Final Project

Group M

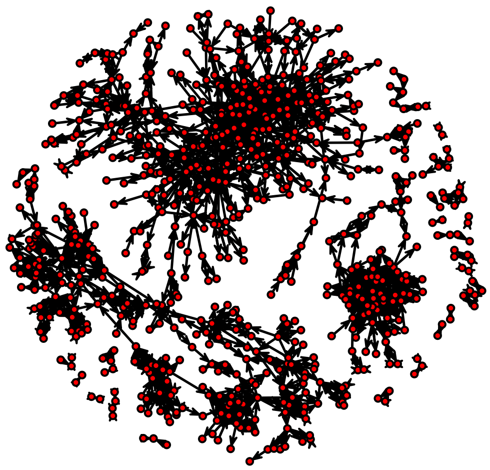
2023-04-30

Read the data

```r
# read the edgelist3.csv
hiv_edges <- read.csv("edgelist_cleaned.csv", header = T, stringsAsFactors = F)

# read the nodes3 CSV file
hiv_nodes <- read.csv("nodes_cleaned.csv", header = T, stringsAsFactors = F)

# creae a network, directed, weighted, with edgelist
hiv_net <- network(hiv_edges, matrix.type="edgelist",directed = T, loops = T)
```

Plot

```r
plot(hiv_net, vertex.col = 'red', displaylabels = F,vertex.cex = 0.7)
```



five-number summary

```r
network.size(hiv_net) # find the size value
```

```
## [1] 694
```

```r
gden(hiv_net) # find the density
```

```
## [1] 0.003761368
```

```r
components(hiv_net) # Components
```

```
## [1] 311
```

```r
diameter( asIgraph (hiv_net) ) # find the diameter
```

```
## [1] 20
```

```r
gtrans(hiv_net,mode="graph") # find the clustering coefficient
```

```
## [1] 0.2143506
```

```r
# Ensure that the node IDs in the network object match the order in the hiv_nodes dataframe
hiv_net %v% "ID" <- as.character(hiv_nodes$ID)

# Assign node attributes to the hiv_net network object
hiv_net %v% "RACE" <- hiv_nodes$RACE
hiv_net %v% "SEX" <- hiv_nodes$SEX
hiv_net %v% "BEHAV" <- hiv_nodes$BEHAV
hiv_net %v% "AGE" <- hiv_nodes$AGE
hiv_net %v% "DISABLE" <- hiv_nodes$DISABLE
hiv_net %v% "UNEMP" <- hiv_nodes$UNEMP
hiv_net %v% "STREETS" <- hiv_nodes$STREETS
hiv_net %v% "EDUC" <- hiv_nodes$EDUC
```

```r
summary(hiv_net,print.adj=FALSE)
```

```
## Network attributes:
##   vertices = 694
##   directed = TRUE
##   hyper = FALSE
##   loops = TRUE
##   multiple = FALSE
##   bipartite = FALSE
##  total edges = 1810
##    missing edges = 0
##    non-missing edges = 1810
##  density = 0.003758025
##
## Vertex attributes:
##
##  AGE:
##    integer valued attribute
##    694 values
##
##  BEHAV:
##    integer valued attribute
##    694 values
##
##  DISABLE:
##    integer valued attribute
##    694 values
##
##  EDUC:
##    integer valued attribute
##    694 values
##
##  ID:
##    character valued attribute
##    attribute summary:
```

```
##    the 10 most common values are:
##      1     10    100   1000 100000 100001 100003 100005 100006 100007
##      1      1      1      1      1      1      1      1      1      1
##
##  RACE:
##    integer valued attribute
##    694 values
##
##  SEX:
##    integer valued attribute
##    694 values
##
##  STREETS:
##    integer valued attribute
##    694 values
##
##  UNEMP:
##    integer valued attribute
##    694 values
##   vertex.names:
##    character valued attribute
##    694 valid vertex names
##
## Edge attributes:
##
##  Weight:
##    integer valued attribute
##    1810values
```

