Graph Formats

Contents


```
#install.packages("igraph")
#install.packages("ggplot2")
library('igraph')
library('ggplot2')
```

1 Graph Formats

1.1 Graph Density

Graph algorithms * breadth first search (BFS traversal): O(m) * connected components: O(n + m) * shortest path (Dijkstra): O(m + n log n) * all shortest paths (Floyd-Warshall): O(n^3)

1.1.1 Graph characteristics (directed graph):

$$\mid V \mid = n, \ \mid E \mid = m$$

Graph Density

$$r = \frac{m}{n(n-1)}$$

Average Degree:

$$< k > = \frac{1}{n} \sum_i deg(v_i) = \frac{2m}{n}$$

Average Path:

$$< L > = \frac{1}{n(n-1)} \sum_{i \neq j} d(v_i, v_j)$$

1.2 Graph formats

1.2.1 GraphML

GraphML is XML-base file format for graphs. It has quite clear and flexible format. Along with XML it represents information in hierarchical way:

```
<?xml version="1.0" encoding="UTF-8"?>
<graphml xmlns="http://graphml.graphdrawing.org/xmlns"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://graphml.graphdrawing.org/xmlns
     http://graphml.graphdrawing.org/xmlns/1.0/graphml.xsd">
  <graph id="G" edgedefault="undirected">
    <node id="n0"/>
    <node id="n1"/>
    <node id="n2"/>
    <node id="n3"/>
    <node id="n4"/>
    <node id="n5"/>
    <node id="n6"/>
    <node id="n7"/>
    <node id="n8"/>
    <node id="n9"/>
    <node id="n10"/>
    <edge source="n0" target="n2"/>
    <edge source="n1" target="n2"/>
    <edge source="n2" target="n3"/>
    <edge source="n3" target="n5"/>
    <edge source="n3" target="n4"/>
    <edge source="n4" target="n6"/>
    <edge source="n6" target="n5"/>
    <edge source="n5" target="n7"/>
    <edge source="n6" target="n8"/>
    <edge source="n8" target="n7"/>
    <edge source="n8" target="n9"/>
    <edge source="n8" target="n10"/>
  </graph>
</graphml>
```

1.2.2 GML

Another flexible format and a bit more readible format is GML (Graph Modelling Language).

```
#change all id to numbers if error occured
graph
[
  directed 1
  node
  [
   id A
   label "Node A"
  ]
  node
  [
   id B
   label "Node B"
  ]
  node
  [
   id C
```

```
   label "Node C"
  ]
   edge
  [
   source B
   target A
   label "Edge B to A"
  ]
  edge
  [
   source C
   target A
   label "Edge C to A"
  ]
]
```

1.2.3 Pajek Format

Pajek format emerges from a large network analysis tool. It is not that readible, however it is often used to store big netoworks

```
*Vertices 82670
1 "entity"
2 "thing"
3 "anything"
4 "something"
5 "nothing"
6 "whole"
*arcs
1 2 5161
1 9 5615
2 9 7894
2 3 812
2 8 123
3 8 15
3 4 456
4 5 456
4 7 4
5 7 4568
5 6 456
6 4 4849
```

1.2.4 Edge List, Adjacency List

In the first case this is simply a pair of connected node that may be supported by a weight.

```
a;b
c;d
```

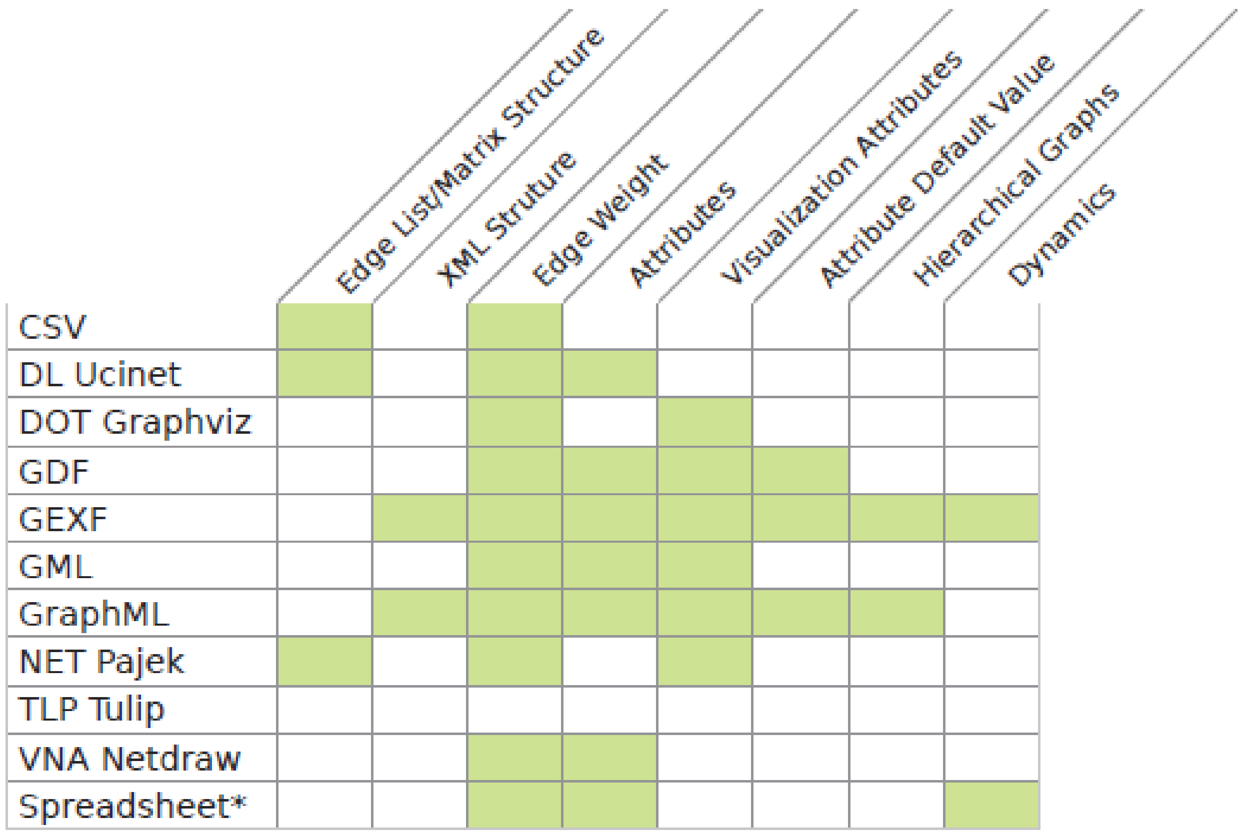
In the second case this is a sequence of nodes – each line contains neigbours of a node that is stays on the first place

```
1 3 4 5
2 3 8
3 9
```

1.2.5 Adjacency matrix

A very rarely used format as it stores minimum amount of information about network and uses lots of space..

```
A;B;C;D;E
A;0;1;0;1;0
B;1;0;0;0;0
C;0;0;1;0;0
D;0;1;0;1;0
E;0;0;0;0;0
```



1.2.6 Loading networks with igraph

Basically a single command is sufficient to read network data to R:

```
g <- read.graph(file = '...',format = '...', )
```

Only in case of adjacency matrix you have to do a bit more..

```
dat <- read.csv(file = '...', sep = ' ')
m <- as.matrix(dat)
g <- graph.adjacency(m, mode='...', weighted = ...)
```

1.2.6.1 Task

Open extract files from supplimentary achive. Try to load and plot some of the networks there.

1.3 Power laws

Now, we are goint to generate power-law distribution syntetically.

During this part we will determine the meaning of α and try to estimate it.

