Faculty Development Programme on

Network Science: Foundation of Social Network Analysis

Introduction to R and Network Science

Hands-on Session (Day 1)

Swagata Duari
Kirti Jain

PART 1: Introduction and R Basics

Introduction and Preliminaries

- R is a free software environment for statistical computing and graphics.
- Rich set of packages to accomplish a wide spectrum of analytics tasks.
- Vibrant and active community.

Installation of Softwares and Packages

- R: https://cran.r-project.org/
- RStudio: https://www.rstudio.com/products/rstudio/download/
- Packages: **igraph, brainGraph**
    - *install.packages(<package_name>)*
    - E.g: *install.packages(igraph)*

Data Types

1. Vectors and sequences
2. Factors
3. Matrices and Arrays
4. Data Frames
5. Lists

Basic Data Visualization

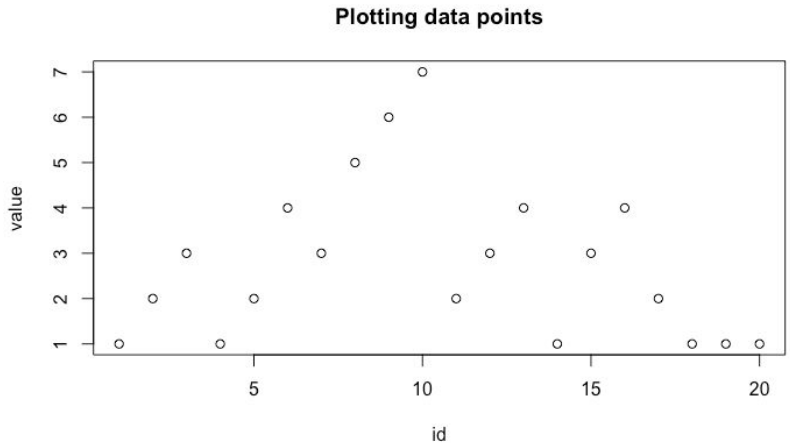
`plot` Command:

```
> v = c(1,2,3,1,2,4,3,5,6,7,2,3,4,1,3,4,2,1,1,1)
> df = data.frame( id = c(1:20), value = v)
> plot(df)        # plots only circled data points
```

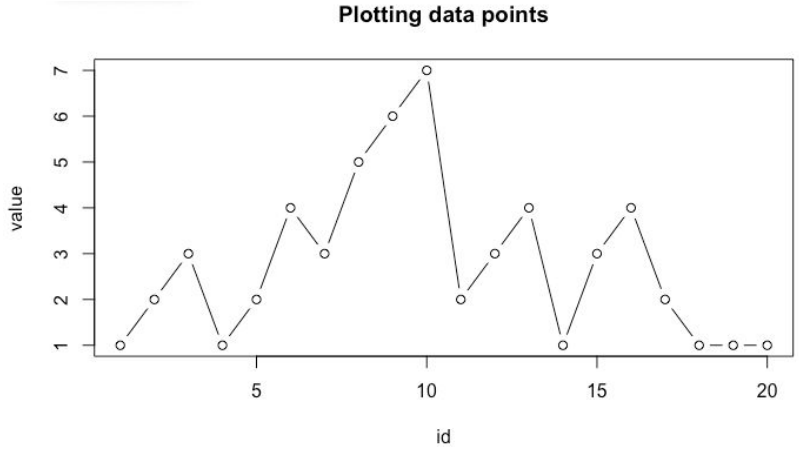
```
> plot(df, type = "b")  # plots a line with circled points
```

```
> ?plot                 # help on plot function
```

`plot` Output:



`plot` Output, `type = "b"`:



Other useful functions

```
> object                    # prints the object
```

```
> length (object)
# number of elements or components
```

```
> class (object)            # class or type of an object
```

```
> str (object)              # structure of an object
```

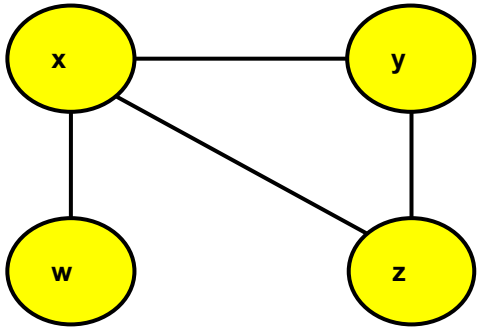
```
> names (object)            # set names of objects
```

```
> ls ()                     # list current objects
```

```
> rm (object)               # delete an object
```

