Project 3: Unit 3

Suman Paudel 33

2024-05-01

Work 1

Use "airquality" data of R and locate median of "Temp" variable graphically

Hint:

1. Divide the "Temp" variable into different class intervals using a statistical rule and get number of frequencies in each class interval
2. Get less than frequency data for less than ogive
3. Get more than frequency data for more than ogive
4. Plot less than and more than ogives in a single plot
5. Intersection of less than and more than ogive in the x-axis is the median
6. Check this value with median code of R

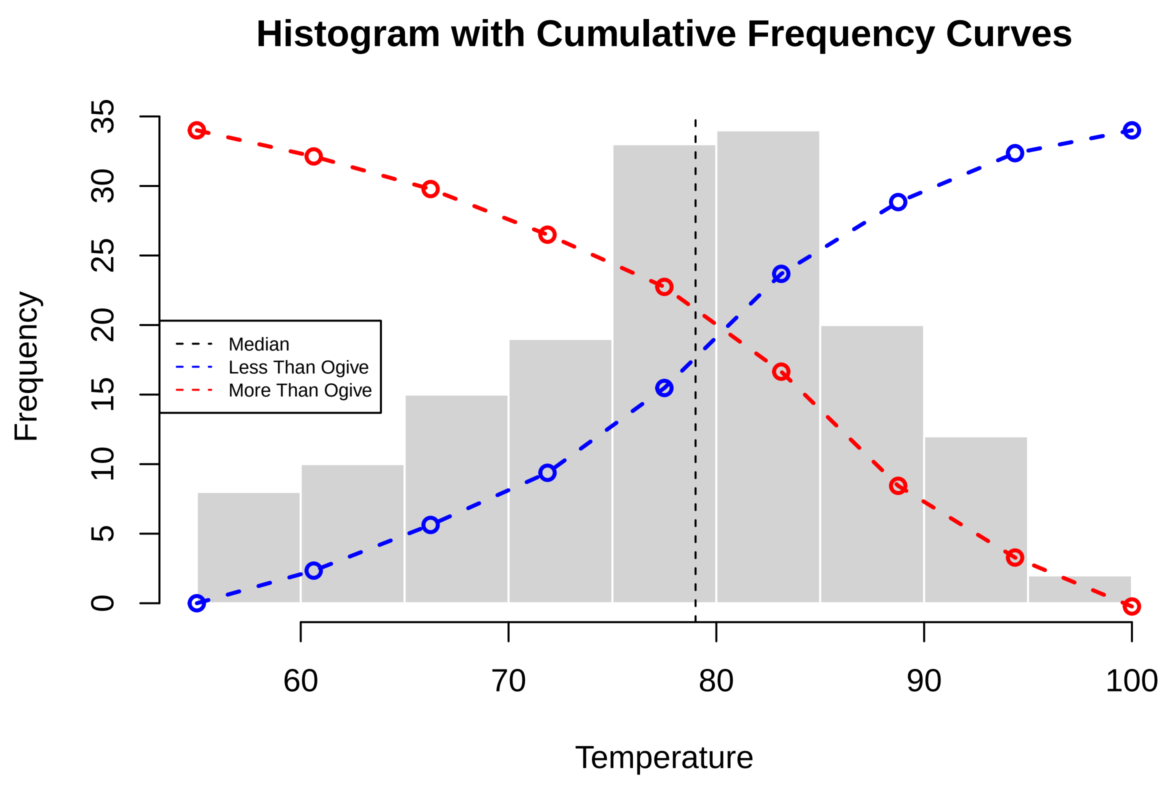
Solution Work 1

Load all of the necessary packages and data needed for Work 1.

```
# finding the class intervals using a statistical rule, such as Sturges' rule
intervals <- ceiling(log2(length(airquality$Temp)) + 1)

# creating class intervals using cut() function
temp_intervals <- cut(airquality$Temp, breaks = intervals)


# Get the frequencies in each class interval
freq <- table(temp_intervals)

# Calculate cumulative frequencies for each interval (ogives)
cum_freq_more_than <- cumsum(freq)
cum_freq_less_than <- rev(cumsum(rev(freq)))