The functions below are for generating random variables distributed according to power-law and creating histogramm based on generated data.

```
generate_dist = function(xmin, xmax, alpha, size) {
  r <- runif(size)
  return((xmin^(-alpha+1)+r*(xmax^(-alpha+1)-xmin^(-alpha+1)))^(1/(-alpha+1)))
}
distribution = function(h){
  prob <- rev(h$counts/sum(h$counts))
  cumProb <- cumsum(prob)
  wage <- cumsum(prob*rev(h$mids))/sum(prob*rev(h$mids))
  return(data.frame(wage=wage, prob=cumProb))
}
```

1.3.1 Meaning

That is a very legitimate question. In fact α shows how fairly degree of the distribution is spread in the networks. To show this, lets generate distributions with various parametes.

```
xmin <- 1
xmax <- 100
alpha <- 3.1
n <- 1000

x <- generate_dist(xmin, xmax, alpha, n)
h <- hist(x, breaks = length(unique(x)))
```

```
l <- distribution(h)
qplot(data=l, x = wage, y = prob, geom = 'line') +
  xlab('Proportion of top degrees') +
  ylab('Proportion of nodes')
```

As you may see, the bigger α is the more fairly degree is distrubuted among the vertices.

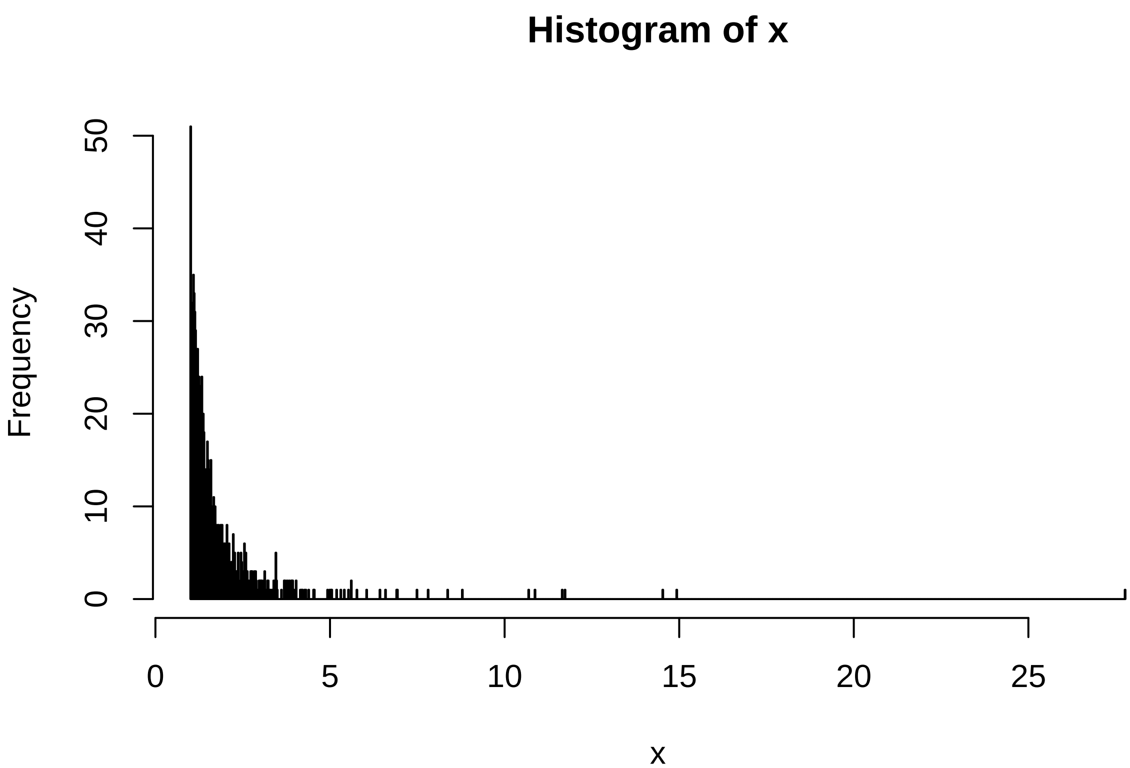


Figure 1:

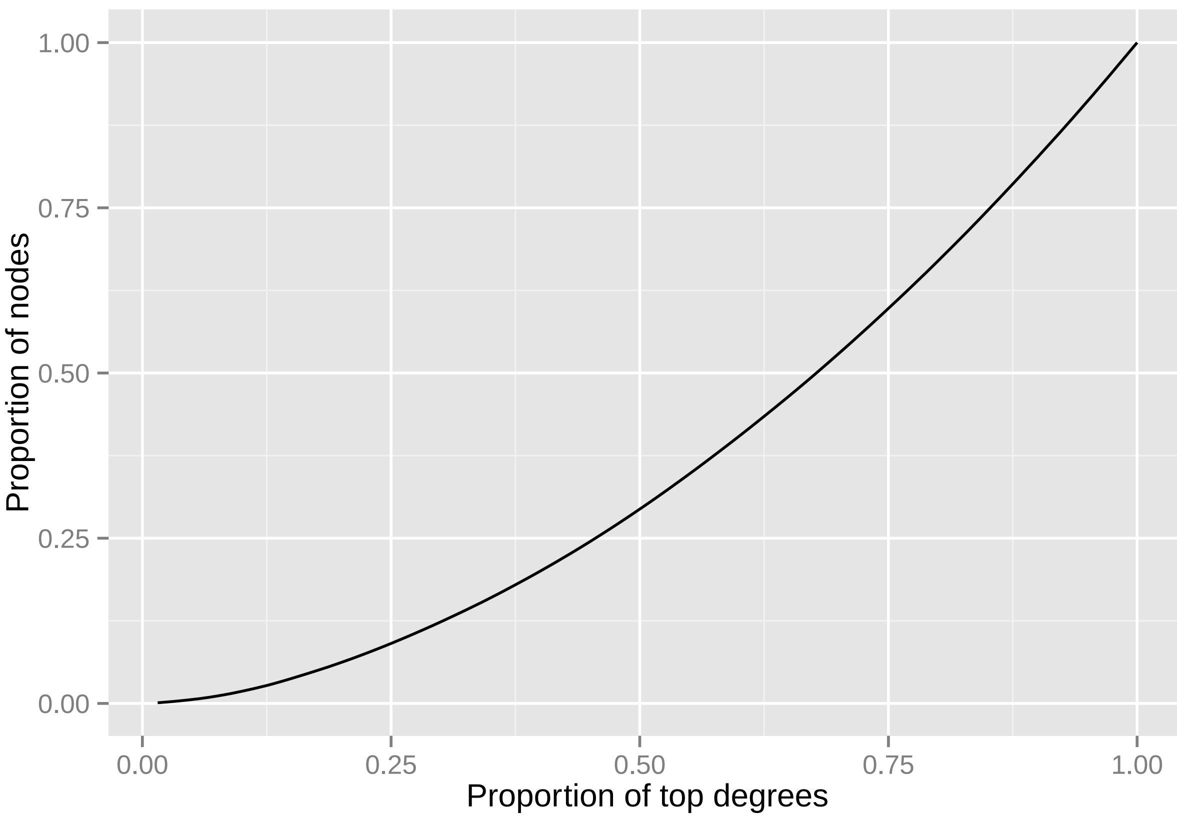

Proportion of nodes
1.00
0.75
0.50
0.25
0.00
0.00
0.25
0.50
0.75
1.00
Proportion of top degrees

Figure 2:

1.3.2 Estimation

We are using build-in igraph function to estimate α of destribution

```
g = read.graph('netscience.gml', format='gml')
d = degree(g)
h = hist(d, breaks = 1000)
```