PART 2: Introduction to Network Science Using R

Representation of *Graph*



wx
xy
xz
yz

Edge List

	w	x	y	z
w	0	1	0	0
x	1	0	1	1
y	0	1	0	1
z	0	1	1	0

Adjacency Matrix

The elements of the matrix indicate whether pairs of vertices are adjacent or not in the graph.

w - {x}
x - {w, y, z}
y - {x, z}
z - {x, y}

Adjacency List

Each list describes the set of neighbors of a vertex in the graph.

Create a graph from an edge list as matrix

Undirected graph

```
> el <- matrix ( c ( "V1" , "V2",
                     "V1" , "V3",
                     "V1" , "V4",
                     "V1" , "V5",
                     "V2" , "V5",
                     "V3" , "V4",
                     "V4" , "V5",
                     "V4" , "V7",
                     "V5" , "V8",
                     "V6" , "V2",
                     "V7" , "V8" ), nc = 2, byrow = TRUE )
> el
```

Data(vector)

#cols

Fill matrix by rows

OUTPUT :

```
     [,1] [,2]
[1,] "V1" "V2"
[2,] "V1" "V3"
[3,] "V1" "V4"
[4,] "V1" "V5"
[5,] "V2" "V5"
[6,] "V3" "V4"
[7,] "V4" "V5"
[8,] "V4" "V7"
[9,] "V5" "V8"
[10,] "V6" "V2"
[11,] "V7" "V8"
```

Edge list or 2-column Matrix

```
> G_el <- Graph_from_edgelist ( el, directed = FALSE )
```

OUTPUT :

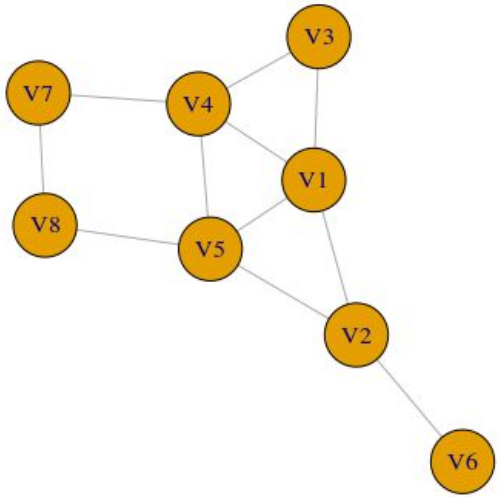
```
IGRAPH  UN-- 8 11 --
+ attr: name (v/c)
+ edges (vertex names):
 [1] V1--V2 V1--V3 V1--V4 V1--V5 V2--V5 V3--V4 V4--V5 V4--V7 V5--V8 V2--V6 V7--V8
```

Igraph graph, Undirected Named graph, 8=#vertices, 11=#edges

Attribute: named vertex/character

```
> plot ( G_el )
```

OUTPUT :



Create a graph from an adjacency matrix

```
> am <- matrix (c(  0, 1, 1, 1, 1, 0, 0, 0,
                    1, 0, 0, 0, 1, 1, 0, 0,
                    1, 0, 0, 1, 0, 0, 0, 0,
                    1, 0, 1, 0, 1, 0, 1, 0,
                    1, 1, 0, 1, 0, 0, 0, 1,
                    0, 1, 0, 0, 0, 0, 0, 0,
                    0, 0, 0, 1, 0, 0, 0, 1,
                    0, 0, 0, 0, 1, 0, 1, 0 ),
                nrow=8,
                ncol=8,
                byrow = TRUE)
> am
```

Data Elements

#rows

#columns

Fill matrix by rows

OUTPUT :

	[,1]	[,2]	[,3]	[,4]	[,5]	[,6]	[,7]	[,8]
[1,]	0	1	1	1	1	0	0	0
[2,]	1	0	0	0	1	1	0	0
[3,]	1	0	0	1	0	0	0	0
[4,]	1	0	1	0	1	0	1	0
[5,]	1	1	0	1	0	0	0	1
[6,]	0	1	0	0	0	0	0	0
[7,]	0	0	0	1	0	0	0	1
[8,]	0	0	0	0	1	0	1	0

