Data Visualization Lab 7

Aryan Vigyat 20BCE1452

2023-02-09

Loading the Libraries Neeeded

```
library(igraph)
```

```
##
## Attaching package: 'igraph'
```

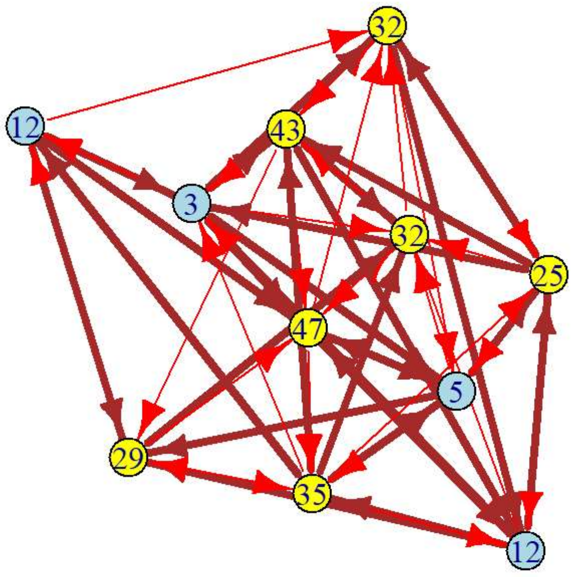
```
## The following objects are masked from 'package:stats':
##
##     decompose, spectrum
```

```
## The following object is masked from 'package:base':
##
##     union
```

1. Loading the Graoh with Adjacency Matrix

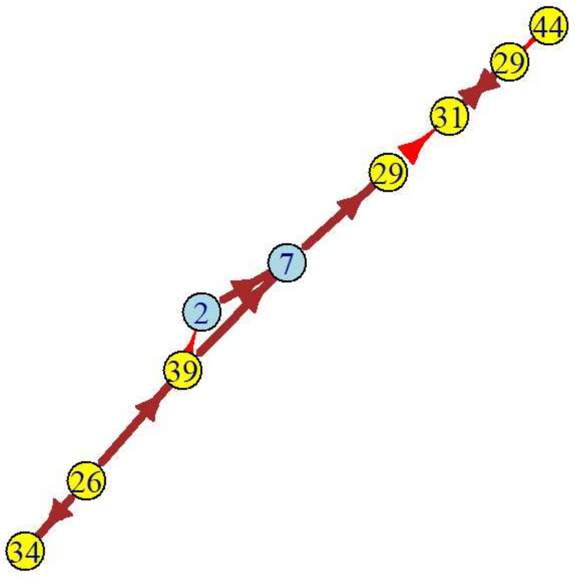
```
dat=read.csv(file.choose(),header=TRUE,sep=',', row.names=1,check.names=FALSE)
m=as.matrix(dat)
net=graph.adjacency(m,mode="directed",weighted=TRUE,diag=FALSE)
V(net)$color <- "yellow"
V(net)$number <- sample(1:50, vcount(net), replace=TRUE)
V(net)[ number < 20 ]$color <- "lightblue"

E(net)$weight <- runif(ecount(net))
E(net)$width <- 1
E(net)$color <- "red"
E(net)[ weight < 0.5 ]$width <- 4
E(net)[ weight < 0.5 ]$color <- "brown"
plot(net, layout = layout.fruchterman.reingold,vertex.label=V(net)$number)
```



2. Loading the Graph with EdgeList

```
e1=read.table(file.choose(),sep=",",header=T)
g2 <- graph.data.frame(e1)
V(g2)$number=sample(1:50, vcount(g2), replace=TRUE)
V(g2)$color="yellow"
V(g2)[ number < 20 ]$color <- "lightblue"
E(g2)$weight <- runif(ecount(g2))
E(g2)$width <- 1
E(g2)$color <- "red"
E(g2)[ weight < 0.5 ]$width <- 4
E(g2)[ weight < 0.5 ]$color <- "brown"
plot(g2, layout = layout.fruchterman.reingold,vertex.label=V(g2)$number)
```