centrality measure

```
# Out-degree centrality
out_degree_centrality <- igraph::degree(asIgraph (hiv_net), mode="out")

# Betweenness centrality
betweenness_centrality <- igraph::betweenness(asIgraph (hiv_net), directed = TRUE, normalized = TRUE)

# Eigenvector centrality
eigenvector_centrality <- igraph::evcent(asIgraph (hiv_net), directed = TRUE)$vector

hiv_nodes$DegreeCentrality <- out_degree_centrality
hiv_nodes$BetweennessCentrality <- betweenness_centrality
hiv_nodes$EigenvectorCentrality <- eigenvector_centrality

attributes <- c("RACE", "SEX", "BEHAV", "AGE", "DISABLE", "UNEMP", "STREETS", "EDUC")
centrality_measures <- c("DegreeCentrality", "BetweennessCentrality", "EigenvectorCentrality")

for (attr in attributes) {
  cat("Centrality measures for", attr, ":\n")
  cat("----------------------------\n")
  attribute_groups <- split(hiv_nodes, hiv_nodes[[attr]])
```

```r
  for (group in names(attribute_groups)) {
    cat("Group", group, ":\n")
    cat("  Mean Degree Centrality: ", mean(attribute_groups[[group]]$DegreeCentrality), "\n")
    cat("  Mean Betweenness Centrality: ", mean(attribute_groups[[group]]$BetweennessCentrality), "\n")
    cat("  Mean Eigenvector Centrality: ", mean(attribute_groups[[group]]$EigenvectorCentrality), "\n\n
  }
  cat("\n")
}
```

```
## Centrality measures for RACE :
## ---------------------------
## Group 1 :
##   Mean Degree Centrality:  1.076923
##   Mean Betweenness Centrality:  0.0002946617
##   Mean Eigenvector Centrality:  3.577915e-17
##
## Group 2 :
##   Mean Degree Centrality:  2.62614
##   Mean Betweenness Centrality:  0.0007092243
##   Mean Eigenvector Centrality:  0.05966844
##
## Group 3 :
##   Mean Degree Centrality:  1.2
##   Mean Betweenness Centrality:  6.646194e-05
##   Mean Eigenvector Centrality:  2.245887e-17
##
## Group 4 :
##   Mean Degree Centrality:  2.674699
##   Mean Betweenness Centrality:  0.000976244
##   Mean Eigenvector Centrality:  0.004258876
##
## Group 5 :
##   Mean Degree Centrality:  2.533333
##   Mean Betweenness Centrality:  0.0008676377
##   Mean Eigenvector Centrality:  0.01674648
##
##
## Centrality measures for SEX :
## ---------------------------
## Group 0 :
##   Mean Degree Centrality:  2.523936
##   Mean Betweenness Centrality:  0.0006484228
##   Mean Eigenvector Centrality:  0.03860347
##
## Group 1 :
##   Mean Degree Centrality:  2.707547
##   Mean Betweenness Centrality:  0.001040309
##   Mean Eigenvector Centrality:  0.02132439
##
##
## Centrality measures for BEHAV :
## ---------------------------
## Group 0 :
```

```
##   Mean Degree Centrality:  2.509091
##   Mean Betweenness Centrality:  0.0007693271
##   Mean Eigenvector Centrality:  0.0258118
##
## Group 2 :
##   Mean Degree Centrality:  2.986111
##   Mean Betweenness Centrality:  0.001052051
##   Mean Eigenvector Centrality:  0.04930256
##
##
## Centrality measures for AGE :
## ---------------------------
## Group 15 :
##   Mean Degree Centrality:  3
##   Mean Betweenness Centrality:  0.0001588275
##   Mean Eigenvector Centrality:  2.08147e-17
##
## Group 16 :
##   Mean Degree Centrality:  1.25
##   Mean Betweenness Centrality:  7.207898e-05
##   Mean Eigenvector Centrality:  1.73936e-17
##
## Group 17 :
##   Mean Degree Centrality:  2.333333
##   Mean Betweenness Centrality:  0.0007214726
##   Mean Eigenvector Centrality:  5.630813e-17
##
## Group 18 :
##   Mean Degree Centrality:  1.5
##   Mean Betweenness Centrality:  0.001194697
##   Mean Eigenvector Centrality:  2.41509e-16
##
## Group 19 :
##   Mean Degree Centrality:  2.2
##   Mean Betweenness Centrality:  0.0006346774
##   Mean Eigenvector Centrality:  0.01566497
##
## Group 20 :
##   Mean Degree Centrality:  1.769231
##   Mean Betweenness Centrality:  0.0005142897
##   Mean Eigenvector Centrality:  3.356064e-17
##
## Group 21 :
##   Mean Degree Centrality:  2.105263
##   Mean Betweenness Centrality:  0.001408733
##   Mean Eigenvector Centrality:  3.455999e-17
##
## Group 22 :
##   Mean Degree Centrality:  3.363636
##   Mean Betweenness Centrality:  0.0009851864
##   Mean Eigenvector Centrality:  0.03341151
##
## Group 23 :
##   Mean Degree Centrality:  4.083333
```

```
##   Mean Betweenness Centrality:  0.0006818227
##   Mean Eigenvector Centrality:  0.08102675
##
## Group 24 :
##   Mean Degree Centrality:  2.64
##   Mean Betweenness Centrality:  0.001632219
##   Mean Eigenvector Centrality:  8.545279e-17
##
## Group 25 :
##   Mean Degree Centrality:  2.764706
##   Mean Betweenness Centrality:  0.001380056
##   Mean Eigenvector Centrality:  9.726511e-17
##
## Group 26 :
##   Mean Degree Centrality:  2.703704
##   Mean Betweenness Centrality:  0.001068683
##   Mean Eigenvector Centrality:  8.083308e-17
##
## Group 27 :
##   Mean Degree Centrality:  2.310345
##   Mean Betweenness Centrality:  0.0008754325
##   Mean Eigenvector Centrality:  0.008229257
##
## Group 28 :
##   Mean Degree Centrality:  2.423077
##   Mean Betweenness Centrality:  0.001267373
##   Mean Eigenvector Centrality:  4.26916e-17
##
## Group 29 :
##   Mean Degree Centrality:  2.111111
##   Mean Betweenness Centrality:  0.00122741
##   Mean Eigenvector Centrality:  0.0192631
##
## Group 30 :
##   Mean Degree Centrality:  3.434783
##   Mean Betweenness Centrality:  0.001473258
##   Mean Eigenvector Centrality:  0.0003185575
##
## Group 31 :
##   Mean Degree Centrality:  2.411765
##   Mean Betweenness Centrality:  0.0007341209
##   Mean Eigenvector Centrality:  0.02482862
##
## Group 32 :
##   Mean Degree Centrality:  2.952381
##   Mean Betweenness Centrality:  0.0007680254
##   Mean Eigenvector Centrality:  0.03411153
##
## Group 33 :
##   Mean Degree Centrality:  2.026316
##   Mean Betweenness Centrality:  0.0003804928
##   Mean Eigenvector Centrality:  0.01825207
##
## Group 34 :
```

```
##   Mean Degree Centrality:  3.782609
##   Mean Betweenness Centrality:  0.001307133
##   Mean Eigenvector Centrality:  0.04860116
##
## Group 35 :
##   Mean Degree Centrality:  1.8
##   Mean Betweenness Centrality:  0.0002747676
##   Mean Eigenvector Centrality:  0.01539213
##
## Group 36 :
##   Mean Degree Centrality:  1.941176
##   Mean Betweenness Centrality:  0.000541201
##   Mean Eigenvector Centrality:  0.01457127
##
## Group 37 :
##   Mean Degree Centrality:  3.5
##   Mean Betweenness Centrality:  0.0004786453
##   Mean Eigenvector Centrality:  0.08766317
##
## Group 38 :
##   Mean Degree Centrality:  2.714286
##   Mean Betweenness Centrality:  0.001010929
##   Mean Eigenvector Centrality:  0.02466741
##
## Group 39 :
##   Mean Degree Centrality:  3.333333
##   Mean Betweenness Centrality:  0.001569572
##   Mean Eigenvector Centrality:  0.04848092
##
## Group 40 :
##   Mean Degree Centrality:  3.208333
##   Mean Betweenness Centrality:  0.0002235184
##   Mean Eigenvector Centrality:  0.1116791
##
## Group 41 :
##   Mean Degree Centrality:  1.3
##   Mean Betweenness Centrality:  7.549464e-05
##   Mean Eigenvector Centrality:  0.02243339
##
## Group 42 :
##   Mean Degree Centrality:  2.9
##   Mean Betweenness Centrality:  0.001002124
##   Mean Eigenvector Centrality:  0.05576334
##
## Group 43 :
##   Mean Degree Centrality:  2.533333
##   Mean Betweenness Centrality:  0.001335686
##   Mean Eigenvector Centrality:  0.004337411
##
## Group 44 :
##   Mean Degree Centrality:  2.5
##   Mean Betweenness Centrality:  0.0007480768
##   Mean Eigenvector Centrality:  0.07718591
##
```