Undirected graph

```
> G_am <- graph_from_adjacency_matrix (adjm, mode = c ("undirected") )
```

OUTPUT :

IGRAPH U--- 8 11 --

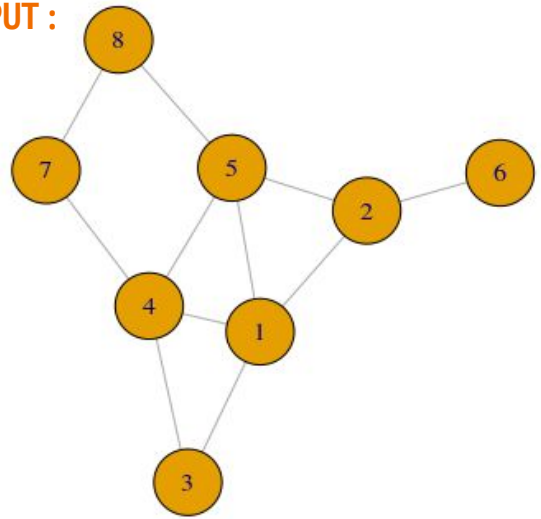
Igraph graph, Undirected graph, 8=#vertices, 11=#edges

+ edges :

[1] 1--2 1--3 1--4 1--5 2--5 2--6 3--4 4--5 4--7 5--8 7--8

```
> plot ( G_am )
```

OUTPUT :



Create a graph from an adjacency list

Make a list of adjacent vertices

```
> a_list <- list ( c(2,3,4,5), c(1,5,6),
                   c(4,1), c(1,3,5,7),
                   c(1,2,4,8), c(2),
                   c(4,8), c(5,7) )
> a_list
```

OUTPUT :

```
[[1]]
[1] 2 3 4 5

[[2]]
[1] 1 5 6

[[3]]
[1] 4 1

[[4]]
[1] 1 3 5 7

[[5]]
[1] 1 2 4 8

[[6]]
[1] 2

[[7]]
[1] 4 8

[[8]]
[1] 5 7
```

```
> names(a_list) <- c(1,2,3,4,5,6,7,8)
> a_list
```

Set the names of list

OUTPUT :

```
$`1`
[1] 2 3 4 5

$`2`
[1] 1 5 6

$`3`
[1] 4 1

$`4`
[1] 1 3 5 7

$`5`
[1] 1 2 4 8

$`6`
[1] 2

$`7`
[1] 4 8

$`8`
[1] 5 7
```

Undirected graph

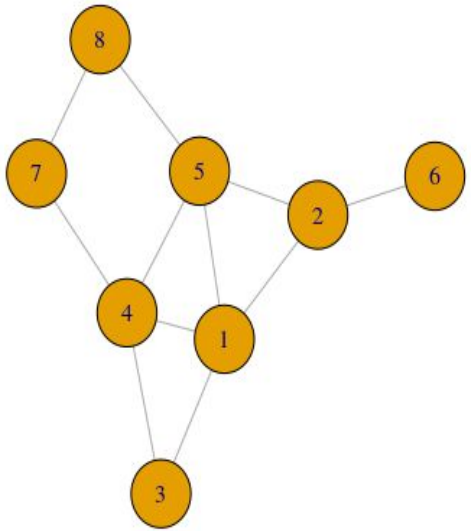
```
> G_a_list <- graph_from_adj_list (a_list, mode = "all")
```

OUTPUT :

```
IGRAPH  U--- 8 11 --
+ edges :
 [1] 1--2 1--3 1--4 1--5 2--5 2--6 3--4 4--5 4--7 5--8 7--8
```

Igraph graph, Undirected graph, 8=#vertices, 11=#edges

```
> plot ( G_a_list )
```



Create Network using "graph" function

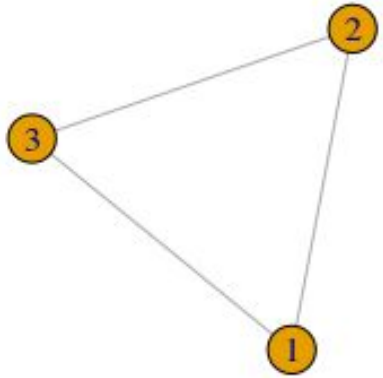
Create a graph from list of edges

#vertices

undirected

```
> g <- graph( edges=c(1,2, 2,3, 3, 1), n =3, directed=F )
> plot (g)
```