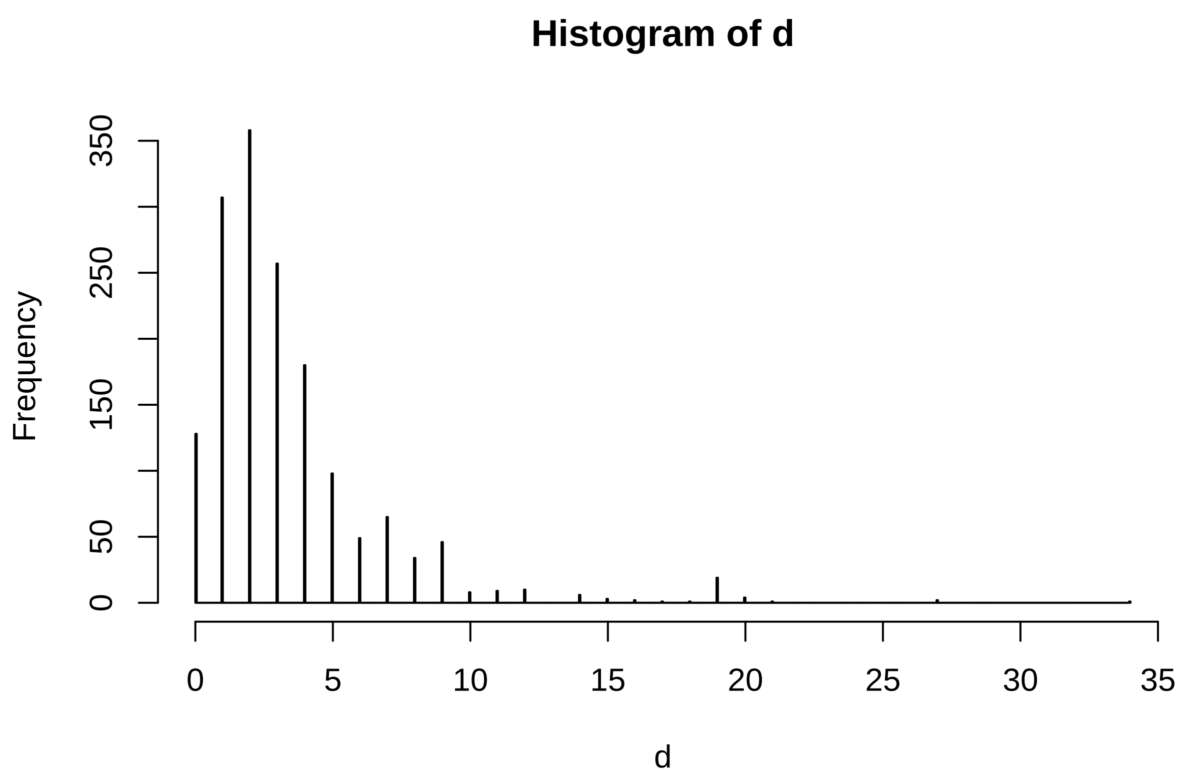


Figure 3:

```
fit = power.law.fit(d, 1, implementation = "plfit")
alpha = fit$alpha
print(alpha)
```

```
## [1] 2
```

The final part of the task is to calculate parameters and goodness-of-fit for several networks: - netscience.gml - Cit-HepPh.txt

Hints:

`g = read.graph ...`

`degree(g)`

`power.law.fit`

```
fit$alpha
```

```
fit$KS.p
```

1.4 Descriptive statistics

1.4.1 Shortest path, diameter

Path is a finite or infinite sequence of edges which connect a sequence of vertices

```
g <- graph.formula(1-2-3-4-5-6, 1-3-7, 1-8-4)
plot(g)
```

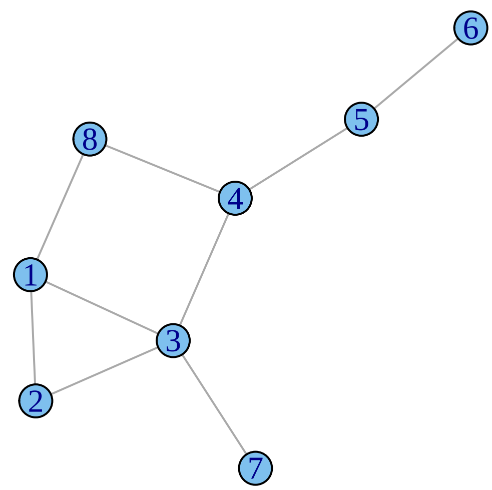


Figure 4:

```
shortest.paths(g)
```

```
##   1 2 3 4 5 6 7 8
## 1 0 1 1 2 3 4 2 1
## 2 1 0 1 2 3 4 2 2
## 3 1 1 0 1 2 3 1 2
## 4 2 2 1 0 1 2 2 1
## 5 3 3 2 1 0 1 3 2
## 6 4 4 3 2 1 0 4 3
```

```
## 7 2 2 1 2 3 4 0 3
## 8 1 2 2 1 2 3 3 0
```

The shortest path from one node to others

```r
get.shortest.paths(g,1)$vpath
```

```
## [[1]]
## [1] 1
##
## [[2]]
## [1] 1 2
##
## [[3]]
## [1] 1 3
##
## [[4]]
## [1] 1 3 4
##
## [[5]]
## [1] 1 3 4 5
##
## [[6]]
## [1] 1 3 4 5 6
##
## [[7]]
## [1] 1 3 7
##
## [[8]]
## [1] 1 8
```

```r
get.shortest.paths(g,1,4)$vpath
```

```
## [[1]]
## [1] 1 3 4
```

```r
get.all.shortest.paths(g,1,4)$res
```

```
## [[1]]
## [1] 1 8 4
##
## [[2]]
## [1] 1 3 4
```

```r
diameter(g) # the length of the "longest shortest path"
```

```
## [1] 4
```

```r
get.diameter(g)
```

```
## [1] 1 3 4 5 6
```

```
E(g)$color <- "blue"
E(g, path=get.diameter(g))$color <- "red"
plot(g)
```

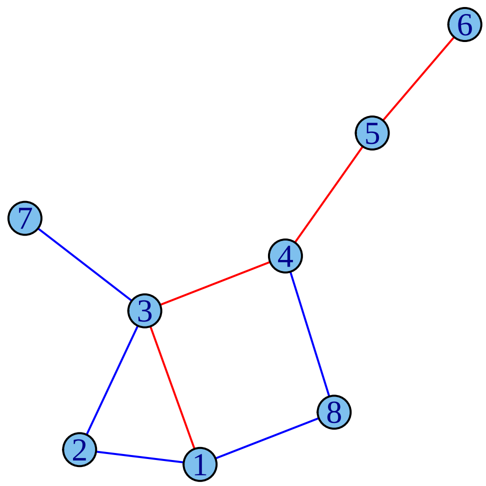


Figure 5:

```
radius(g) # is the length of the shortest paths among longest ones
```

```
## [1] 2
```

The shortest path between two certain nodes

```
get.shortest.paths(g,1,4)$vpath # only one arbitary path
```

```
## [[1]]
## [1] 1 3 4
```

```
get.all.shortest.paths(g,1,4)$res #  all paths
```

```
## [[1]]
## [1] 1 8 4
##
## [[2]]
## [1] 1 3 4
```

```r
diameter(g) # the length of the "longest shortest path"
```

```
## [1] 4
```

The longest path among shortest paths

```r
diameter(g) # the length of the "longest shortest path"
```

```
## [1] 4
```

```r
get.diameter(g)
```

```
## [1] 1 3 4 5 6
```

```r
E(g)$color <- "blue"
E(g, path=get.diameter(g))$color <- "red"
plot(g)
```

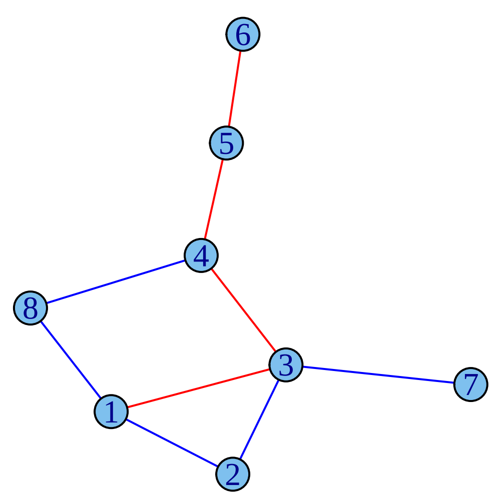


Figure 6:

```
radius(g) # is the length of the shortest paths among longest ones
```

```
## [1] 2
```

Recall from the previous seminar

```
vcount(g) # number of nodes
```

```
## [1] 8
```

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

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

```
E(g) # list of edges
```

```
## Edge sequence:
##
## [1] 2 -- 1
## [2] 3 -- 1
## [3] 8 -- 1
## [4] 3 -- 2
## [5] 4 -- 3
## [6] 7 -- 3
## [7] 5 -- 4
## [8] 8 -- 4
## [9] 6 -- 5
```

```
V(g) # list of nodes
```

```
## Vertex sequence:
## [1] "1" "2" "3" "4" "5" "6" "7" "8"
```

1.4.1.1 The shortest path length distribution *Geting a histogram, by calculating the shortest path length between each pair of vertices*

```
g<-graph.formula(1-2-3-4-5-6, 1-3-7, 1-8-4, 3-9-10-12, 11-12-13-3)
tab <- as.table(path.length.hist(g)$res)
names(tab) <- 1:length(tab)
barplot(tab)
```

1.4.2 Density

The density of a graph is the ratio of the number of edges and the number of possible edges.