```
## Group 45 :
##   Mean Degree Centrality:  2.333333
##   Mean Betweenness Centrality:  0.0001892859
##   Mean Eigenvector Centrality:  0.1003732
##
## Group 46 :
##   Mean Degree Centrality:  0.9
##   Mean Betweenness Centrality:  0.0006029814
##   Mean Eigenvector Centrality:  0.003398982
##
## Group 47 :
##   Mean Degree Centrality:  2.785714
##   Mean Betweenness Centrality:  0.0003931542
##   Mean Eigenvector Centrality:  0.08859855
##
## Group 48 :
##   Mean Degree Centrality:  1.333333
##   Mean Betweenness Centrality:  0.00016908
##   Mean Eigenvector Centrality:  3.079688e-17
##
## Group 49 :
##   Mean Degree Centrality:  3.777778
##   Mean Betweenness Centrality:  0.0001303908
##   Mean Eigenvector Centrality:  0.048254
##
## Group 50 :
##   Mean Degree Centrality:  1.142857
##   Mean Betweenness Centrality:  0.0002431722
##   Mean Eigenvector Centrality:  0.004820387
##
## Group 51 :
##   Mean Degree Centrality:  3.666667
##   Mean Betweenness Centrality:  0.0007276151
##   Mean Eigenvector Centrality:  0.2730893
##
## Group 52 :
##   Mean Degree Centrality:  4.333333
##   Mean Betweenness Centrality:  0.0006584795
##   Mean Eigenvector Centrality:  3.88786e-17
##
## Group 53 :
##   Mean Degree Centrality:  3.8
##   Mean Betweenness Centrality:  0.0002460344
##   Mean Eigenvector Centrality:  0.09432409
##
## Group 54 :
##   Mean Degree Centrality:  6.333333
##   Mean Betweenness Centrality:  0.0005998594
##   Mean Eigenvector Centrality:  0.06825093
##
## Group 55 :
##   Mean Degree Centrality:  0.5
##   Mean Betweenness Centrality:  2.293788e-05
##   Mean Eigenvector Centrality:  0.06741408
```

```
##
## Group 56 :
##   Mean Degree Centrality:  1
##   Mean Betweenness Centrality:  0
##   Mean Eigenvector Centrality:  2.233593e-17
##
## Group 57 :
##   Mean Degree Centrality:  2
##   Mean Betweenness Centrality:  2.488413e-05
##   Mean Eigenvector Centrality:  0.08365869
##
## Group 58 :
##   Mean Degree Centrality:  2.5
##   Mean Betweenness Centrality:  0.001761164
##   Mean Eigenvector Centrality:  1.401379e-16
##
## Group 61 :
##   Mean Degree Centrality:  3
##   Mean Betweenness Centrality:  7.819733e-05
##   Mean Eigenvector Centrality:  5.679656e-17
##
## Group 63 :
##   Mean Degree Centrality:  1.5
##   Mean Betweenness Centrality:  0
##   Mean Eigenvector Centrality:  2.376963e-17
##
## Group 64 :
##   Mean Degree Centrality:  4.666667
##   Mean Betweenness Centrality:  0.001019924
##   Mean Eigenvector Centrality:  7.614881e-17
##
## Group 66 :
##   Mean Degree Centrality:  4
##   Mean Betweenness Centrality:  0.001085379
##   Mean Eigenvector Centrality:  7.41668e-17
##
## Group 67 :
##   Mean Degree Centrality:  3
##   Mean Betweenness Centrality:  0.0001628454
##   Mean Eigenvector Centrality:  0.04165939
##
## Group 72 :
##   Mean Degree Centrality:  6
##   Mean Betweenness Centrality:  0.003640137
##   Mean Eigenvector Centrality:  1.596276e-16
##
## Group 74 :
##   Mean Degree Centrality:  6
##   Mean Betweenness Centrality:  0.002839224
##   Mean Eigenvector Centrality:  1.229717e-17
##
##
## Centrality measures for DISABLE :
## ---------------------------
```

```
## Group 0 :
##   Mean Degree Centrality:  2.541738
##   Mean Betweenness Centrality:  0.0008635367
##   Mean Eigenvector Centrality:  0.02574563
##
## Group 1 :
##   Mean Degree Centrality:  2.990476
##   Mean Betweenness Centrality:  0.0006423729
##   Mean Eigenvector Centrality:  0.05888929
##
## Group 10 :
##   Mean Degree Centrality:  2
##   Mean Betweenness Centrality:  0.0001400254
##   Mean Eigenvector Centrality:  0
##
##
## Centrality measures for UNEMP :
## ---------------------------
## Group 0 :
##   Mean Degree Centrality:  2.293233
##   Mean Betweenness Centrality:  0.0007658545
##   Mean Eigenvector Centrality:  0.02012783
##
## Group 1 :
##   Mean Degree Centrality:  2.803738
##   Mean Betweenness Centrality:  0.0008666074
##   Mean Eigenvector Centrality:  0.03724779
##
##
## Centrality measures for STREETS :
## ---------------------------
## Group 0 :
##   Mean Degree Centrality:  2.561345
##   Mean Betweenness Centrality:  0.0008472776
##   Mean Eigenvector Centrality:  0.01979941
##
## Group 1 :
##   Mean Degree Centrality:  2.888889
##   Mean Betweenness Centrality:  0.0007120717
##   Mean Eigenvector Centrality:  0.09611525
##
##
## Centrality measures for EDUC :
## ---------------------------
## Group 2 :
##   Mean Degree Centrality:  3.5
##   Mean Betweenness Centrality:  0.0009135118
##   Mean Eigenvector Centrality:  0.0208297
##
## Group 3 :
##   Mean Degree Centrality:  4
##   Mean Betweenness Centrality:  0.001085379
##   Mean Eigenvector Centrality:  7.41668e-17
##
```

```
## Group 4 :
##   Mean Degree Centrality:  3
##   Mean Betweenness Centrality:  7.819733e-05
##   Mean Eigenvector Centrality:  5.679656e-17
##
## Group 5 :
##   Mean Degree Centrality:  2
##   Mean Betweenness Centrality:  0.001213379
##   Mean Eigenvector Centrality:  5.800764e-17
##
## Group 6 :
##   Mean Degree Centrality:  1.5
##   Mean Betweenness Centrality:  6.704118e-05
##   Mean Eigenvector Centrality:  1.260166e-17
##
## Group 7 :
##   Mean Degree Centrality:  3.9375
##   Mean Betweenness Centrality:  0.001415209
##   Mean Eigenvector Centrality:  0.0868347
##
## Group 8 :
##   Mean Degree Centrality:  2.541667
##   Mean Betweenness Centrality:  0.0007599624
##   Mean Eigenvector Centrality:  1.000788e-16
##
## Group 9 :
##   Mean Degree Centrality:  2.130435
##   Mean Betweenness Centrality:  0.0007823877
##   Mean Eigenvector Centrality:  0.008589215
##
## Group 10 :
##   Mean Degree Centrality:  2.681159
##   Mean Betweenness Centrality:  0.000542107
##   Mean Eigenvector Centrality:  0.01918244
##
## Group 11 :
##   Mean Degree Centrality:  2.29703
##   Mean Betweenness Centrality:  0.0008987138
##   Mean Eigenvector Centrality:  0.04173698
##
## Group 12 :
##   Mean Degree Centrality:  2.679245
##   Mean Betweenness Centrality:  0.0008795962
##   Mean Eigenvector Centrality:  0.03501383
##
## Group 13 :
##   Mean Degree Centrality:  2.617021
##   Mean Betweenness Centrality:  0.001062757
##   Mean Eigenvector Centrality:  0.03842029
##
## Group 14 :
##   Mean Degree Centrality:  2.947368
##   Mean Betweenness Centrality:  0.0008205425
##   Mean Eigenvector Centrality:  0.02779866
```

```
##
## Group 15 :
##   Mean Degree Centrality:  1.65
##   Mean Betweenness Centrality:  0.0002193906
##   Mean Eigenvector Centrality:  0.03467
##
## Group 16 :
##   Mean Degree Centrality:  3
##   Mean Betweenness Centrality:  0.0006015295
##   Mean Eigenvector Centrality:  0.002885759
##
## Group 17 :
##   Mean Degree Centrality:  2.5
##   Mean Betweenness Centrality:  0.0009495807
##   Mean Eigenvector Centrality:  1.066693e-17
##
## Group 18 :
##   Mean Degree Centrality:  2
##   Mean Betweenness Centrality:  0.0002671287
##   Mean Eigenvector Centrality:  8.242581e-18
```