OUTPUT :




```
> g
```

OUTPUT :

```
IGRAPH  U--- 3 3 --
+ edges :
[1] 1--2 2--3 1--3
```

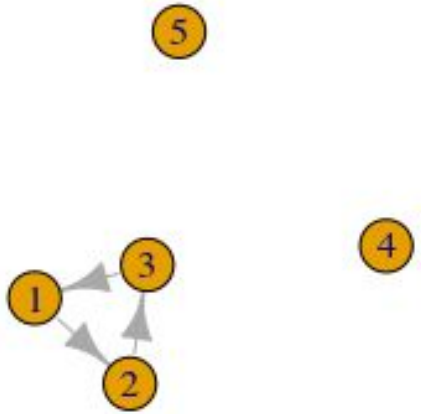
Igraph graph, Undirected graph, 3=#vertices, 3=#edges

#vertices

Directed by default

```
> g <- graph( edges=c(1,2, 2,3, 3, 1), n =5 )
> plot (g)
```

OUTPUT :




```
> g
```

OUTPUT :

```
IGRAPH  D--- 5 3 --
+ edges :
[1] 1->2 2->3 3->1
```

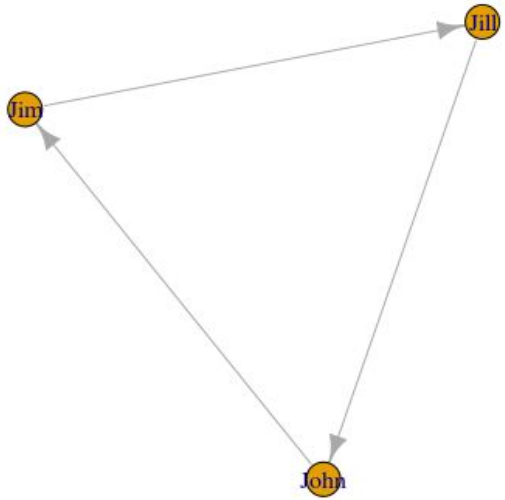
Igraph graph, Directed graph, 5=#vertices, 3=#edges

When the edge list has vertex names, the number of nodes is not needed :

```
> g <- graph ( c ("John", "Jim", "Jim", "Jill", "Jill", "John"))
> plot (g)
```

Named vertices

OUTPUT :



Directed by default

```
> g
```

OUTPUT :

```
IGRAPH DN-- 3 3 --
+ attr: name (v/c)
+ edges (vertex names):
[1] John->Jim  Jim ->Jill Jill->John
```

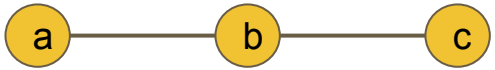
Igraph graph, Directed Named graph, 3=#vertices, 3=#edges
Attribute: named vertex/character

Small graphs can also be generated with a description of this kind:

- for *undirected* tie

```
> plot ( graph_from_literal (a-b, b-c) )
```

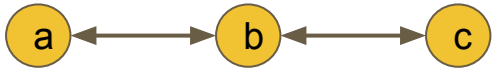
OUTPUT :



++ for *symmetric* tie

```
> plot ( graph_from_literal (a++b, b++c) )
```

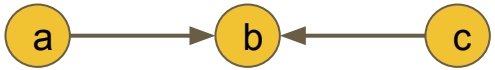
OUTPUT :



+- or **-+** for *directed* ties pointing left & right

```
> plot ( graph_from_literal (a-+b, b+-c) )
```

OUTPUT :



":" for *sets of vertices*

```
> plot ( graph_from_literal (a:b---c:d) )
```

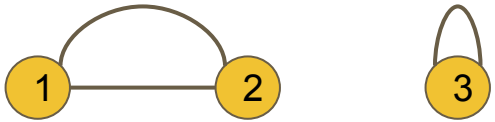
OUTPUT :

Loops and Multiple Edges

```
> g <- graph ( c(1,2,1,2,3,3) , dir=FALSE)
> plot (g)
```

Undirected

OUTPUT :