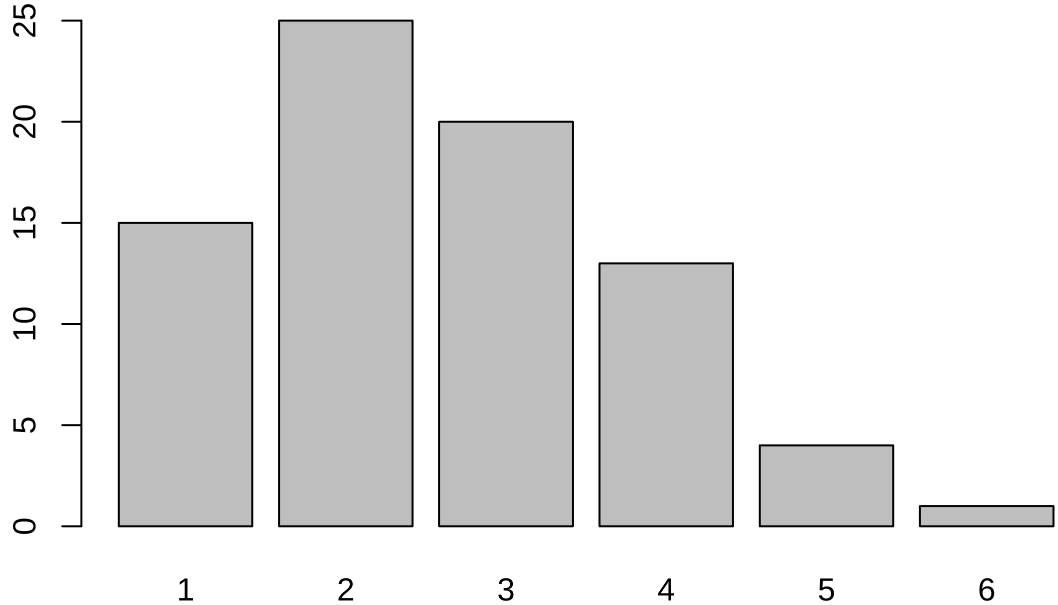


Figure 7:

```r
graph.density(g)
```

```
## [1] 0.1923077
```

1.4.2.1 Working Examples

1.4.2.1.1 Zachary's Karate Club *The data was collected from the members of a university karate club by Wayne Zachary in 1977. Each node represents a member of the club, and each edge represents a tie between two members of the club. Zachary studied conflict and fission in this community network, as the karate club was split into two separate clubs. The network is very small: it has 34 vertices and 78 undirected edges.*

```r
g <- graph.famous("Zachary")
plot(g)
```

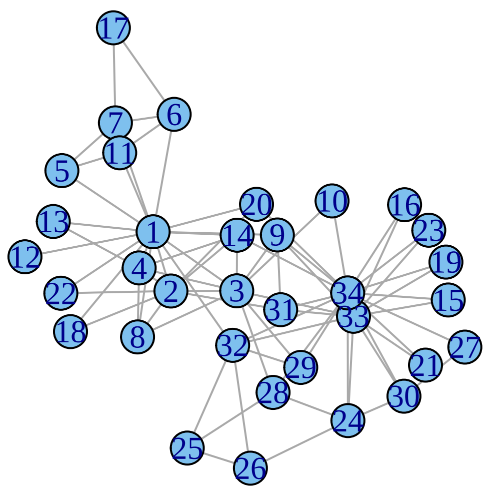


Figure 8:

```r
graph.density(g)
```

```
## [1] 0.1390374
```

```
diameter(g) # the length of the "longest shortest path"
```

```
## [1] 5
```

```
E(g)$color <- "blue"
E(g, path=get.diameter(g))$color <- "red"
plot(g,layout = layout.fruchterman.reingold)
```

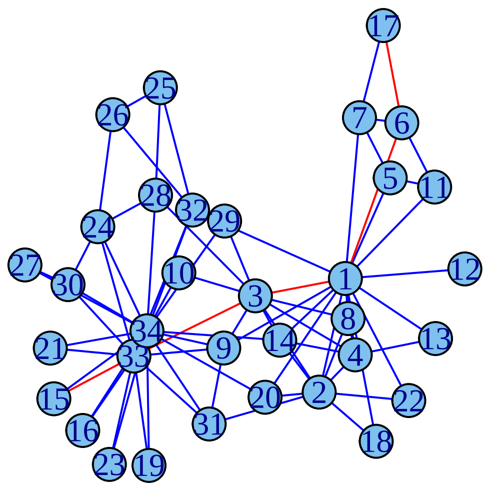


Figure 9:

```
tab <- as.table(path.length.hist(g)$res)
names(tab) <- 1:length(tab)
barplot(tab,legend = c(path.length.hist(g)$res),main = "The shortest path lengths distribution", col = c
```

1.4.2.2 High-energy physics citation network See more information on http://snap.stanford.edu/data/cit-HepPh.html

```
g <- read.graph("Cit-HepPh.txt", format="edgelist")
vcount(g)
```

```
## [1] 9912554
```

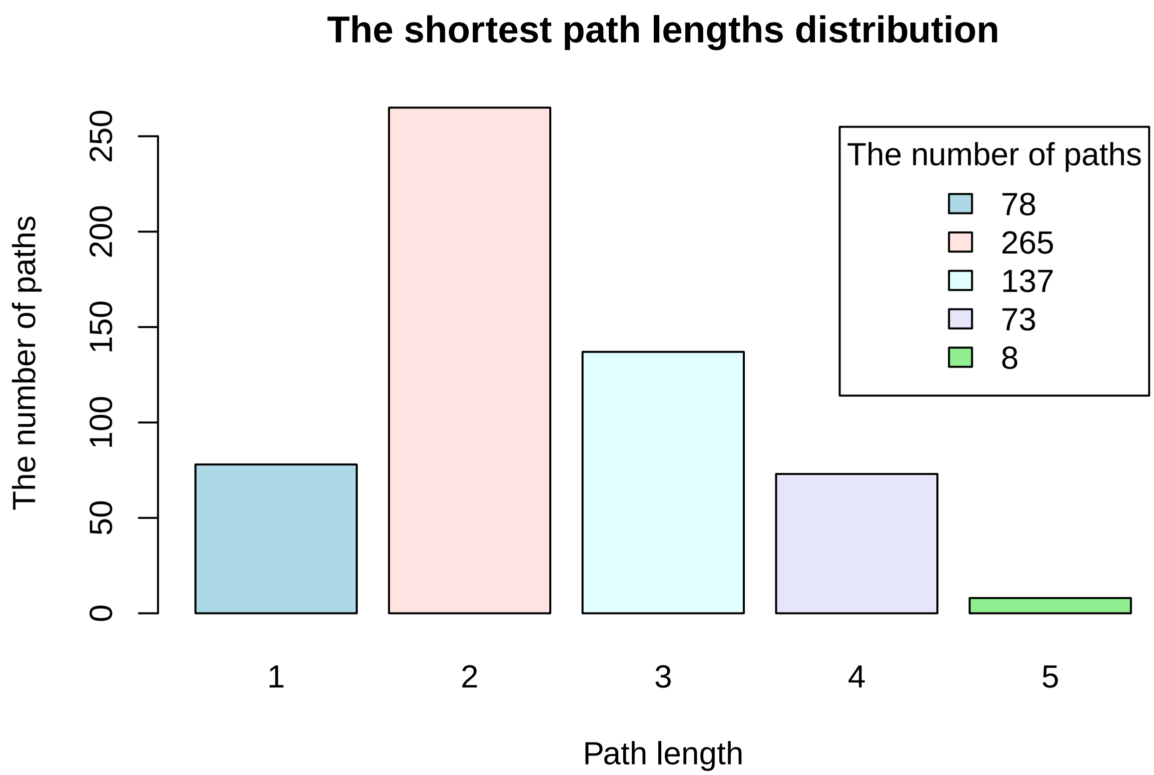

The shortest path lengths distribution
The number of paths
78
265
137
73
8
0
50
100
150
200
250
The number of paths
1
2
3
4
5
Path length