community detection

```r
# Community detection using infomap method
set.seed(123)
hiv_net_ig <- asIgraph(hiv_net)
community <- igraph::cluster_infomap(hiv_net_ig , e.weights = E(hiv_net_ig)$Weight)
V(hiv_net_ig)$community <- membership(community)

communities <- unique(V(hiv_net_ig)$community)
community_summary <- data.frame(community = communities,
                                avg_in_degree = length(communities),
                                avg_out_degree = length(communities),
                                avg_edge_weight = length(communities))

for (comm in communities) {
  nodes_in_community <- which(V(hiv_net_ig)$community == comm)

  community_subgraph <- induced_subgraph(hiv_net_ig, nodes_in_community)

  community_summary[comm, "avg_in_degree"] <- mean(igraph::degree(community_subgraph, mode = "in"))
  community_summary[comm, "avg_out_degree"] <- mean(igraph::degree(community_subgraph, mode = "out"))

  edge_weights <- E(community_subgraph)$Weight
  community_summary[comm, "avg_edge_weight"] <- mean(edge_weights)
}

# Sort the data frame by the average in-degree, average out-degree, and average edge weight
community_summary_sorted <- community_summary[order(community_summary$avg_in_degree, community_summary$a

# Print the top 10 communities
print(community_summary_sorted[1:10, ])
```

```
##     community avg_in_degree avg_out_degree avg_edge_weight
## 119       119      4.890909       4.890909        2.527881
## 104       104      3.272727       3.272727        2.805556
## 100       100      3.148148       3.148148        3.894118
## 96         96      3.130435       3.130435        2.375000
## 98         98      3.100000       3.100000        3.483871
## 10         10      2.814815       2.814815        1.921053
## 109       109      2.700000       2.700000        4.111111
## 110       110      2.625000       2.625000        4.238095
## 102       102      2.583333       2.583333        3.290323
## 18         18      2.428571       2.428571        1.823529
```