cum_freq_more_than
```

```
##   (56,60.6] (60.6,65.1] (65.1,69.7] (69.7,74.2] (74.2,78.8] (78.8,83.3]
##           8          18          32          48          74         109
## (83.3,87.9] (87.9,92.4]   (92.4,97]
##         131         146         153
```

```r
# create the histogram with less than and more than gives
my_hist <- hist(airquality$Temp, breaks = intervals,
          main = "Histogram with Cumulative Frequency Curves",
          xlab = "Temperature", ylab = "Frequency", border = "white")
abline(v = median(airquality$Temp), col = 'black', lty = 2, xlim = c(50, 110))
par(new=TRUE)
with (my_hist, {plot(cum_freq_more_than, col = "blue", type = "o", lwd = 2, lty = 2, axes = FALSE,
               xlab = "", ylab = "")
  lines(cum_freq_less_than, type = "o", col = "red", lwd = 2, lty = 2)
})
# Add legend
legend("left", legend = c("Median", "Less Than Ogive", "More Than Ogive"),
       col = c("black", "blue", "red"), lty = c(2, 2, 2), cex=0.6)
```



Work 2

Work 2: Use "airquality" data of R and locate mode of "Temp" variable graphically

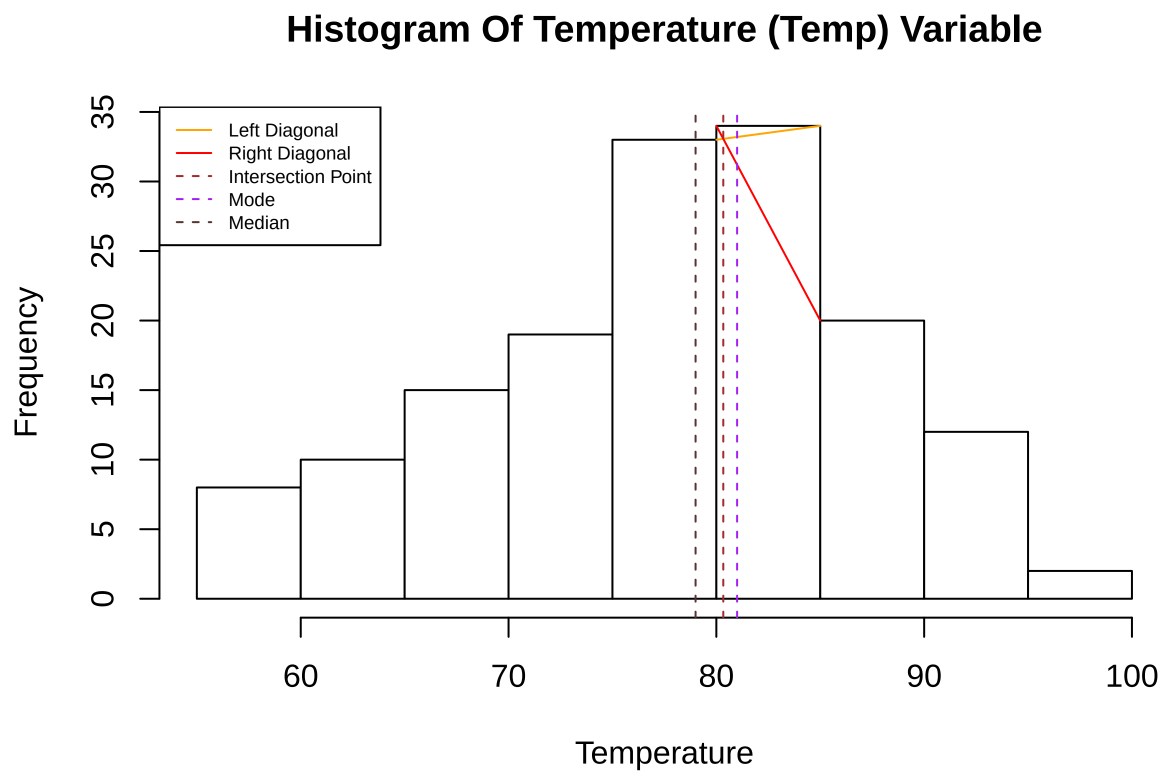
Hint:

7. Get histogram of "Temp" variable
8. Draw a diagonal line from en edge of the largest bar to the tip of the opposite adjacent bar
9. Draw another diagonal line from other edge of the largest bar to the tip of of the opposite adjacent bar

10. Intersection of the two diagonal lines in the x-axis in the mode
11. Check this value with mode code of R

```r
# Histogram of Temperature (Temp) variable
my_histogram <- hist(airquality$Temp,
                     main = "Histogram Of Temperature (Temp) Variable",
                     xlab = "Temperature",
                     ylab = "Frequency",
                     col = 'white')

# Find the location of the highest bar
max_bar_my_histogram <- which.max(my_histogram$counts)

# Get the leftmost and rightmost values of the highest bar
max__left_value_my_histogram <- my_histogram$breaks[max_bar_my_histogram]
max__right_value_my_histogram <- my_histogram$breaks[max_bar_my_histogram + 1]

# Get the highest count value
max_value_hist <- my_histogram$counts[max_bar_my_histogram]

# Get the locations of the left and right adjacent bars
loc_left_adj <- max_bar_my_histogram - 1
loc_right_adj <- max_bar_my_histogram + 1

# Get the rightmost frequency count value for the left adjacent bar and
# the leftmost frequency count value for the right adjacent bar
left_adj_bar_rightmost_count <- my_histogram$counts[loc_left_adj]
right_adj_bar_leftmost_count <- my_histogram$counts[loc_right_adj]

# Calculate the slopes for the left and right diagonals
slope_1 <- (max_value_hist - left_adj_bar_rightmost_count) /
            (max__right_value_my_histogram - max__left_value_my_histogram)
slope_2 <- (right_adj_bar_leftmost_count - max_value_hist) /
            (max__right_value_my_histogram - max__left_value_my_histogram)

# Calculate the equations of the left and right diagonals
eqn_1_line1 <- paste("y -",
                      left_adj_bar_rightmost_count,
                      "=", slope_1,
                      "(x -", max__left_value_my_histogram, ")",
                      sep="")
eqn_2_line2 <- paste("y -", max_value_hist,
                      "=", slope_2,
                      "(x -", max__left_value_my_histogram,
                      ")",
                      sep="")

# Calculate the constants for the left and right diagonals
constants_1 <- left_adj_bar_rightmost_count - slope_1 * max__left_value_my_histogram
constants_2 <- max_value_hist - slope_2 * max__left_value_my_histogram

# Calculate the intersection point
x_intersect <- (constants_2 - constants_1) / (slope_1 - slope_2)
y_intersect <- slope_1 * x_intersect + constants_1
```

```r
# Draw the diagonals and the intersection point
segments(max__right_value_my_histogram,
         max_value_hist, max__left_value_my_histogram,
         left_adj_bar_rightmost_count,
         col = "orange")

segments(max__left_value_my_histogram,
         max_value_hist,
         max__right_value_my_histogram,
         right_adj_bar_leftmost_count,
         col = "red")

abline(v = x_intersect, col = 'brown', lty = 2)

# Custom function to calculate mode
calculate_mode <- function(x) {
  unique_values <- unique(x)
  frequencies <- tabulate(match(x, unique_values))
  mode <- unique_values[which.max(frequencies)]
  return(mode)
}

# Calculate and draw the mode and median
mode <- calculate_mode(airquality$Temp)
abline(v = mode, col = 'purple', lty = 2)
median_temp <- median(airquality$Temp)
abline(v = median_temp, col = '#543533', lty = 2)

# Add legend
legend('topleft', legend = c("Left Diagonal", "Right Diagonal", "Intersection Point", "Mode", "Median")
       col = c("orange", "red", "brown", 'purple', "#543533"), lty =c(1,1,2,2,2), cex = 0.6)
```