Figure 10:

```r
ecount(g)
```

```
## [1] 421578
```

```r
tab <- as.table(path.length.hist(g)$res)
max(tab)
```

```
## [1] 42044044
```

```r
min(tab)
```

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

```r
names(tab) <- 1:length(tab)
barplot(tab,main = "The shortest path lengths distribution",xlab = "Path length", ylab = "The number of
```

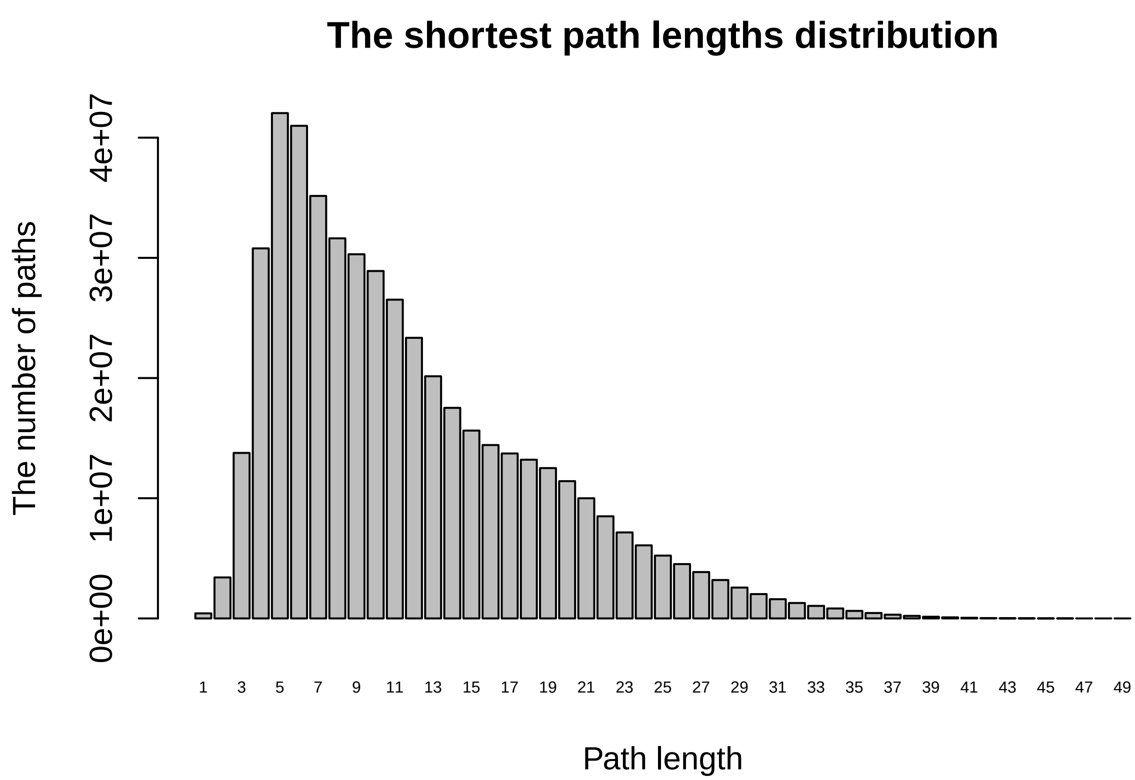


Figure 11:

1.4.2.3 Other examples. . .

```r
g <- graph.ring(8)
plot(g)
```

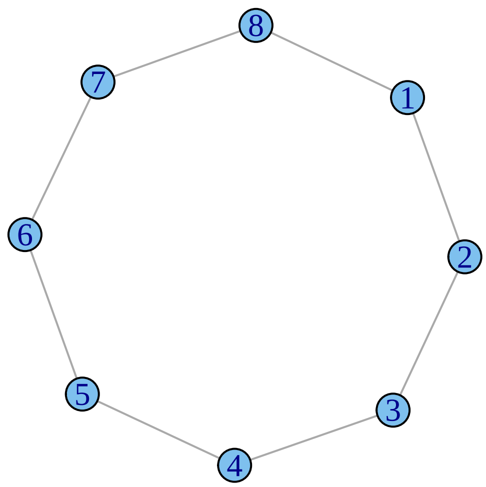


Figure 12:

```r
shortest.paths(g)
```

1.4.2.3.1 1. Ring graph

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

```r
get.diameter(g)
```

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

```r
diameter(g)
```

```
## [1] 4
```

```r
radius(g)
```

```
## [1] 4
```

```r
g <- graph.formula(1-2-3-4-5-6,7-8-9-3-10, 4-11)
E(g)$color <- "blue"
E(g, path=get.diameter(g))$color <- "red"
plot(g)
```

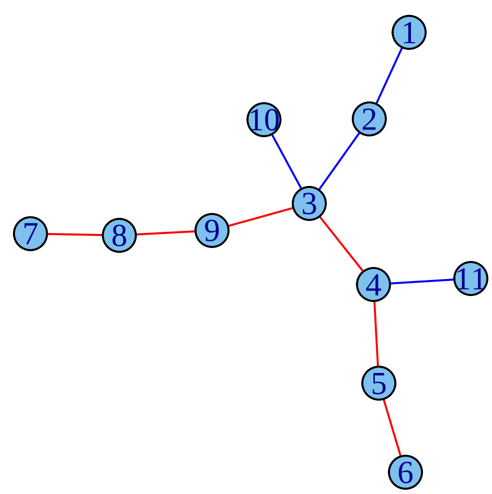


Figure 13:

```r
shortest.paths(g)
```

1.4.2.3.2 2. An arbitary graph

```
##    1 2 3 4 5 6 7 8 9 10 11
## 1  0 1 2 3 4 5 5 4 3  3  4
## 2  1 0 1 2 3 4 4 3 2  2  3
## 3  2 1 0 1 2 3 3 2 1  1  2
## 4  3 2 1 0 1 2 4 3 2  2  1
## 5  4 3 2 1 0 1 5 4 3  3  2
## 6  5 4 3 2 1 0 6 5 4  4  3
## 7  5 4 3 4 5 6 0 1 2  4  5
## 8  4 3 2 3 4 5 1 0 1  3  4
## 9  3 2 1 2 3 4 2 1 0  2  3
## 10 3 2 1 2 3 4 4 3 2  0  3
## 11 4 3 2 1 2 3 5 4 3  3  0
```

```r
diameter(g)
```

```
## [1] 6
```

```r
get.diameter(g)
```

```
## [1] 6 5 4 3 9 8 7
```

```r
radius(g)
```

```
## [1] 3
```

```r
set.seed(1)
g <- graph.formula(1-2-3-4-5-1)
#E(g)$weight <- seq_len(10)
E(g)$weight <- sample(1:(ecount(g)))
E(g)$color <- "blue"
E(g, path=get.diameter(g))$color <- "red"
plot(g,edge.label = E(g)$weight)
```

```r
shortest.paths(g)
```