3. Display the edges & vertices, the network as matrix and the names of vertices

```
#For Graph 1
get.edgelist(net)
```

```
##       [,1]   [,2]  
##  [1,] "2000" "2001"
##  [2,] "2000" "2003"
##  [3,] "2000" "2005"
##  [4,] "2000" "2007"
##  [5,] "2001" "2000"
##  [6,] "2001" "2002"
##  [7,] "2001" "2004"
##  [8,] "2001" "2006"
##  [9,] "2001" "2007"
## [10,] "2001" "2009"
## [11,] "2001" "2010"
## [12,] "2002" "2000"
## [13,] "2002" "2001"
## [14,] "2002" "2004"
## [15,] "2003" "2000"
## [16,] "2003" "2001"
## [17,] "2003" "2002"
## [18,] "2003" "2005"
## [19,] "2003" "2007"
## [20,] "2003" "2009"
## [21,] "2004" "2000"
## [22,] "2004" "2002"
## [23,] "2004" "2005"
## [24,] "2004" "2007"
## [25,] "2004" "2009"
## [26,] "2004" "2010"
## [27,] "2005" "2002"
## [28,] "2005" "2003"
## [29,] "2005" "2004"
## [30,] "2005" "2006"
## [31,] "2005" "2010"
## [32,] "2006" "2002"
## [33,] "2006" "2004"
## [34,] "2006" "2008"
## [35,] "2006" "2010"
## [36,] "2007" "2002"
## [37,] "2007" "2003"
## [38,] "2007" "2004"
## [39,] "2007" "2006"
## [40,] "2007" "2010"
## [41,] "2008" "2000"
## [42,] "2008" "2004"
## [43,] "2008" "2006"
## [44,] "2008" "2009"
## [45,] "2009" "2000"
## [46,] "2009" "2001"
## [47,] "2009" "2002"
## [48,] "2009" "2004"
## [49,] "2009" "2005"
## [50,] "2009" "2007"
## [51,] "2009" "2008"
## [52,] "2010" "2001"
## [53,] "2010" "2002"
## [54,] "2010" "2003"
## [55,] "2010" "2005"
## [56,] "2010" "2006"
## [57,] "2010" "2007"
## [58,] "2010" "2008"
## [59,] "2010" "2009"
```

```
get.adjacency(net)
```

```
## 11 x 11 sparse Matrix of class "dgCMatrix"
```

```
##   [[ suppressing 11 column names '2000', '2001', '2002' ... ]]
```

```
##
## 2000 . 1 . 1 . 1 . 1 . . .
## 2001 1 . 1 . 1 . 1 1 . 1 1
## 2002 1 1 . . 1 . . . . . .
## 2003 1 1 1 . . 1 . 1 . 1 .
## 2004 1 . 1 . . 1 . 1 . 1 1
## 2005 . . 1 1 1 . 1 . . . 1
## 2006 . . 1 . 1 . . . 1 . 1
## 2007 . . 1 1 1 . 1 . . . 1
## 2008 1 . . . 1 . 1 . . 1 .
## 2009 1 1 1 . 1 1 . 1 1 . .
## 2010 . 1 1 1 . 1 1 1 1 1 .
```

```
#For Graph 2
get.edgelist(g2)
```

```
##       [,1]   [,2]
##  [1,] "2000" "2001"
##  [2,] "2001" "2002"
##  [3,] "2002" "2004"
##  [4,] "2003" "2004"
##  [5,] "2004" "2003"
##  [6,] "2004" "2002"
##  [7,] "2006" "2008"
##  [8,] "2006" "2007"
##  [9,] "2008" "2010"
## [10,] "2008" "2000"
## [11,] "2010" "2000"
```

```
get.adjacency(g2)
```

```
## 9 x 9 sparse Matrix of class "dgCMatrix"
##      2000 2001 2002 2003 2004 2006 2008 2010 2007
## 2000    .    1    .    .    .    .    .    .    .
## 2001    .    .    1    .    .    .    .    .    .
## 2002    .    .    .    .    1    .    .    .    .
## 2003    .    .    .    .    1    .    .    .    .
## 2004    .    .    1    1    .    .    .    .    .
## 2006    .    .    .    .    .    .    1    .    1
## 2008    1    .    .    .    .    .    .    1    .
## 2010    1    .    .    .    .    .    .    .    .
## 2007    .    .    .    .    .    .    .    .    .
```