Work 3

Work 3: Use "SNA__School.csv" data and perform social network analysis of first and second variables

Hint:

12. Import and create a data frame "s" with first and second column of the data
13. Save it as network graph data object "net" with directed = T argument
14. Check number of vertices, edges, degree of "net" and interpret the carefully
15. Get histogram of net degree and interpret it carefully
16. Get network diagram of "net" and interpret it carefully
17. Get network diagram of "net" with kamada.kawai layout and interpret it carefully
18. Get hubs using hubs score and interpret it carefully
19. Get authority using authority score and interpret it carefully
20. Get community using the a special network diagram parameter and interpret it carefully

```
# load the necessary library
library(igraph)
library(readr)

# load the data from the current work space in the object sna then
sna <- read.csv('sna.csv')
head(sna)
```

```
##   first second grade spec
## 1    AA     DD     6    Y
## 2    AB     DD     6    R
## 3    AF     BA     6    Q
## 4    DD     DA     6    Q
## 5    CD     EC     6    X
## 6    DD     CE     6    Y
```

```
# since we need first and column from the dataframe
sna <- sna[,1:2]
head(sna)
```

```
##   first second
## 1    AA     DD
## 2    AB     DD
## 3    AF     BA
## 4    DD     DA
## 5    CD     EC
## 6    DD     CE
```

```
# now saving network graph data object "net" with directed = T argument
net <- graph_from_data_frame(sna, directed = TRUE)
net
```

```
## IGRAPH 1d1006b DN-- 52 290 --
## + attr: name (v/c)
## + edges from 1d1006b (vertex names):
##  [1] AA->DD AB->DD AF->BA DD->DA CD->EC DD->CE CD->FA CD->CC BA->AF CB->CA
## [11] CC->CA CD->CA BC->CA DD->DA ED->AD AE->AC AB->BA CD->EC CA->CC EB->CC
## [21] BF->CE BB->CD AC->AE CC->FB DC->BB BD->CF DB->DA DD->DA DB->DD BC->AF
## [31] CF->DE DF->BF CB->CA BE->CA EA->CA CB->CA CB->CA CC->CA CD->CA BC->CA
## [41] BF->CA CE->CA AC->AD BD->BE AE->DF CB->DF AC->DF AA->DD AA->DD AA->DD
## [51] CD->DD AA->DD EE->DD CD->DD DB->AA AA->FC BE->CC EF->FD CF->FE BB->DD
## [61] CD->DD BA->AB CD->EC BE->EE CE->CC CD->CC ED->CC BB->CC BE->CE DD->CE
## [71] AC->CD ED->CD FF->CD AC->CD DD->CD DD->CD AE->GA AE->GA AE->GA AE->GA
## + ... omitted several edges
```

```
# check number of the vertices, edges. degree of the 'net' and interpret the carefully

# number of vertices
vcount(net)
```

```
## [1] 52
```