characteristics of the top 10 communities

```
# Get the top 10 communities
top_communities <- community_summary_sorted[1:10, "community"]

# Define a function to calculate proportions of each attribute
attribute_proportions <- function(attribute, community_nodes) {
  attribute_values <- vertex_attr(hiv_net_ig, attribute, index = V(hiv_net_ig))[community_nodes]
  prop_table <- prop.table(table(attribute_values))
  return(prop_table)
}

# Create a data frame to store the proportions of each attribute in each community
attributes <- c("RACE", "SEX", "BEHAV", "AGE", "DISABLE", "UNEMP", "STREETS", "EDUC")
top_community_characteristics <- data.frame(community = numeric(),
                                            attribute = character(),
                                            category = character(),
                                            proportion = numeric())

# Calculate the proportions for each attribute in each top community
for (comm in top_communities) {
  nodes_in_community <- c(which(V(hiv_net_ig)$community == comm))

  for (attr in attributes) {
    prop_table <- attribute_proportions(attr, nodes_in_community)
    max_prop <- max(prop_table)
    max_prop_category <- names(prop_table)[which.max(prop_table)]

    new_row <- data.frame(community = comm,
                          attribute = attr,
                          category = max_prop_category,
                          proportion = max_prop)

    top_community_characteristics <- rbind(top_community_characteristics, new_row)
  }
}

print(top_community_characteristics)
```

```
##    community attribute category proportion
## 1        119      RACE        2 0.92727273
## 2        119       SEX        0 0.67272727
## 3        119     BEHAV        0 0.72727273
## 4        119       AGE       37 0.16363636
## 5        119   DISABLE        0 0.65454545
## 6        119     UNEMP        1 0.70909091
## 7        119   STREETS        0 0.60000000
## 8        119      EDUC       12 0.45454545
## 9        104      RACE        2 0.90909091
## 10       104       SEX        1 0.54545455
## 11       104     BEHAV        0 0.63636364
## 12       104       AGE       45 0.18181818
## 13       104   DISABLE        0 0.90909091
## 14       104     UNEMP        1 0.90909091
## 15       104   STREETS        0 0.81818182
## 16       104      EDUC       12 0.54545455
## 17       100      RACE        4 0.55555556
## 18       100       SEX        0 0.59259259
## 19       100     BEHAV        0 0.88888889
## 20       100       AGE       22 0.07407407
## 21       100   DISABLE        0 0.81481481
## 22       100     UNEMP        0 0.55555556
## 23       100   STREETS        0 0.96296296
## 24       100      EDUC       12 0.55555556
## 25        96      RACE        4 0.56521739
## 26        96       SEX        0 0.60869565
## 27        96     BEHAV        0 0.95652174
## 28        96       AGE       22 0.08695652
## 29        96   DISABLE        0 0.86956522
## 30        96     UNEMP        1 0.65217391
## 31        96   STREETS        0 0.86956522
## 32        96      EDUC       12 0.34782609
## 33        98      RACE        4 0.50000000
## 34        98       SEX        0 0.50000000
## 35        98     BEHAV        2 0.60000000
## 36        98       AGE       33 0.30000000
## 37        98   DISABLE        0 0.90000000
## 38        98     UNEMP        1 0.60000000
## 39        98   STREETS        0 0.90000000
## 40        98      EDUC        9 0.30000000
## 41        10      RACE        4 0.59259259
## 42        10       SEX        1 0.51851852
## 43        10     BEHAV        0 0.81481481
## 44        10       AGE       30 0.11111111
## 45        10   DISABLE        0 0.96296296
## 46        10     UNEMP        1 0.62962963
## 47        10   STREETS        0 0.96296296
## 48        10      EDUC       12 0.29629630
## 49       109      RACE        4 0.80000000
## 50       109       SEX        0 0.50000000
## 51       109     BEHAV        0 0.90000000
```

```
## 52       109       AGE       38 0.20000000
## 53       109   DISABLE        0 0.90000000
## 54       109     UNEMP        1 0.80000000
## 55       109   STREETS        0 0.90000000
## 56       109      EDUC       12 0.30000000
## 57       110      RACE        2 0.62500000
## 58       110       SEX        0 0.62500000
## 59       110     BEHAV        0 0.87500000
## 60       110       AGE       29 0.25000000
## 61       110   DISABLE        0 0.75000000
## 62       110     UNEMP        1 0.75000000
## 63       110   STREETS        0 0.62500000
## 64       110      EDUC       12 0.37500000
## 65       102      RACE        2 0.58333333
## 66       102       SEX        1 0.66666667
## 67       102     BEHAV        0 0.75000000
## 68       102       AGE       44 0.16666667
## 69       102   DISABLE        0 0.91666667
## 70       102     UNEMP        1 0.75000000
## 71       102   STREETS        0 0.66666667
## 72       102      EDUC       12 0.33333333
## 73        18      RACE        2 0.71428571
## 74        18       SEX        1 0.57142857
## 75        18     BEHAV        0 1.00000000
## 76        18       AGE       19 0.14285714
## 77        18   DISABLE        0 1.00000000
## 78        18     UNEMP        0 0.57142857
## 79        18   STREETS        0 0.85714286
## 80        18      EDUC       12 0.57142857
```

```
library(tidyr)
```

```
## Warning: package 'tidyr' was built under R version 4.2.3

##
## Attaching package: 'tidyr'

## The following object is masked from 'package:igraph':
##
##     crossing
```