4. Find the count of vertices and edges of the created graph

```
#For Graph 1
print(vcount(net))
```

```
## [1] 11
```

```
print(ecount(net))
```

```
## [1] 59
```

```
#For Graph 2
print(vcount(g2))
```

```
## [1] 9
```

```
print(ecount(g2))
```

```
## [1] 11
```

5. Display the adjacency vertices of each vertex(individual) in the created graph

```
#For Graph 1
for (i in 1:vcount(net)) {
  # get the adjacent vertices of each vertex
  adj_vertices <- neighbors(net, i)

  # print the vertex and its adjacent vertices
  cat("Vertex", i, "has the following neighbouring vertices:", adj_vertices, "\n")
}
```

```
## Vertex 1 has the following neighbouring vertices: 2 4 6 8
## Vertex 2 has the following neighbouring vertices: 1 3 5 7 8 10 11
## Vertex 3 has the following neighbouring vertices: 1 2 5
## Vertex 4 has the following neighbouring vertices: 1 2 3 6 8 10
## Vertex 5 has the following neighbouring vertices: 1 3 6 8 10 11
## Vertex 6 has the following neighbouring vertices: 3 4 5 7 11
## Vertex 7 has the following neighbouring vertices: 3 5 9 11
## Vertex 8 has the following neighbouring vertices: 3 4 5 7 11
## Vertex 9 has the following neighbouring vertices: 1 5 7 10
## Vertex 10 has the following neighbouring vertices: 1 2 3 5 6 8 9
## Vertex 11 has the following neighbouring vertices: 2 3 4 6 7 8 9 10
```

```
#For Graph 2
for (i in 1:vcount(g2)) {
  # get the adjacent vertices of each vertex
  adj_vertices <- neighbors(g2, i)

  # print the vertex and its adjacent vertices
  cat("Vertex", i, "has the following neighbouring vertices:", adj_vertices, "\n")
}
```

```
## Vertex 1 has the following neighbouring vertices: 2
## Vertex 2 has the following neighbouring vertices: 3
## Vertex 3 has the following neighbouring vertices: 5
## Vertex 4 has the following neighbouring vertices: 5
## Vertex 5 has the following neighbouring vertices: 3 4
## Vertex 6 has the following neighbouring vertices: 7 9
## Vertex 7 has the following neighbouring vertices: 1 8
## Vertex 8 has the following neighbouring vertices: 1
## Vertex 9 has the following neighbouring vertices:
```

6.Find the min and max degree of the created graph

```
#For Graph 1
vertex_degrees <- degree(net)

# find the minimum and maximum degree
min_degree <- min(vertex_degrees)
max_degree <- max(vertex_degrees)

# print the minimum and maximum degree
cat("The minimum degree is", min_degree, "\n")
```

```
## The minimum degree is 7
```

```
cat("The maximum degree is", max_degree, "\n")
```

```
## The maximum degree is 13
```

```
#For Graph 2
vertex_degrees <- degree(g2)

# find the minimum and maximum degree
min_degree <- min(vertex_degrees)
max_degree <- max(vertex_degrees)

# print the minimum and maximum degree
cat("The minimum degree is", min_degree, "\n")
```

```
## The minimum degree is 1
```

```
cat("The maximum degree is", max_degree, "\n")
```

```
## The maximum degree is 4
```