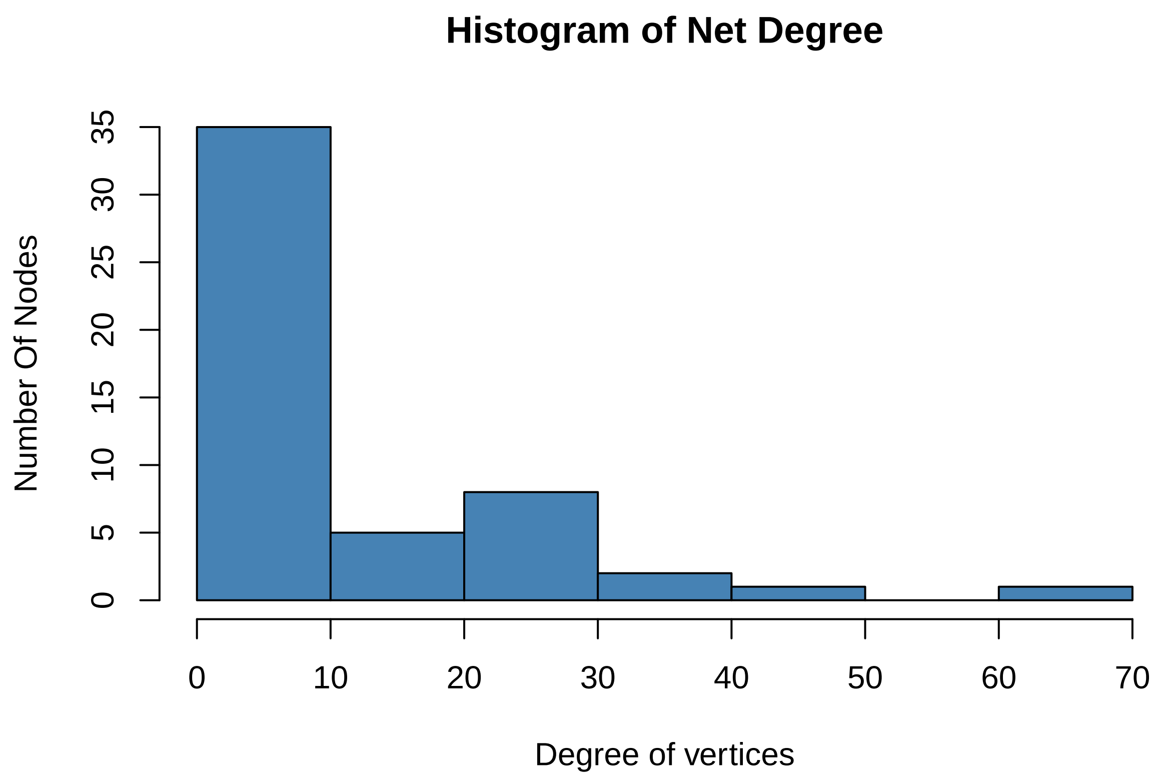
```
# number of edges
ecount(net)
```

```
## [1] 290
```

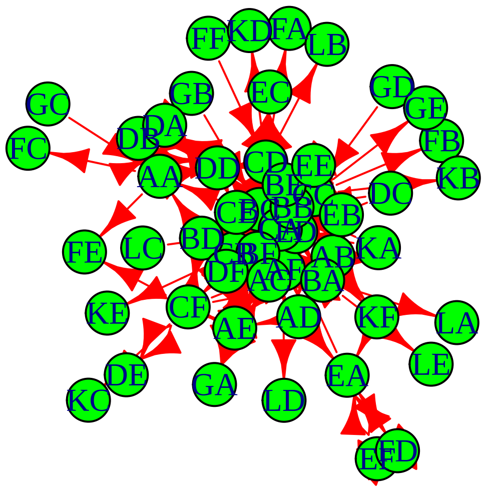
```
## degree
degree(net)
```

```
## AA AB AF DD CD BA CB CC BC ED AE CA EB BF BB AC DC BD DB CF DF BE EA CE EE EF
## 18  9 23 36 40 26 24 50 21 27 15 62  7 12 23 27  2  4  8 12 23 20  8 10  6  8
## FF FD GB GC GD AD KA KF LC DA EC FA FB DE FC FE GA GE KB KC KD KE LB LA LD LE
##  1  8  1  1  1  9  3  3  1  7  3  1  1  2  1  2  5  1  1  1  1  1  1  1  1  1
```