1.4.2.3.3 3. Weighted graph

```
##   1 2 3 4 5
## 1 0 2 6 6 5
## 2 2 0 4 7 7
## 3 6 4 0 3 4
## 4 6 7 3 0 1
## 5 5 7 4 1 0
```

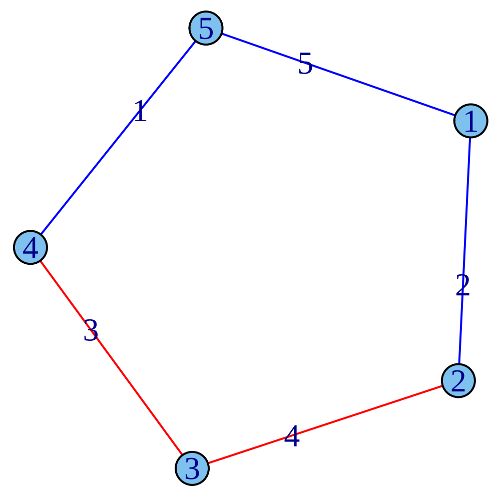


Figure 14:

```r
diameter(g)
```

```
## [1] 7
```

```r
diameter(g, weights=NA)
```

```
## [1] 2
```

```r
E(g)$color <- "blue"
E(g, path=get.diameter(g, weights=NA))$color <- "red"
plot(g, edge.label = E(g)$weight)
```

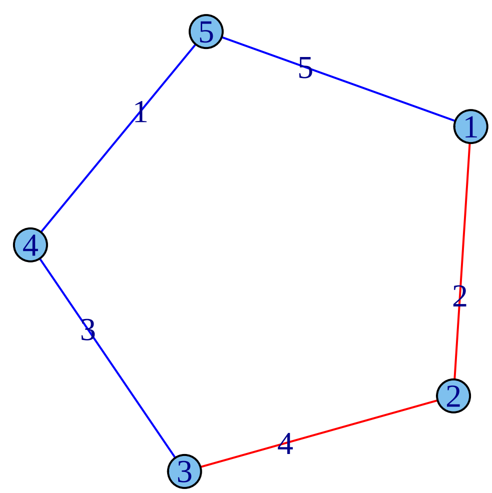


Figure 15:

```r
shortest.paths(g, weights=NA)
```

```
##   1 2 3 4 5
## 1 0 1 2 2 1
## 2 1 0 1 2 2
## 3 2 1 0 1 2
## 4 2 2 1 0 1
## 5 1 2 2 1 0
```

1.4.3 Connected components of a graph

A connected component is a maximal connected subgraph of G. Each vertex belongs to exactly one connected component, as does each edge. To check, whether the graph is connected use command

```
is.connected(g)
```

A directed graph is called **weakly connected** if replacing all of its directed edges with undirected edges produces a connected (undirected) graph. It is **connected** if it contains a directed path from u to v or a directed path from v to u for every pair of vertices u, v. It is **strongly connected** if it contains a directed path from u to v and a directed path from v to u for every pair of vertices u, v.

```
clusters(graph, mode=c("weak", "strong")) # select the required mode
no.clusters(graph, mode=c("weak", "strong"))
```

```


g<-read.graph(file='tutorial2_1.txt', format="edgelist")
plot(g)
```

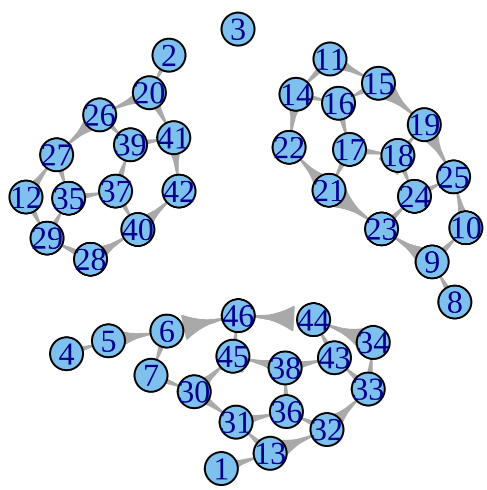


Figure 16:

```r
clusters(g, mode=c("weak")) # select the required mode
```

1.4.3.1 Directed graphs

```
## $membership
##  [1] 1 2 3 1 1 1 1 4 4 4 4 2 1 4 4 4 4 4 4 2 4 4 4 4 4 2 2 2 2 1 1 1 1 1 2
## [36] 1 2 1 2 2 2 2 1 1 1 1
##
## $csize
## [1] 17 13  1 15
##
## $no
## [1] 4
```

```r
clusters(g, mode=c("strong")) # select the required mode
```

```
## $membership
##  [1] 35 25 24 19 20 21 22 13 14 17  4  2 36  6  5  7  8 11 12 26  9 10 15
## [24] 16 18 27 28  1  3 23 40 37 38 39 29 41 30 42 32 31 33 34 44 45 43 46
##
## $csize
##  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [36] 1 1 1 1 1 1 1 1 1 1 1
##
## $no
## [1] 46
```

```r
lst <-cluster.distribution(g, cumulative = FALSE)
distr<-table(lst,seq(1,length(lst)))[2,]
barplot(distr,main = "The maximal connected component sizes distribution", ylab = "The number of compon
```

```r
g<-read.graph(file='tutorial2_2.txt', format="edgelist")
plot(g)
```

```r
clusters(g, mode=c("weak")) # select the required mode
```

```
## $membership
##  [1] 1 1 1 1 1 1 1 1 1 1 1
##
## $csize
## [1] 11
##
## $no
## [1] 1
```

```r
clusters(g, mode=c("strong")) # select the required mode
```

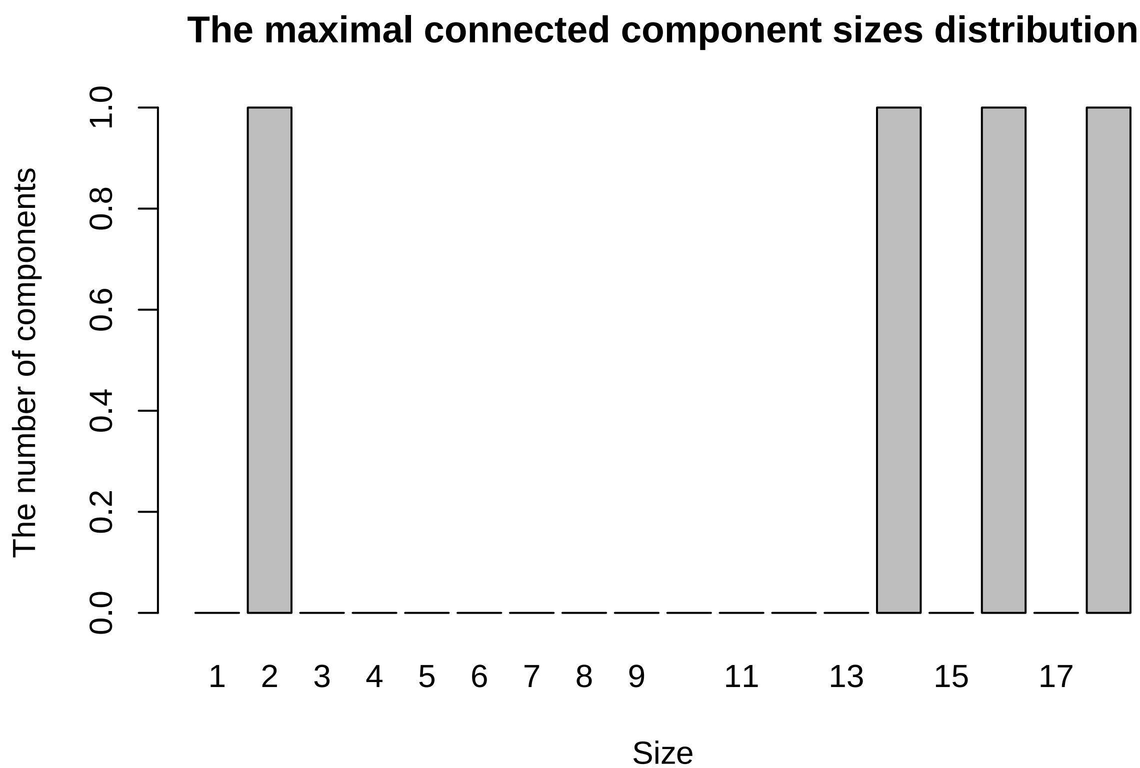

The maximal connected component sizes distribution
The number of components
0.0
0.2
0.4
0.6
0.8
1.0
1
2
3
4
5
6
7
8
9
11
13
15
17
Size