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:igraph':
##
##     as_data_frame, groups, union

## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
# Reshape the data frame to have one row per community and attribute category
wide_top_community_characteristics <- top_community_characteristics %>%
  pivot_wider(names_from = attribute, values_from = c(category, proportion), names_sep = "_",
              names_glue = "{attribute}_{.value}") %>%
  select(community, matches("cate"), matches("prop"))

print(wide_top_community_characteristics)
```

ERGM

```
# null model
null_model <- ergm(hiv_net ~ edges)
```

```
## [1] "Warning:  This network contains loops"
```

```
## Starting maximum pseudolikelihood estimation (MPLE):
```

```
## Evaluating the predictor and response matrix.
```

```
## Maximizing the pseudolikelihood.
```

```
## Finished MPLE.
```

```
## Stopping at the initial estimate.
```

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

```
summary(null_model)
```

```
## Call:
## ergm(formula = hiv_net ~ edges)
##
## Maximum Likelihood Results:
##
##       Estimate Std. Error MCMC % z value Pr(>|z|)
## edges -5.58010    0.02355      0    -237   <1e-04 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##      Null Deviance: 667689  on 481636  degrees of freedom
##  Residual Deviance:  23827  on 481635  degrees of freedom
##
## AIC: 23829  BIC: 23840  (Smaller is better. MC Std. Err. = 0)
```

```r
reciprocity_model <- ergm(hiv_net ~ edges + mutual)
```

```
## [1] "Warning:  This network contains loops"

## Starting maximum pseudolikelihood estimation (MPLE):

## Evaluating the predictor and response matrix.

## Maximizing the pseudolikelihood.

## Finished MPLE.

## Starting Monte Carlo maximum likelihood estimation (MCMLE):

## Iteration 1 of at most 60:

## Warning: 'glpk' selected as the solver, but package 'Rglpk' is not available;
## falling back to 'lpSolveAPI'. This should be fine unless the sample size and/or
## the number of parameters is very big.

## Optimizing with step length 0.3227.

## The log-likelihood improved by 3.1936.

## Estimating equations are not within tolerance region.

## Iteration 2 of at most 60:

## Optimizing with step length 0.8023.

## The log-likelihood improved by 3.4464.

## Estimating equations are not within tolerance region.

## Iteration 3 of at most 60:

## Optimizing with step length 1.0000.

## The log-likelihood improved by 0.0180.

## Convergence test p-value: 0.0221. Not converged with 99% confidence; increasing sample size.
## Iteration 4 of at most 60:
## Optimizing with step length 1.0000.
## The log-likelihood improved by 0.0221.
## Convergence test p-value: 0.1892. Not converged with 99% confidence; increasing sample size.
## Iteration 5 of at most 60:
## Optimizing with step length 1.0000.
## The log-likelihood improved by 0.0039.
```

```
## Convergence test p-value: 0.0693. Not converged with 99% confidence; increasing sample size.
## Iteration 6 of at most 60:
## Optimizing with step length 1.0000.
## The log-likelihood improved by 0.0070.
## Convergence test p-value: 0.0308. Not converged with 99% confidence; increasing sample size.
## Iteration 7 of at most 60:
## Optimizing with step length 1.0000.
## The log-likelihood improved by 0.0017.
## Convergence test p-value: < 0.0001. Converged with 99% confidence.
## Finished MCMLE.
## Evaluating log-likelihood at the estimate.

## [1] "Warning:  This network contains loops"

## Fitting the dyad-independent submodel...
## Bridging between the dyad-independent submodel and the full model...
## Setting up bridge sampling...
## Using 16 bridges: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 .
## Bridging finished.
## This model was fit using MCMC.  To examine model diagnostics and check
## for degeneracy, use the mcmc.diagnostics() function.
```

```
summary(reciprocity_model)
```

```
## Call:
## ergm(formula = hiv_net ~ edges + mutual)
##
## Monte Carlo Maximum Likelihood Results:
##
##        Estimate Std. Error MCMC % z value Pr(>|z|)
## edges  -6.03323    0.02998      0  -201.3   <1e-04 ***
## mutual  5.48972    0.08181      0    67.1   <1e-04 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##      Null Deviance: 667689  on 481636  degrees of freedom
##  Residual Deviance:  21161  on 481634  degrees of freedom
##
## AIC: 21165  BIC: 21187  (Smaller is better. MC Std. Err. = 3.441)
```

```
# Fit an ERGM with the selected node attributes
full_model <- ergm(hiv_net ~ edges + nodematch("RACE") + nodematch("SEX") + nodematch("BEHAV") + nodema
```

```
## [1] "Warning:  This network contains loops"

## Starting maximum pseudolikelihood estimation (MPLE):

## Evaluating the predictor and response matrix.

## Maximizing the pseudolikelihood.

## Finished MPLE.
```

```
## Stopping at the initial estimate.

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

```r
summary(full_model)
```