Check if graph is simple?

```
> is_simple(g)
# False
```

Remove multiple edges

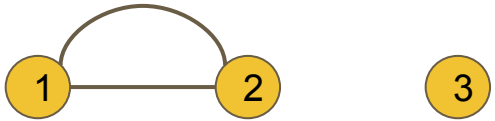
```
> simplify (g, remove.multiple=TRUE)
```

OUTPUT :


```
> simplify (g, remove.loops=TRUE)
```

Remove loop edges

OUTPUT :



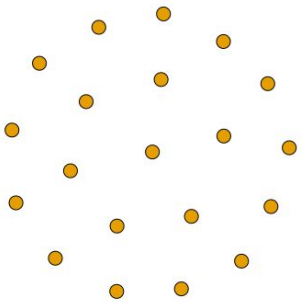
Type of Graphs

Empty Graph

#vertices

```
> eg <- make_empty_graph (20)
> plot (eg)
```

OUTPUT :

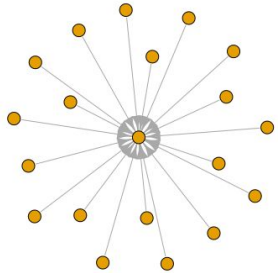


Star Graph

Directed by default

```
> st <- make_star (20)
> plot  (st)
```

OUTPUT :

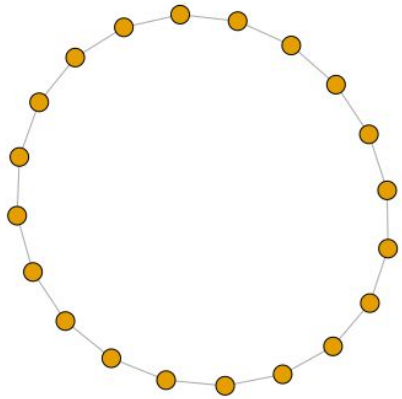


Ring Graph

```
> rg <- make_ring (20)
> plot  (rg)
```

Undirected by default

OUTPUT :

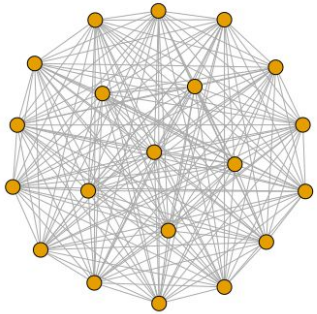


Full Graph

```
> fg <-make_full_graph (20)
> plot  (fg)
```

Undirected by default

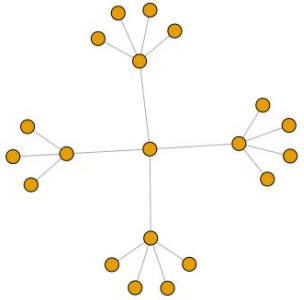
OUTPUT :



Tree Graph

```
> tr<-make_tree(20, children=4, mode="undirected")
> plot  (tr)
```

OUTPUT :

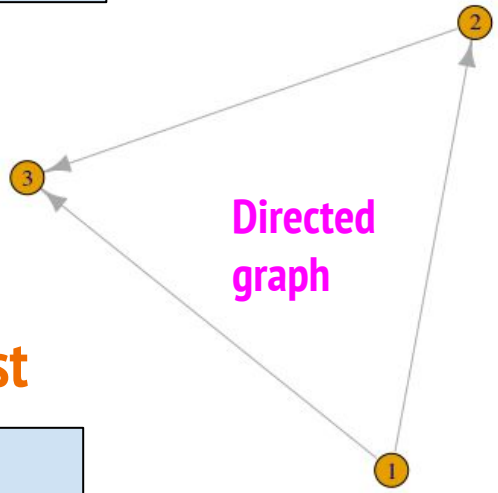


Graph to Edge list

```
> as_edgelist ( graph )
```

OUTPUT :

```
     [,1] [,2]
[1,]    1    2
[2,]    1    3
[3,]    2    3
```



Graph to Adjacency Matrix

```
> as_adjacency_matrix ( graph )
```

OUTPUT :

```
[1,] . 1 1
[2,] . . 1
[3,] . . .
```

Graph to Adjacency List

```
> as_adj_list ( graph )
```

OUTPUT :

```
[[1]]
[1] 2 3

[[2]]
[1] 1 3

[[3]]
[1] 1 2
```

Graph to Data Frame

```
> as_data_frame( graph )
```

OUTPUT :

```
  from to
1    1  2
2    1  3
3    2  3
```

Reading edge list data from file

```
> edges <- read.csv("sample_graph.csv", header=TRUE)
> edges
```

OUTPUT :

	Vertex 1	Vertex 2
1	V1	V2
2	V1	V3
3	V1	V4
4	V1	V5
5	V2	V5
6	V3	V4
7	V4	V5
8	V4	V7
9	V5	V8
10	V6	V2
11	V7	V8

indicating whether the file contains the names of the variables as its first line.