Figure 17:

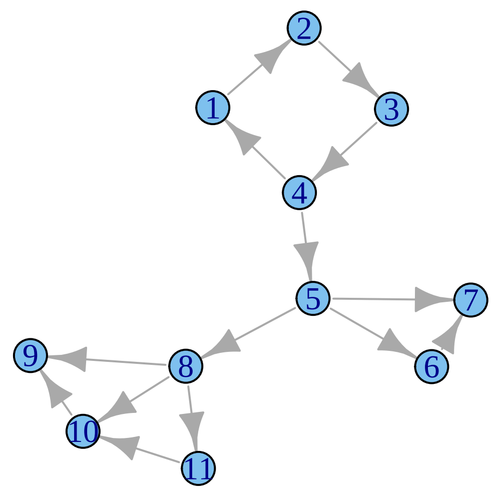


Figure 18:

```
## $membership
##  [1] 1 1 1 1 2 7 8 3 6 5 4
##
## $csize
## [1] 4 1 1 1 1 1 1 1
##
## $no
## [1] 8
```

```
lst <-cluster.distribution(g, cumulative = FALSE)
distr<-table(lst,seq(1,length(lst)))[2,]
barplot(distr,main = "The maximal connected component sizes distribution", ylab = "The number of compone
```

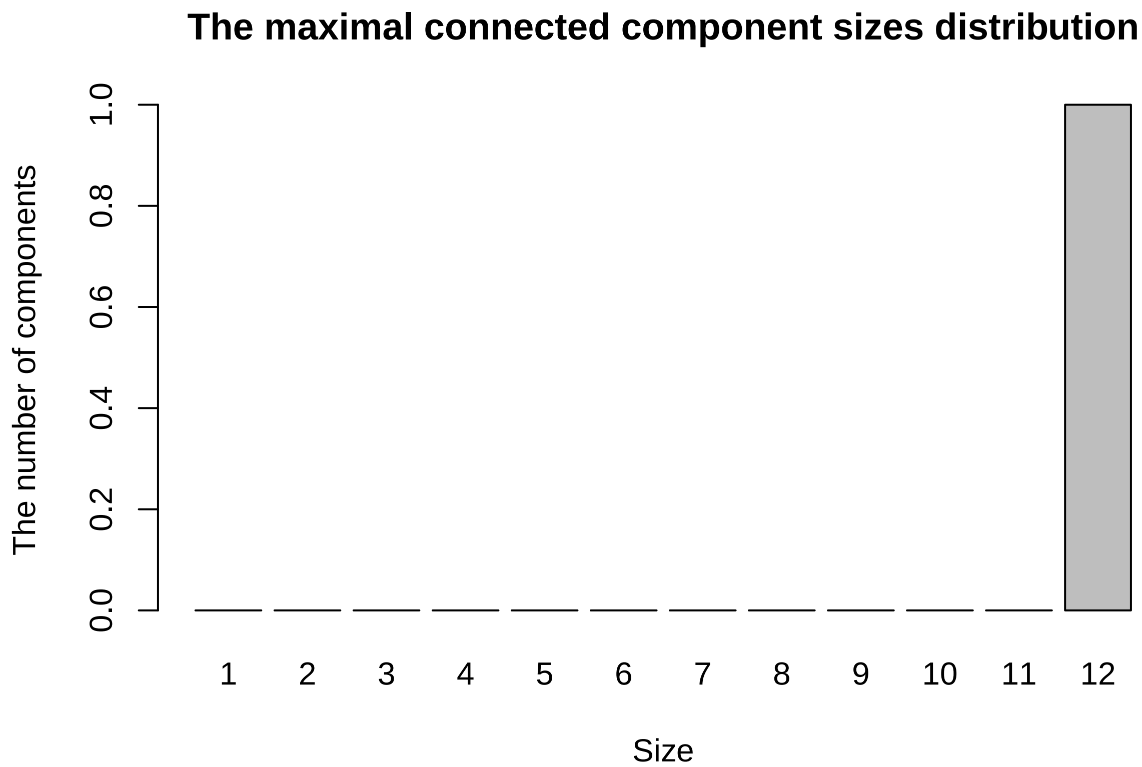


Figure 19:

```
g<-read.graph(file='tutorial2_3.txt', format="edgelist")
plot(g)
```

```
clusters(g, mode=c("weak")) # select the required mode
```

```
## $membership
##  [1] 1 1 1 1 1 1 1 2 3 3
##
```

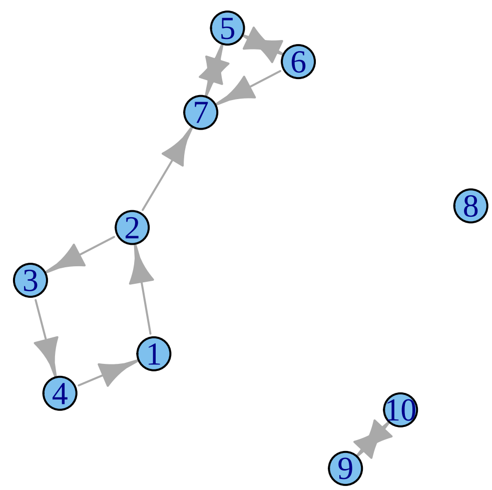


Figure 20:

```
## $csize
## [1] 7 1 2
##
## $no
## [1] 3
```

```
clusters(g, mode=c("strong")) # select the required mode
```

```
## $membership
##  [1] 3 3 3 3 4 4 4 2 1 1
##
## $csize
## [1] 2 1 4 3
##
## $no
## [1] 4
```

```
lst <-cluster.distribution(g, cumulative = FALSE)
distr<-table(lst,seq(1,length(lst)))[2,]
barplot(distr,main = "The maximal connected component sizes distribution", ylab = "The number of compone
```