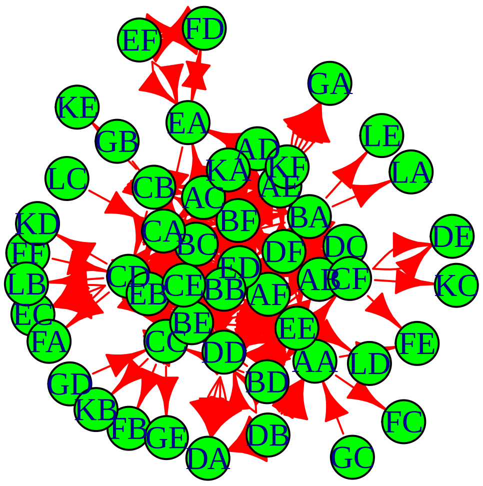
```
# create histogram of net degree and interpret it carefully
hist(degree(net), main = "Histogram of Net Degree",
     xlab = "Degree of vertices", ylab = "Number Of Nodes",
     col = 'steelblue')
```



```
# Interpretation: The histogram shows the distribution of vertex degrees in the network.
# Most nodes have a degree between 0 and 10, indicating that they are connected to a
# small number of other nodes. There are fewer nodes with higher degrees,
# indicating that there are some nodes that serve as hubs
# or connectors within the network.
# Get network diagram of "net" and interpret it carefully


plot(net, edge.size = 15, edge.color = 'red',
     vertex.size = 20, vertex.color = 'green', )
```



```
# Interpretation: The network diagram shows the connections between nodes in the network.
# Nodes are represented as circles (vertices), and edges between them represent connections.
# The size and color of vertices and edges can be adjusted for better visualization.
# Here, edges are in red, and vertices are in green.


# Get network diagram of "net" with kamada.kawai layout and interpret it carefully
plot(net, edge.size = 10, edge.color = 'red',
     vertex.size = 20, vertex.color = 'green',
     layout = layout.kamada.kawai)
```



```
# Interpretation: The Kamada-Kawai layout algorithm positions the nodes in the graph based on the
# strength of their connections. Nodes that are more closely connected are placed closer together,
# while nodes with weaker connections are positioned farther apart.
# This layout can help reveal the overall structure and clustering of the network.
```

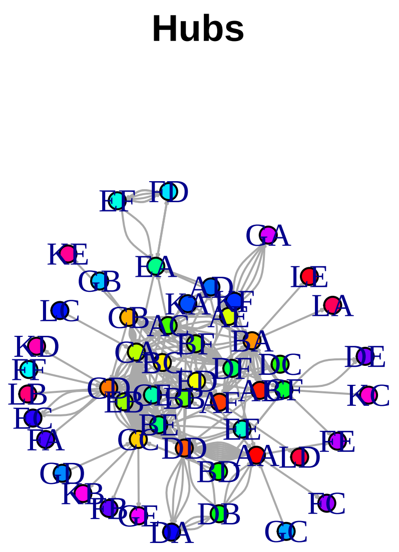
```
# Get hubs using hubs score and interpret it carefully
# Get hubs using hubs score and interpret it carefully
hubs <- hub_score(net)$vector
hubs
```