	A	B
1	Vertex1	Vertex2
2	V1	V2
3	V1	V3
4	V1	V4
5	V1	V5
6	V2	V5
7	V3	V4
8	V4	V5
9	V4	V7
10	V5	V8
11	V6	V2
12	V7	V8

sample_graph.csv

```
> head(edges, n=3L)    #retrieves 3 rows from edge list
```

OUTPUT :

	Vertex 1	Vertex 2
1	V1	V2
2	V1	V3
3	V1	V4

Turning data frame(having edge list) into graph object

Use igraph ***graph_from_data_frame*** function :

```
> network<- graph_from_data_frame ( edges, dir = FALSE)
> network
```

OUTPUT :

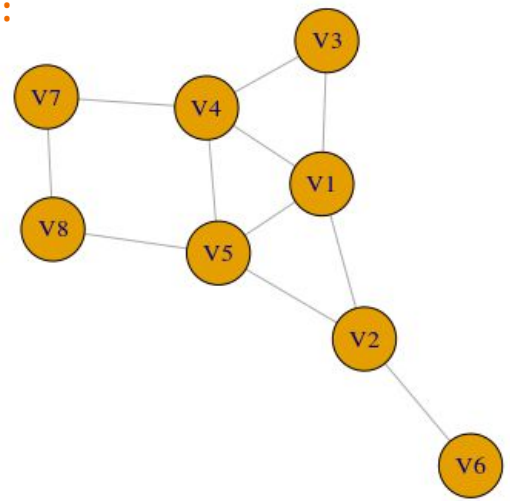
IGRAPH UN-- 8 11 --
+ attr: name (v/c)
+ edges (vertex names):
[1] V1--V2 V1--V3 V1--V4 V1--V5 V2--V5 V3--V4 V4--V5 V4--V7 V5--V8
V2--V6 V7--V8

Igraph graph, Undirected Named graph, 8=#vertices, 11=#edges

Attribute: named vertex/character

```
> plot (network)
```

OUTPUT :



Node Details

```
> V (network)
```

OUTPUT :