7. Create & set vertex attribute property named profit and values(“+”, “-”, “+”, “-”, “+”, “-”, “+”, “-”, “+”)

```
vertex_attributes <- c("+", "-", "+", "-", "+", "-", "+", "-", "+")

# set the vertex attributes as the "profit" property
set_vertex_attr(g2, "profit", value = vertex_attributes)
```

```
## IGRAPH 167d1eb DNW- 9 11 --
## + attr: name (v/c), number (v/n), color (v/c), profit (v/c), weight
## | (e/n), width (e/n), color (e/c)
## + edges from 167d1eb (vertex names):
##  [1] 2000->2001 2001->2002 2002->2004 2003->2004 2004->2003 2004->2002
##  [7] 2006->2008 2006->2007 2008->2010 2008->2000 2010->2000
```

```
# check the vertex attributes
summary(g2)
```

```
## IGRAPH 167d1eb DNW- 9 11 --
## + attr: name (v/c), number (v/n), color (v/c), weight (e/n), width
## | (e/n), color (e/c)
```

8. Create & set vertex attribute property named type and values(either leap or non-leap year)

```
V(g2)$name <- as.numeric(V(g2)$name)
V(g2)$type = ifelse(V(g2)$name %% 4 == 0, "Leap", "Non-Leap")
V(g2)$type
```

```
## [1] "Leap"     "Non-Leap" "Non-Leap" "Non-Leap" "Leap"     "Non-Leap" "Leap"
## [8] "Non-Leap" "Non-Leap"
```

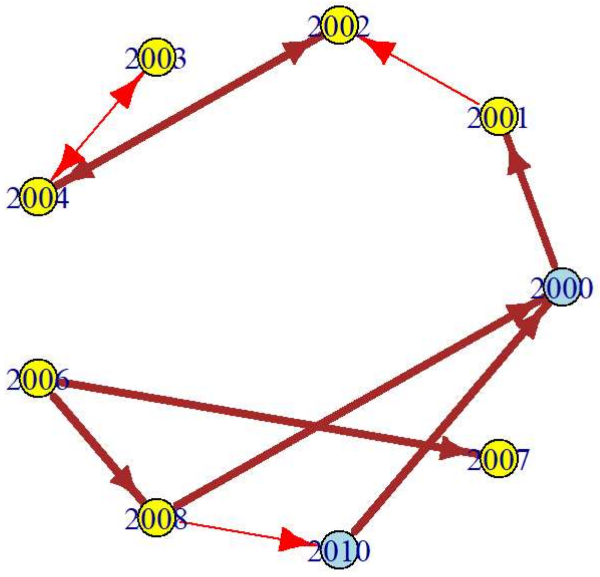
9.Create & set edge attribute named weight and values (if edge exits in between leap year vertices then 5 else 1)

```
E(g2)$weight <- as.numeric(ifelse(all(V(g2)$type == "Leap"), 5, 1))
print(E(g2)$weight)
```

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

10. Convert the created un-directed graph into directed graph based on the following rule a. edge directed towards high value vertex

```
dg = as.directed(g2, mode = "arbitrary")
plot(dg,layout=layout.circle)
```



12. Display the adjacency matrix of the resultant directed graph

```
E(dg, P=NULL, path=NULL, directed=TRUE)
```

```
## + 11/11 edges from 1741560 (vertex names):
##  [1] 2000->2001 2001->2002 2002->2004 2003->2004 2004->2003 2004->2002
##  [7] 2006->2008 2006->2007 2008->2010 2008->2000 2010->2000
```

13. Display the in-degree and out-degree of each vertex of resultant directed graph

```
indegree <- degree(dg, mode = "in")
outdegree <- degree(dg, mode = "out")

# Combine the indegree and outdegree into a data frame
df <- data.frame(vertex = V(dg)$name, indegree = indegree, outdegree = outdegree)

# Display the result
print(df)
```

```
##      vertex indegree outdegree
## 2000   2000        2         1
## 2001   2001        1         1
## 2002   2002        2         1
## 2003   2003        1         1
## 2004   2004        2         2
## 2006   2006        0         2
## 2008   2008        1         2
## 2010   2010        1         1
## 2007   2007        1         0
```