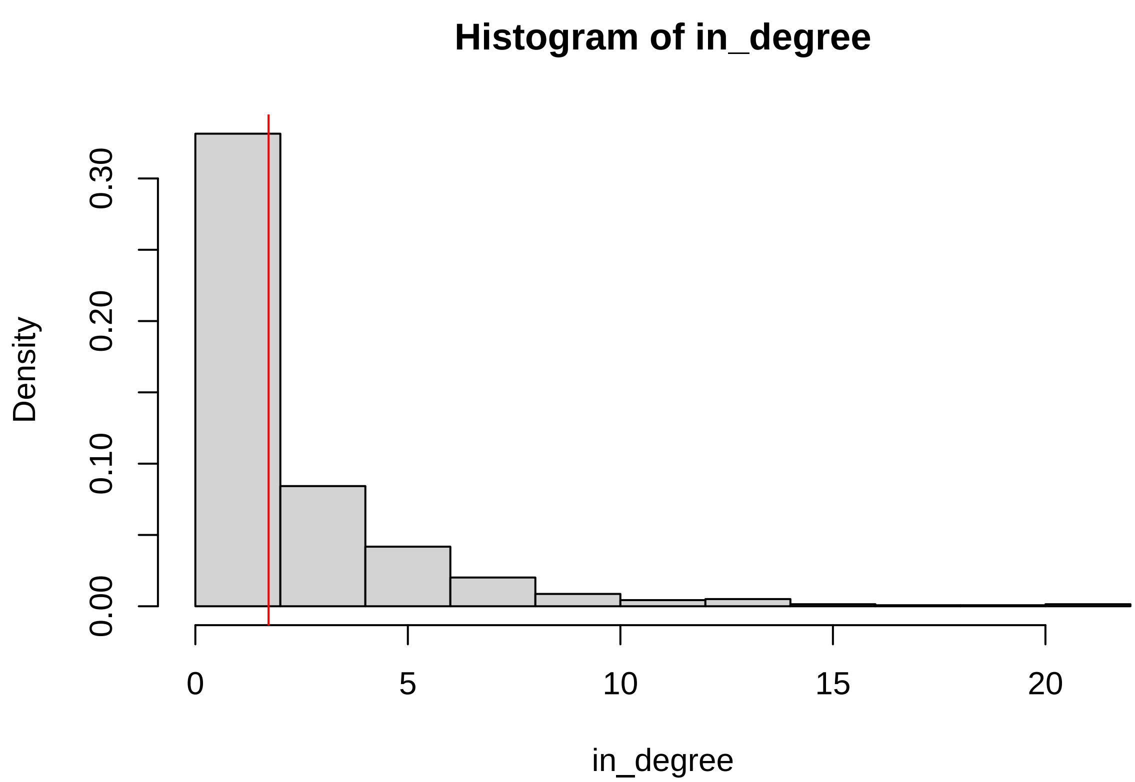
```
## Call:
## ergm(formula = hiv_net ~ edges + nodematch("RACE") + nodematch("SEX") +
##     nodematch("BEHAV") + nodematch("AGE") + nodematch("DISABLE") +
##     nodematch("UNEMP") + nodematch("STREETS") + nodematch("EDUC"),
##     control = control.ergm(MCMC.burnin = 5000, MCMC.interval = 1000))
##
## Maximum Likelihood Results:
##
##                    Estimate Std. Error MCMC % z value Pr(>|z|)
## edges             -5.994068   0.084005      0 -71.354  < 1e-04 ***
## nodematch.RACE     0.577145   0.048095      0  12.000  < 1e-04 ***
## nodematch.SEX      0.051874   0.047320      0   1.096  0.27298
## nodematch.BEHAV   -0.061710   0.049995      0  -1.234  0.21709
## nodematch.AGE     -0.098848   0.138574      0  -0.713  0.47564
## nodematch.DISABLE -0.047755   0.053240      0  -0.897  0.36973
## nodematch.UNEMP    0.125353   0.047509      0   2.639  0.00833 **
## nodematch.STREETS  0.122385   0.057040      0   2.146  0.03190 *
## nodematch.EDUC     0.003197   0.058846      0   0.054  0.95667
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##      Null Deviance: 667689  on 481636  degrees of freedom
##  Residual Deviance:  23661  on 481627  degrees of freedom
##
## AIC: 23679  BIC: 23779  (Smaller is better. MC Std. Err. = 0)
```

```r
# Extract the coefficients from the ERGM fit
coefficients_full <- coef(full_model)

# Sort the coefficients by their absolute values to determine the strongest impact on network formation
sorted_coefficients_full <- coefficients_full[order(abs(coefficients_full), decreasing = TRUE)]

print(sorted_coefficients_full)
```

```
##             edges    nodematch.RACE   nodematch.UNEMP nodematch.STREETS
##      -5.994067604       0.577145311       0.125352507       0.122384864
##     nodematch.AGE   nodematch.BEHAV     nodematch.SEX nodematch.DISABLE
##      -0.098848272      -0.061709721       0.051873831      -0.047754830
##    nodematch.EDUC
##       0.003197121
```

predictive model

```r
in_degree <- igraph::degree(hiv_net_ig, mode = "in")
dens <- density(in_degree)
cumulative_density <- cumsum(dens$y)/sum(dens$y)
threshold <- dens$x[which.min(abs(cumulative_density-0.5))]
hist(in_degree,freq=F)
abline(v = threshold, col = "red")
```




```r
in_degree_binary <- ifelse(in_degree >= threshold, 1, 0)

# Create a data frame with the in_degree_binary and hiv_nodes
hiv_nodes_df <- hiv_nodes
hiv_nodes_df$in_degree <- in_degree_binary
hiv_nodes_df[1:10, ] # show only first 10 rows
```

```
##    ID RACE SEX AGE BEHAV DISABLE UNEMP STREETS EDUC DegreeCentrality
## 1   1    4   1  24     2       0     1       0   12                4
## 2   2    4   1  24     2       0     1       0   12                1
## 3   3    4   0  72     0       0     0       0    5                6
## 4   5    4   1  30     0       0     1       0   12                1
## 5   6    2   0  58     0       0     0       0   12                4
## 6   8    4   0  43     2       0     0       0   13                6
## 7   9    4   1  26     0       0     1       0   14                1
## 8  10    4   1  28     0       0     1       0   12                2
## 9  12    4   0  37     0       0     0       0   12                8
```

```
## 10 13    4   1  33     2       0     1       0   12                5
##    BetweennessCentrality EigenvectorCentrality in_degree
## 1           2.586293e-03          4.666030e-17         1
## 2           0.000000e+00          2.233593e-17         0
## 3           3.640137e-03          1.596276e-16         1
## 4           3.197402e-05          0.000000e+00         1
## 5           3.440683e-05          3.403278e-17         1
## 6           5.517714e-03          1.488510e-16         1
## 7           2.085262e-06          1.104280e-17         0
## 8           0.000000e+00          2.233593e-17         0
## 9           2.571456e-03          4.977793e-18         1
## 10          0.000000e+00          2.233593e-17         0
```

```
# Load the "caret" package for model training
library(caret)
```

```
## Warning: package 'caret' was built under R version 4.2.3
```

```
## Loading required package: ggplot2
```

```
## Warning: package 'ggplot2' was built under R version 4.2.3
```

```
## Loading required package: lattice
```

```
library(ggplot2)

# Prepare the data for training and testing
set.seed(123)
trainIndex <- createDataPartition(hiv_nodes_df$in_degree, p = 0.7, list = FALSE)
trainData <- hiv_nodes_df[trainIndex, ]
testData <- hiv_nodes_df[-trainIndex, ]

# Create a logistic regression model using the glm() function
model <- glm(in_degree ~ .- ID, data = trainData, family = "binomial")

# Print the model summary
summary(model)
```

```
##
## Call:
## glm(formula = in_degree ~ . - ID, family = "binomial", data = trainData)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -4.1003  -0.9493   0.0098   1.0607   1.7273
##
## Coefficients:
##                           Estimate Std. Error z value Pr(>|z|)
## (Intercept)               -0.05749    0.91149  -0.063  0.94971
## RACE                       0.13169    0.11300   1.165  0.24386
## SEX                       -0.13945    0.23828  -0.585  0.55840
## AGE                        0.01764    0.01328   1.329  0.18399
```

```
## BEHAV                    -0.01534    0.14361  -0.107  0.91493
## DISABLE                  -0.14593    0.30804  -0.474  0.63569
## UNEMP                     0.14884    0.22502   0.661  0.50832
## STREETS                  -0.32042    0.34120  -0.939  0.34768
## EDUC                     -0.12775    0.05256  -2.430  0.01508 *
## DegreeCentrality          0.04304    0.06876   0.626  0.53139
## BetweennessCentrality 1719.40035  342.07794   5.026    5e-07 ***
## EigenvectorCentrality   20.29727    7.53407   2.694  0.00706 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 667.73  on 485  degrees of freedom
## Residual deviance: 509.56  on 474  degrees of freedom
## AIC: 533.56
##
## Number of Fisher Scoring iterations: 8
```

```
# Overall, the model indicates that BetweennessCentrality and EigenvectorCentrality are significantly a
```

