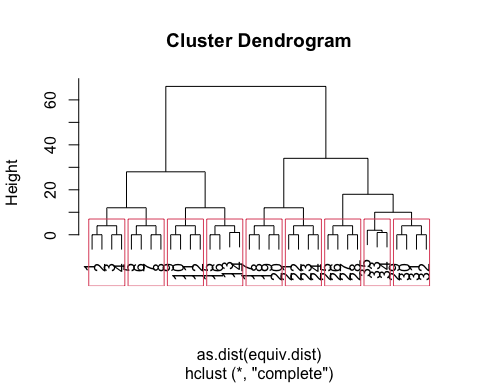


#Partitioning Ward.D method -15  
plot(cg.wse,labels = cg.wse$glabels)  
rect.hclust(cg.se$cluster, h = 15)

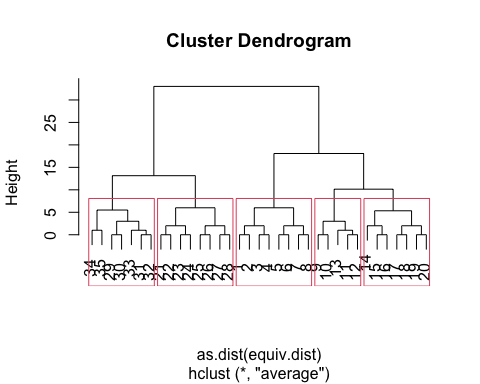


### Height equal to 10

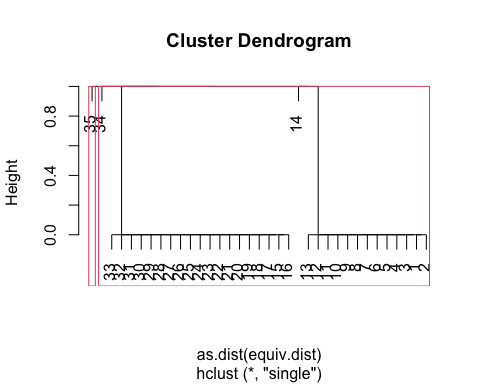
#Partitioning regular clustering -10  
plot(cg.se,labels = cg.se$glabels)  
rect.hclust(cg.se$cluster, h = 10)



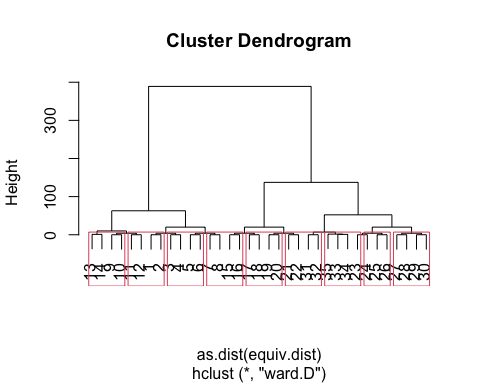
#Partitioning Average Cluster Method -10  
plot(cg.ase,labels = cg.ase$glabels)  
rect.hclust(cg.ase$cluster, h = 10)



#Partitioning Single Cluster Method -10  
plot(cg.sse,labels = cg.sse$glabels)  
rect.hclust(cg.sse$cluster, h = 10)



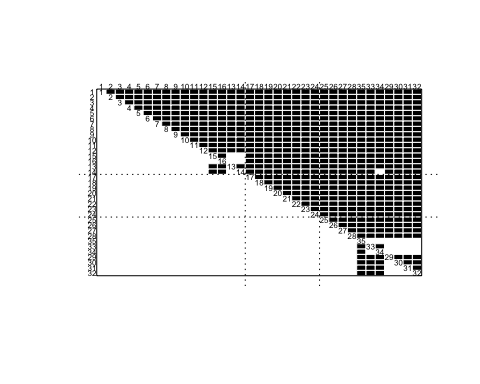
#Partitioning Ward.D method -10  
plot(cg.wse,labels = cg.wse$glabels)  
rect.hclust(cg.se$cluster, h = 10)



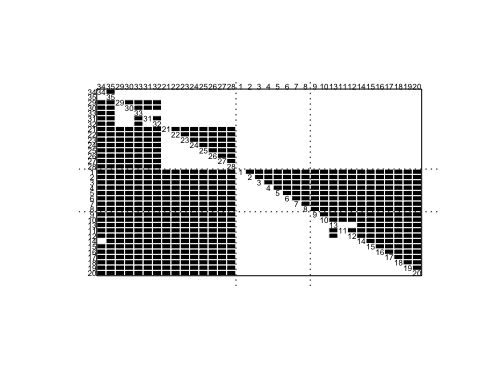
## BlockModeling

### Height at 15 k=3

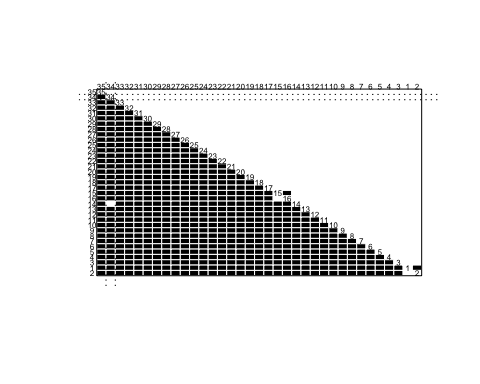
#testing block model  
block\_se <-blockmodel(cg.stat, cg.se, k=3, h=15)  
block\_ase <-blockmodel(cg.stat, cg.ase, k=3, h=15)  
block\_sse <-blockmodel(cg.stat, cg.sse, k=3, h=15)  
block\_wse <-blockmodel(cg.stat, cg.wse, k=5, h=15) #tryin out 5  
  