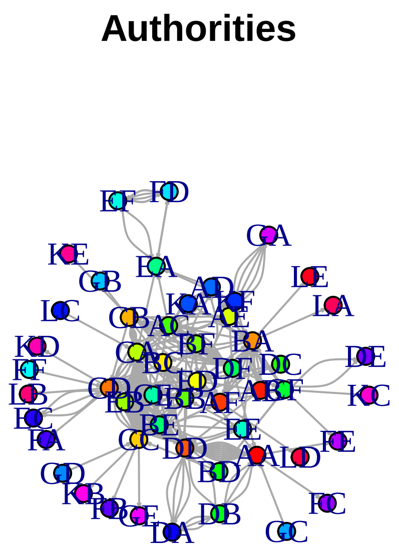
```
##           AA           AB           AF           DD           CD           BA
## 8.042807e-02 1.300950e-02 4.731440e-03 1.255496e-02 2.350165e-01 3.239013e-02
##           CB           CC           BC           ED           AE           CA
## 7.145888e-01 1.000000e+00 2.321828e-01 4.858357e-02 7.949708e-02 4.494956e-02
##           EB           BF           BB           AC           DC           BD
## 1.870058e-01 1.272314e-01 1.150808e-01 2.093518e-01 7.455515e-03 1.305960e-02
##           DB           CF           DF           BE           EA           CE
## 1.738176e-02 7.102816e-03 2.505787e-02 2.195271e-01 5.693409e-02 6.319085e-02
##           EE           EF           FF           FD           GB           GC
## 1.273824e-02 1.119098e-03 4.485458e-03 4.563724e-04 3.594258e-04 2.507046e-04
##           GD           AD           KA           KF           LC           DA
## 6.374463e-03 9.643170e-03 6.276463e-02 4.418478e-04 5.681639e-02 3.734849e-17
##           EC           FA           FB           DE           FC           FE
## 3.267993e-17 7.002841e-18 7.002841e-18 1.400568e-17 7.002841e-18 1.400568e-17
##           GA           GE           KB           KC           KD           KE
## 4.668561e-17 7.002841e-18 7.002841e-18 7.002841e-18 7.002841e-18 7.002841e-18
##           LB           LA           LD           LE
## 7.002841e-18 7.002841e-18 7.002841e-18 7.002841e-18
```

```
# Interpretation: The hubs scores represent the centrality of each node in the network,
# indicating how important or central each node is in terms of connecting to other nodes.
# Higher hub scores suggest nodes that act as hubs or connectors within the network.


# Get authority using authority score and interpret it carefully
authority <- authority_score(net)$vector
authority
```

```
##           AA           AB           AF           DD           CD           BA
## 4.412541e-03 3.968841e-03 8.375004e-02 1.409698e-01 7.894656e-02 2.146204e-02
##           CB           CC           BC           ED           AE           CA
## 3.660444e-03 1.121941e-01 6.326094e-03 4.799095e-02 1.524499e-02 1.000000e+00
##           EB           BF           BB           AC           DC           BD
## 0.000000e+00 6.072030e-03 8.323026e-02 6.262068e-02 0.000000e+00 2.486679e-04
##           DB           CF           DF           BE           EA           CE
## 0.000000e+00 8.146610e-03 1.925650e-01 1.095155e-01 7.776767e-03 2.437300e-02
##           EE           EF           FF           FD           GB           GC
## 2.419075e-02 6.390996e-05 0.000000e+00 2.071608e-03 0.000000e+00 0.000000e+00
##           GD           AD           KA           KF           LC           DA
## 0.000000e+00 1.806051e-02 0.000000e+00 2.156359e-03 0.000000e+00 3.583779e-03
##           EC           FA           FB           DE           FC           FE
## 2.468362e-02 8.227873e-03 3.500976e-02 4.973358e-04 2.815768e-03 3.064436e-03
##           GA           GE           KB           KC           KD           KE
## 1.391587e-02 3.500976e-02 3.500976e-02 2.486679e-04 8.227873e-03 2.501759e-02
##           LB           LA           LD           LE
## 8.227873e-03 1.133971e-03 1.656466e-04 1.133971e-03
```

```
# lets plot 2 network diagram side by side
# with 1 row and 2 column
#
# PLOT FOR HUBS
par(mfrow = c(1, 2))
set.seed(123)
plot(
  net,
  vertex.size = 10,
  # vary vertex size based on hubscore
  main = "Hubs",
  vertex.color = rainbow(52),
  edge.arrow.size = 0.1,
  layout = layout.kamada.kawai
)
# Vertex CC has highest hub score as the vertex size is bigger
# It means Vertex `CC` has maximum outgoing links
# Second biggest is CB
# PLOT FOR AUTHORITIES
plot(
  net,
  vertex.size = 10,
  main = "Authorities",
  vertex.color = rainbow(52),
  edge.arrow.size = 0.1,
  layout = layout.kamada.kawai
)
```