k-shell decomposition

```
# Calculate k-shell values for each node
k_shells <- coreness(hiv_net_ig)
V(hiv_net_ig)$k_shell <- k_shells

# Find the maximum k-shell value
max_k_shell <- max(k_shells)

# Extract the nodes with the highest k-shell value
key_individuals_k_shell <- V(hiv_net_ig)[k_shell == max_k_shell]

# Print the key individuals
print(key_individuals_k_shell)
```

```
## + 35/694 vertices, from 45c93ab:
##  [1] 343 344 364 369 370 377 439 440 513 515 516 517 518 519 520 521 523 524 525
## [20] 526 527 529 533 538 539 540 549 550 557 558 560 563 565 573 577
```

snowball sampling

```
# Start with a seed node (choose a node ID from your dataset)
seed_node <- 1

# Define the number of steps (depth) for the snowball sampling
steps <- 2
```

```r
# Perform snowball sampling
snowball_sample <- igraph::neighborhood(hiv_net_ig, order = steps, nodes = seed_node, mode = "all")

# Extract the nodes in the snowball sample
key_individuals_snowball <- V(hiv_net_ig)[snowball_sample[[1]]]

# Print the key individuals
print(key_individuals_snowball)
```

```
## + 38/694 vertices, from 45c93ab:
##  [1]   1  18  33  84 121 585   3  15  19  25  28  43  98 127 131 151 234 241  10
## [20]  26  52  96 106 114  73  17  31  34  93 132 133 135 136 193 251 310 599 226
```

Burt's constraint

```r
# Calculate Burt's constraint for each node
burt_constraint <- igraph::constraint(hiv_net_ig, nodes = V(hiv_net_ig), weights = E(hiv_net_ig)$Weight)
V(hiv_net_ig)$constraint <- burt_constraint

# Create a data frame with node attributes and Burt's constraint values
node_attributes_and_constraint <- data.frame(node_id = as.vector(V(hiv_net_ig)),
                                            RACE = V(hiv_net_ig)$RACE,
                                            SEX = V(hiv_net_ig)$SEX,
                                            BEHAV = V(hiv_net_ig)$BEHAV,
                                            AGE = V(hiv_net_ig)$AGE,
                                            DISABLE = V(hiv_net_ig)$DISABLE,
                                            UNEMP = V(hiv_net_ig)$UNEMP,
                                            STREETS = V(hiv_net_ig)$STREETS,
                                            EDUC = V(hiv_net_ig)$EDUC,
                                            constraint = burt_constraint)



# Print the data frame with first 10 rows
print(node_attributes_and_constraint[1:10, ])
```

```
##    node_id RACE SEX BEHAV AGE DISABLE UNEMP STREETS EDUC constraint
## 1        1    4   1     2  24       0     1       0   12  0.3390928
## 2        2    4   1     2  24       0     1       0   12  1.0000000
## 3        3    4   0     0  72       0     0       0    5  0.2065079
## 4        4    4   1     0  30       0     1       0   12  0.5000000
## 5        5    2   0     0  58       0     0       0   12  0.4749138
## 6        6    4   0     2  43       0     0       0   13  0.1397772
## 7        7    4   1     0  26       0     1       0   14  0.5555556
## 8        8    4   1     0  28       0     1       0   12  0.5000000
## 9        9    4   0     0  37       0     0       0   12  0.1717598
## 10      10    4   1     2  33       0     1       0   12  0.2489251
```

```r
# Find the nodes with the lowest constraint values (top brokers)
top_brokers <- head(node_attributes_and_constraint[order(node_attributes_and_constraint$constraint), ],
```

```r
# Print the top brokers
print(top_brokers)
```

```
##     node_id RACE SEX BEHAV AGE DISABLE UNEMP STREETS EDUC constraint
## 52       52    4   0     0  32       0     0       0   12 0.06542113
## 43       43    4   1     2  30       0     1       0   12 0.07971723
## 133     133    2   1     0  26       0     1       0   12 0.08156873
## 135     135    4   0     0  20       0     1       0   11 0.08374918
## 346     346    4   1     0  34       0     0       0   12 0.08576506
## 136     136    2   1     2  22       0     1       0   10 0.11024187
## 98       98    2   0     0  39       0     1       0   14 0.11456026
## 527     527    2   0     0  54       0     1       0    7 0.11839886
## 457     457    2   0     0  42       1     1       1    7 0.11877319
## 80       80    2   0     0  26       0     0       0   12 0.11915254
```