#View models  
plot.block(block\_se, cex.lab=.5)



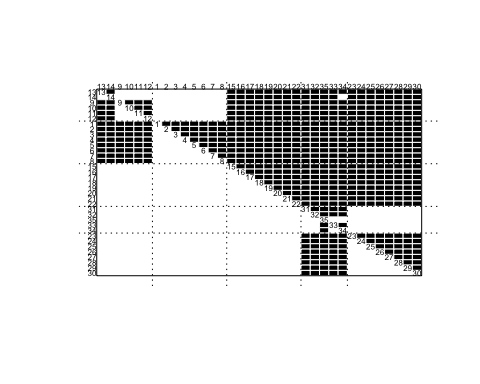
plot.block(block\_ase, cex.lab=.5)



plot.block(block\_sse, cex.lab=.5)

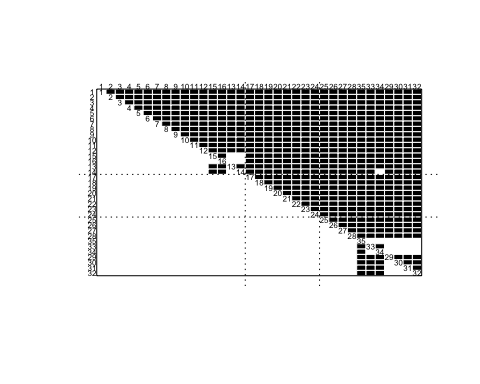


plot.block(block\_wse, cex.lab=.5)

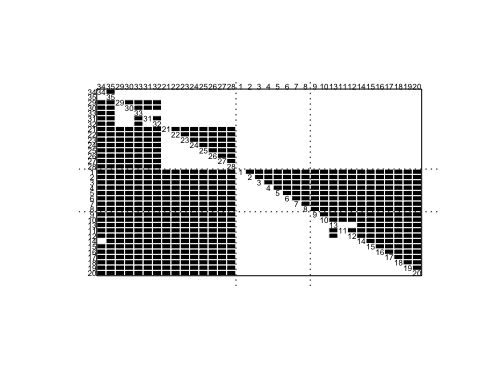


### Height at 10, k=3

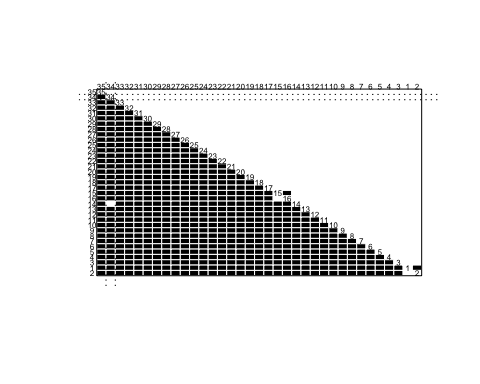
#testing block model  
block\_se <-blockmodel(cg.stat, cg.se, k=3, h=10)  
block\_ase <-blockmodel(cg.stat, cg.ase, k=3, h=10)  
block\_sse <-blockmodel(cg.stat, cg.sse, k=3, h=10)  
block\_wse <-blockmodel(cg.stat, cg.wse, k=5, h=10) #tryin out 5  
  
#View models  
plot.block(block\_se, cex.lab=.5)



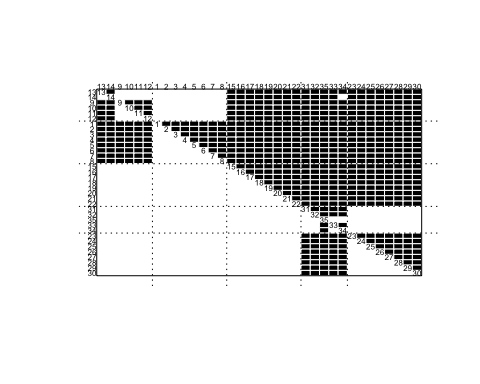
plot.block(block\_ase, cex.lab=.5)



plot.block(block\_sse, cex.lab=.5)



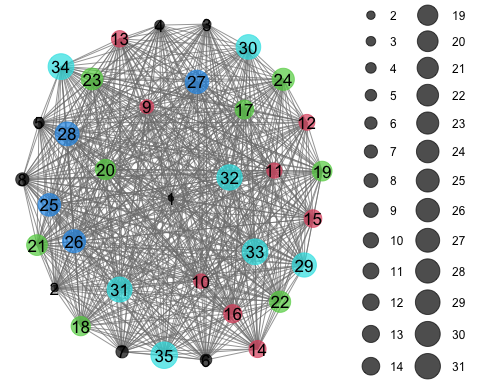
plot.block(block\_wse, cex.lab=.5)



## Let’s Plot this

### Statnet

# chosen blockmodel and chose partition partitions  
cg\_mod <- blockmodel(cg.stat, cg.se, k=5)  
# assign block membership to vertex attribute  
cg.stat%v%"role" <- cg\_mod$block.membership[match(cg.stat%v%"vertex.names",  
 cg\_mod$plabels)]  
  
GGally::ggnet2(cg.stat,  
 node.color = "role",  
 node.size = degree(cg.stat, gmode = "graph"),  
 node.label = "vertex.names",  
 node.alpha = .7)



### igraph

cg.ig <- graph\_from\_data\_frame(cg\_tnet)  
V(cg.ig)$role <- cg\_mod$block.membership[match(V(cg.ig)$name, cg\_mod$plabels)]  
plot.igraph(cg.ig,  
 vertex.color = V(cg.ig)$role,  
 vertex.size = 0.5+(igraph::degree(cg.ig)\*0.5))