+ 8/8 vertices, named:
[1] V1 V2 V3 V4 V5 V6 V7 V8

Edges Details

```
> E (network)
```

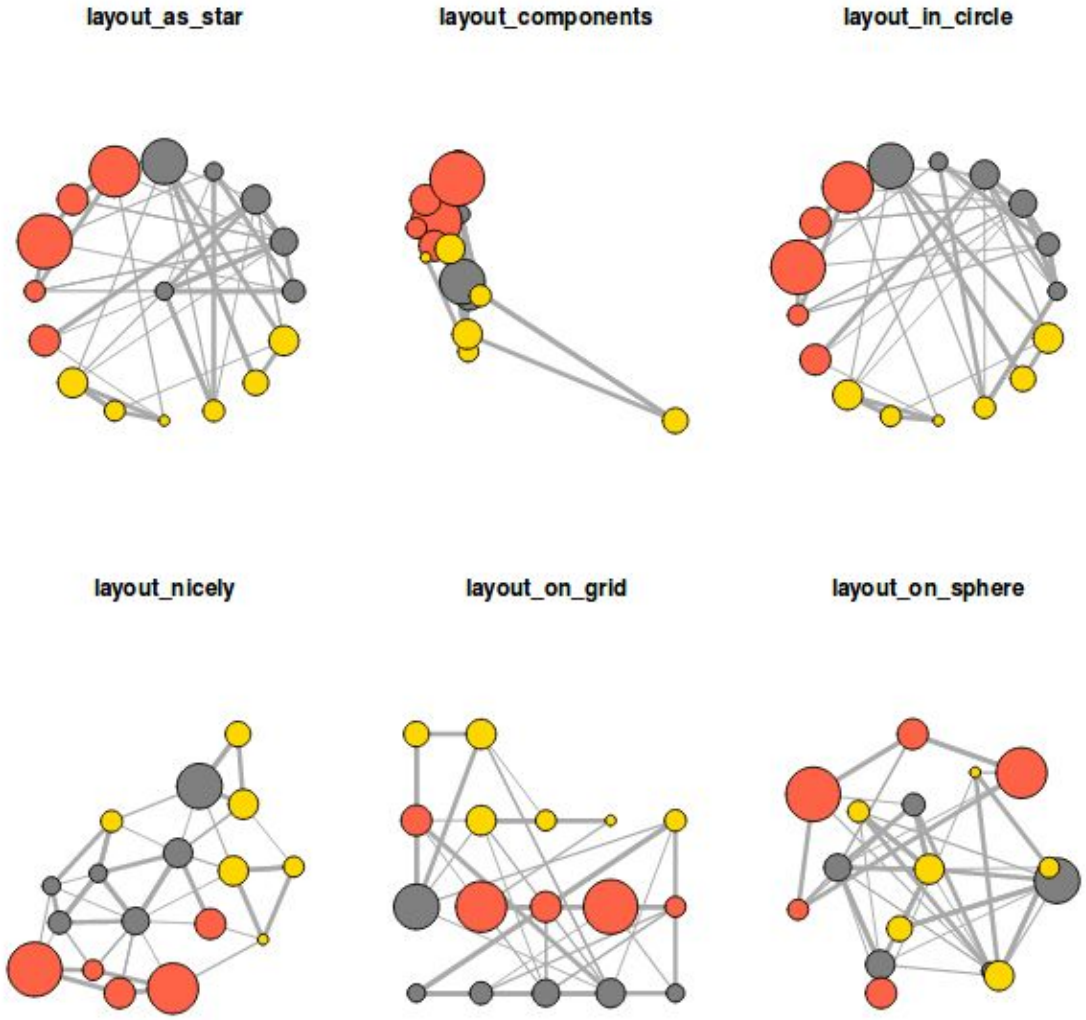
OUTPUT :

+ 11/11 edges (vertex names):
[1] V1--V2 V1--V3 V1--V4 V1--V5 V2--V5 V3--V4
V4--V5 V4--V7 V5--V8 V2--V6 V7--V8

Types of Graph Layouts

```
> plot ( graph, layout = layout_as_star )
```

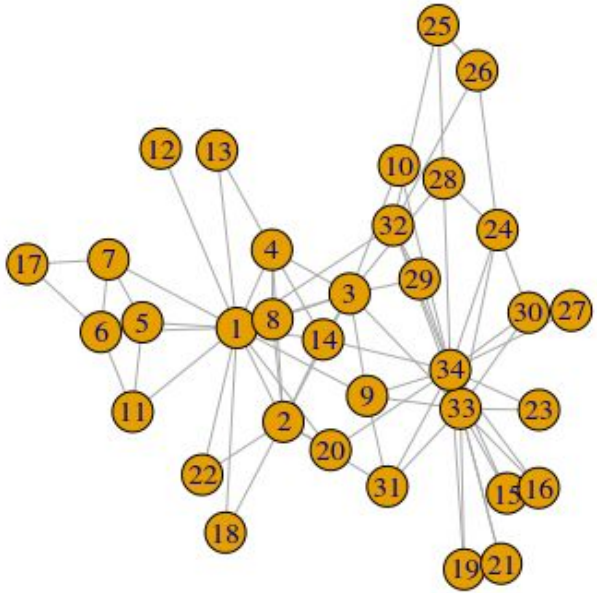
Set different layouts in parameter.



Network from built-in datasets

```
> gz <- graph ("Zachary")  #This is the well-known and much-used Zachary karate club network.

> plot ( gz )
```



Save Graphs

```
# as RData file (.Rda)

> g_el = as_edgelist(g, names = TRUE)
> saveRDS ( g_el,  file = "/Save/To/Location/g.Rda")
```

R object

name of the file and location where the R object is to be saved

```
# as CSV file (.csv)

> g_el = as_edgelist(g, names = TRUE)

> write.csv(g_el, file = "/Save/To/Location/g.csv", row.names = F, col.names = F)
```

Thankyou !