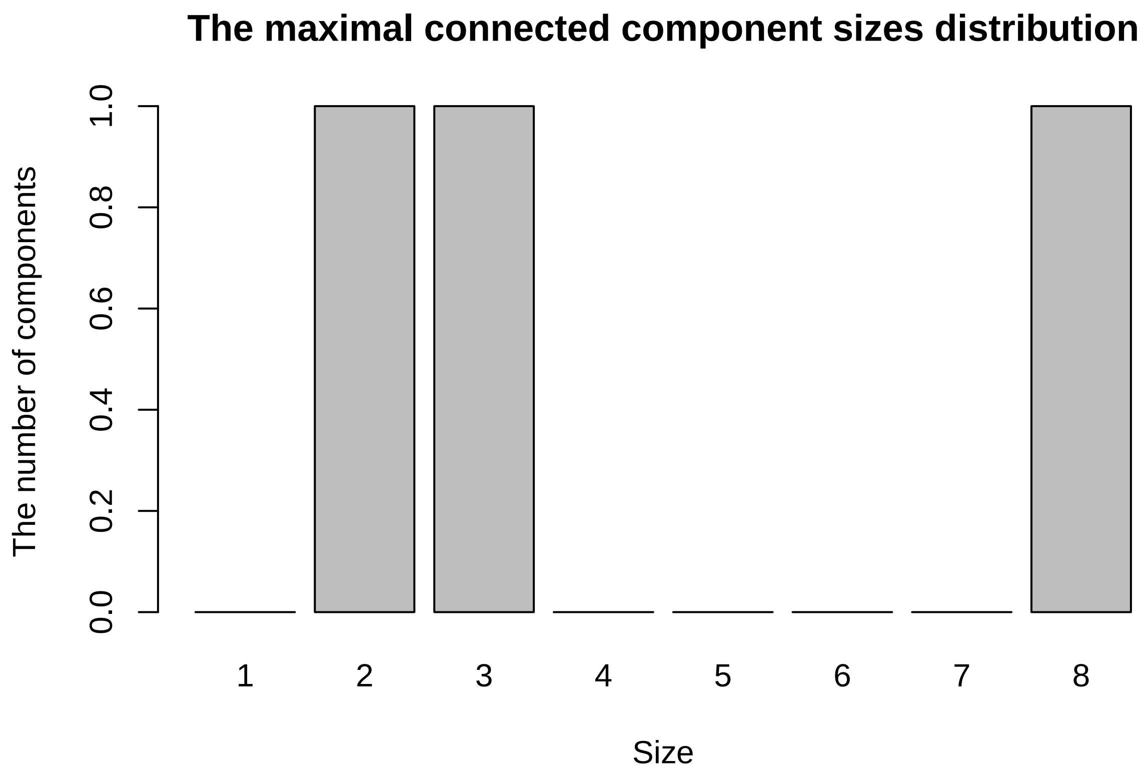


Figure 21:

1.4.4 Measures of vertices clustering

2pen triplet

```
plot(graph.formula(A-B-C))
```

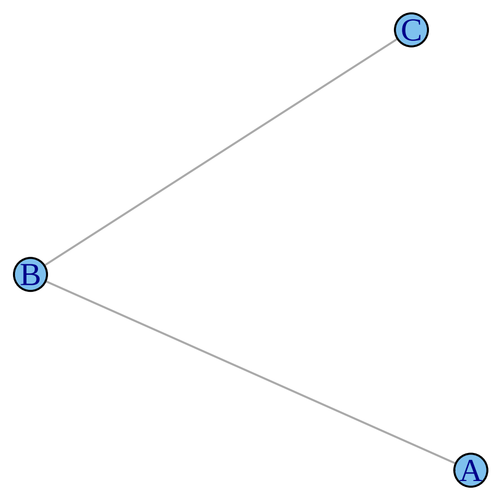


Figure 22:

Closed triplet

```
plot(graph.formula(A-B-C-A))
```

The transitivity

$$\frac{the\ number\ of\ closed\ triplets}{the\ total\ number\ of\ triples\ centered\ on\ each\ of\ the\ vertices}$$

v_i - the i-th vertex of a graph k_i - vertex degree (the number of edges incident to the vertex) e_{ij} - the edge between i-th and j-th verticies

The local clustering coefficient for node v_i

$$C_v = \frac{the\ number\ of\ closed\ triplets\ of\ the\ node}{the\ number\ of\ triples\ centered\ around\ the\ node}$$

The global clustering coefficient of the graph is average over all local clustering coefficients C_v.

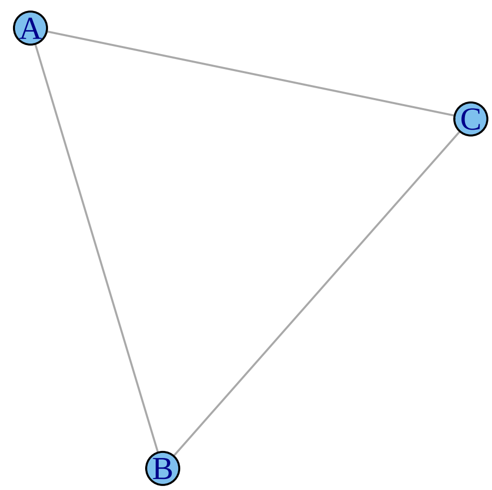


Figure 23:

$$\overline{C} = \frac{1}{n}\sum_{v} C_v$$

```
g <- graph.formula(A-B-C-A,C-D)
plot(g)
```

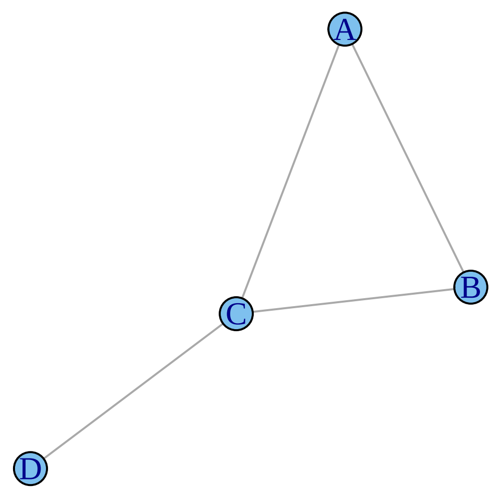


Figure 24:

```
transitivity(g, type = "local")
```

```
## [1] 1.0000000 1.0000000 0.3333333       NaN
```

```
transitivity(g, type = "global") # transitivity(g)
```

```
## [1] 0.6
```

```
transitivity(g, vids="C", type = "local")
```

```
## [1] 0.3333333
```

```r
transitivity(g, vids="A", type = "local")
```

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

```r
transitivity(g, vids="B", type = "local")
```

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

```r
g <- graph.formula(A - D, B - D, C - D, A - B - C)
plot(g)
```

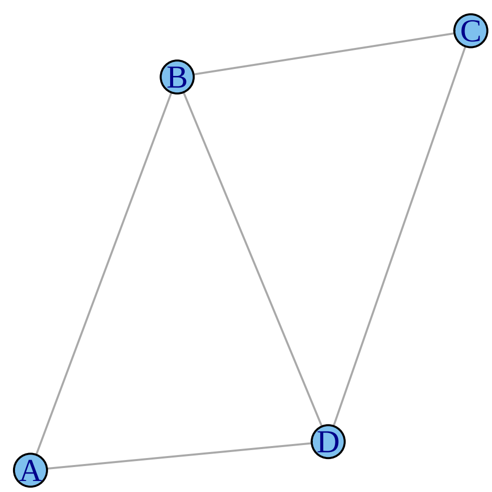


Figure 25:

```r
transitivity(g, type = "local")  #  local clustering coefficient
```

```
## [1] 1.0000000 0.6666667 0.6666667 1.0000000
```

```r
transitivity(g, vids="A", type = "local")
```

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

```r
transitivity(g, type = "global") #  global clustering coefficient
```

```
## [1] 0.75
```

```r
transitivity(g, type="localaverage") # average the local clustering coefficient
```

```
## [1] 0.8333333
```

```r
transitivity(g, vids="C", type = "local")
```

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

```r
diameter(g)
```

```
## [1] 2
```

```r
gw <- graph.formula(A-B-C-A : D-E, E-Y-A)
plot(gw)
```

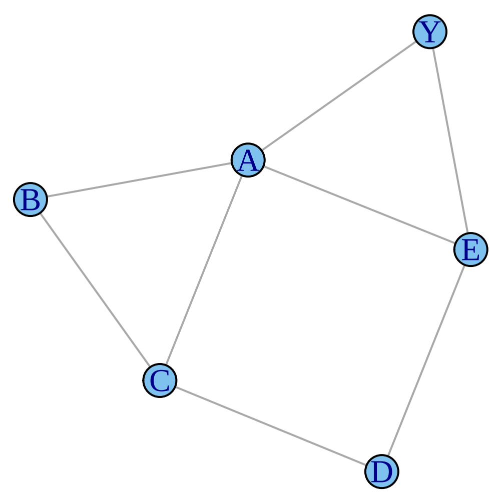


Figure 26:

```r
transitivity(gw, vids="A", type = "local")
```

```
## [1] 0.3333333
```

```r
transitivity(gw, vids="B", type = "local")
```

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

```r
transitivity(gw, vids="C", type = "local")
```

```
## [1] 0.3333333
```

```r
transitivity(gw, vids="D", type = "local")
```

```
## [1] 0
```

```r
transitivity(gw, type="localaverage")
```

```
## [1] 0.5
```

```r
transitivity(gw, type = "global")
```

```
## [1] 0.4
```