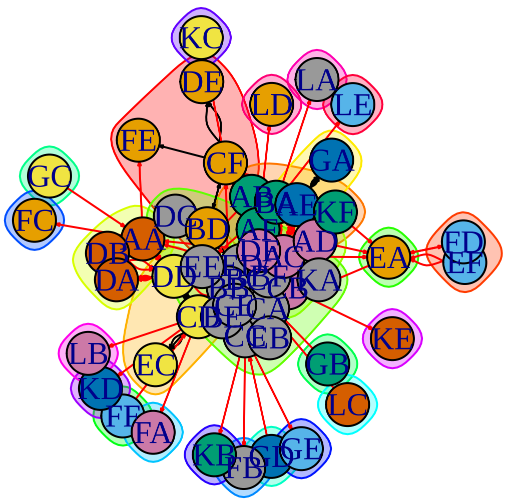




```
# Interpretation: The authority scores represent the level of expertise or
# influence of each node in the network. # Nodes with higher authority scores are
# considered more authoritative or influential within the network.
# Get community using the a special network diagram parameter and interpret it carefully

# Perform community detection using Louvain method
communities <- cluster_walktrap(net)
communities
```

```
## IGRAPH clustering walktrap, groups: 26, mod: 0.3
## + groups:
##   $`1`
##   [1] "BD" "CF" "DE" "FE"
##
##   $`2`
##   [1] "EF" "FD"
##
```

```
##   $`3`
##   [1] "AB" "AF" "BA" "KF"
##
##   $`4`
##   + ... omitted several groups/vertices

# Get the membership vector indicating which community each node belongs to
membership <- membership(communities)
membership

## AA AB AF DD CD BA CB CC BC ED AE CA EB BF BB AC DC BD DB CF DF BE EA CE EE EF
##  6  3  3  4  4  3  7  8  8  8  5  8  8  8  8  7  8  1  6  1  7  8  9  8  8  2
## FF FD GB GC GD AD KA KF LC DA EC FA FB DE FC FE GA GE KB KC KD KE LB LA LD LE
## 10  2 11 12 13  7  8  3 14  6  4 15 16  1 17  1  5 18 19 20 21 22 23 24 25 26

# Plot the network with communities highlighted
par(mfrow=c(1,1))
plot(communities,
     net,
     edge.size = 10,
     dge.color = 'red',
     vertex.size = 20,
     vertex.color = 'green',
     edge.arrow.size = 0.1)
```



```
# Interpretation
# The plot will display the network with nodes colored according to their community membership.
# Nodes belonging to the same community will have the same color, helping to visually
# identify distinct communities within the network.
# You can also get information about the communities themselves
# Number of communities
num_communities <- length(communities)
num_communities
```

```
## [1] 26
```

```
# Sizes of communities
community_sizes <- sizes(communities)
community_sizes
```

```
## Community sizes
##  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26
##  4  2  4  3  2  3  4 12  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1
```

```
# Average modularity of the partition
modularity <- modularity(communities)
modularity
```

```
## [1] 0.2987931
```

```
# Interpretation
# The number of communities gives you an idea of how fragmented or cohesive the network is.
# Larger networks may naturally have more communities.
# Community sizes indicate the distribution of nodes among the communities.
# You might have some dominant communities and some smaller ones.
# Modularity measures how well the network is divided into communities.
# Higher modularity values indicate a better division, with nodes more densely
# connected within communities and sparsely connected between communities.
```