Graph Theory: Basic Concepts

Roberto Alvarez

13 June 2018

Contents

Please don't copy-paste the code!
Type by yourself the code!
Use TAB to do code completion (a lot) !
You will only fail to learn if you do not learn from failing.
Use TAB to do code completion (~~a lot~~ ALWAYS) !

---

igraph is a huge R library. Igraph is available as R, python and C library.
igraph. This course is not intended as an igraph course.

In R, igraph is installed in the usual way, i. e.

install.packages("igraph")

```
library(igraph)
```

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

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

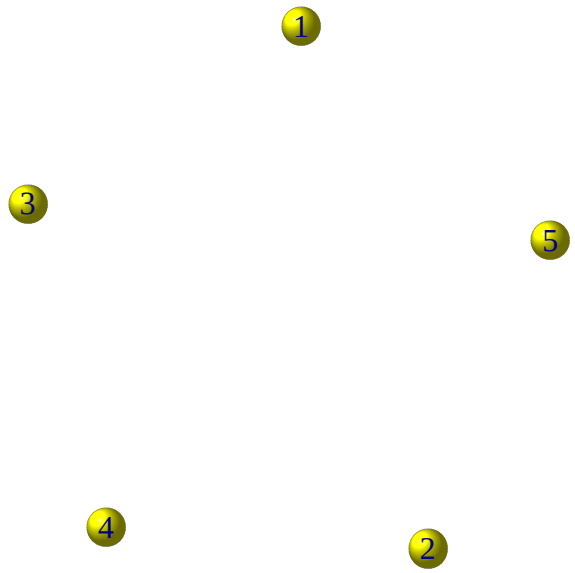
```
## The following object is masked from 'package:base':
##
##     union
```

An empty graph/network with five spherical yellow vertex is created by

```
g <- make_empty_graph(n=5, directed=TRUE)
#V(g) means Vertex(g)
V(g)$color = "yellow"
V(g)$shape = "sphere"
```

The command for visualizing a network is the function *plot()*

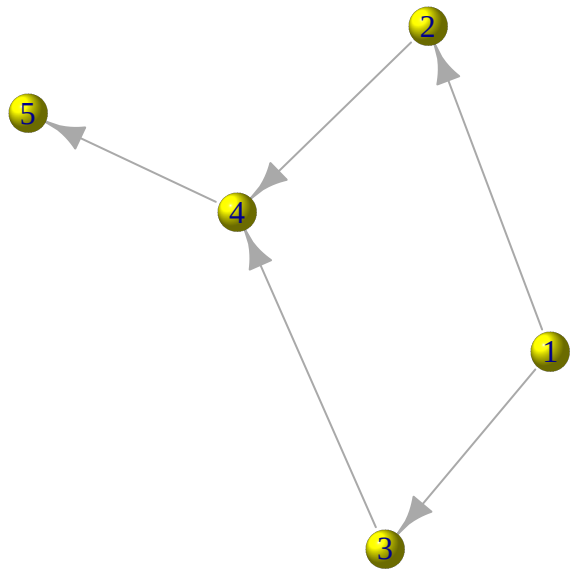
```
plot(g)
```



Add the next edges to the network 1->2, 1->3, 2->4, 3->4, 4->5.

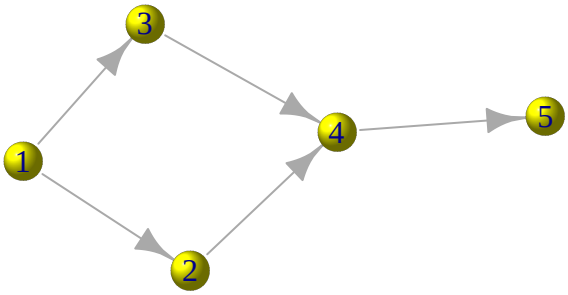
```
g <- add.edges(g, c(1,2, 1,3, 2,4, 3,4, 4,5))
```

```
plot(g)
```



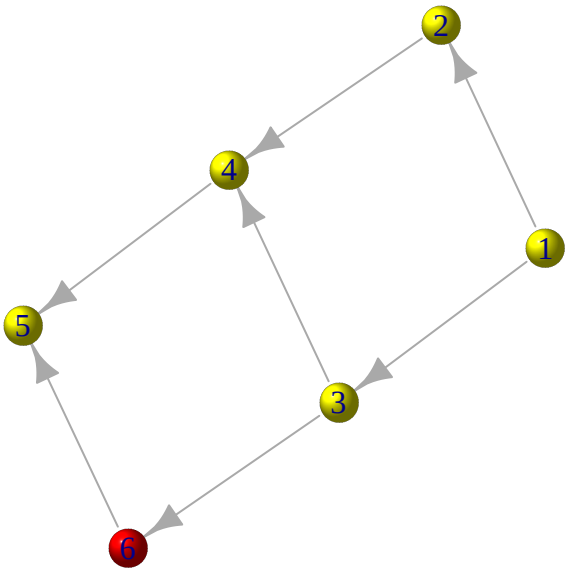
Add a red spherical vertex and add up to the network the next edges : 3->6, 6->5.

```
g <- add.vertices(g, 1, color="red", shape="sphere")
plot(g)
```




```
g <- add.edges(g, c(3,6, 6,5))
```

```
plot(g)
```



The class (vector, matrix,data.frame, network, etc) of an object in R is obtained using the command **class(*object*)**. Igraph provides its own class ("igraph")

```
class(g)
```

```
## [1] "igraph"
```

The structure of an object in R is obtained using the function **str(*object*)**.

Replace connection 1-> 3 with connection 3-> 1.

```
g <- delete.edges(g, c(2))
g <- add.edges(g, c(3,1))
```

Name the nodes consecutively with the letters A-F.

```
V(g)$name <- LETTERS[1:6]
V(g)
```

```
## + 6/6 vertices, named, from 9935286:
## [1] A B C D E F
```

```
E(g)
```

```
## + 7/7 edges from 9935286 (vertex names):
## [1] A->B B->D C->D D->E C->F F->E C->A
```

1 Degree

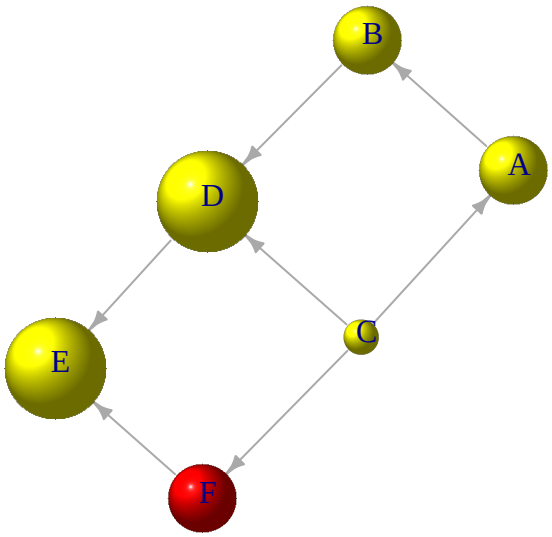
The command for the degree is the function **degree(*igraph_object*)**

```
degree(g)
```

```
## A B C D E F
## 2 2 3 3 2 2
```

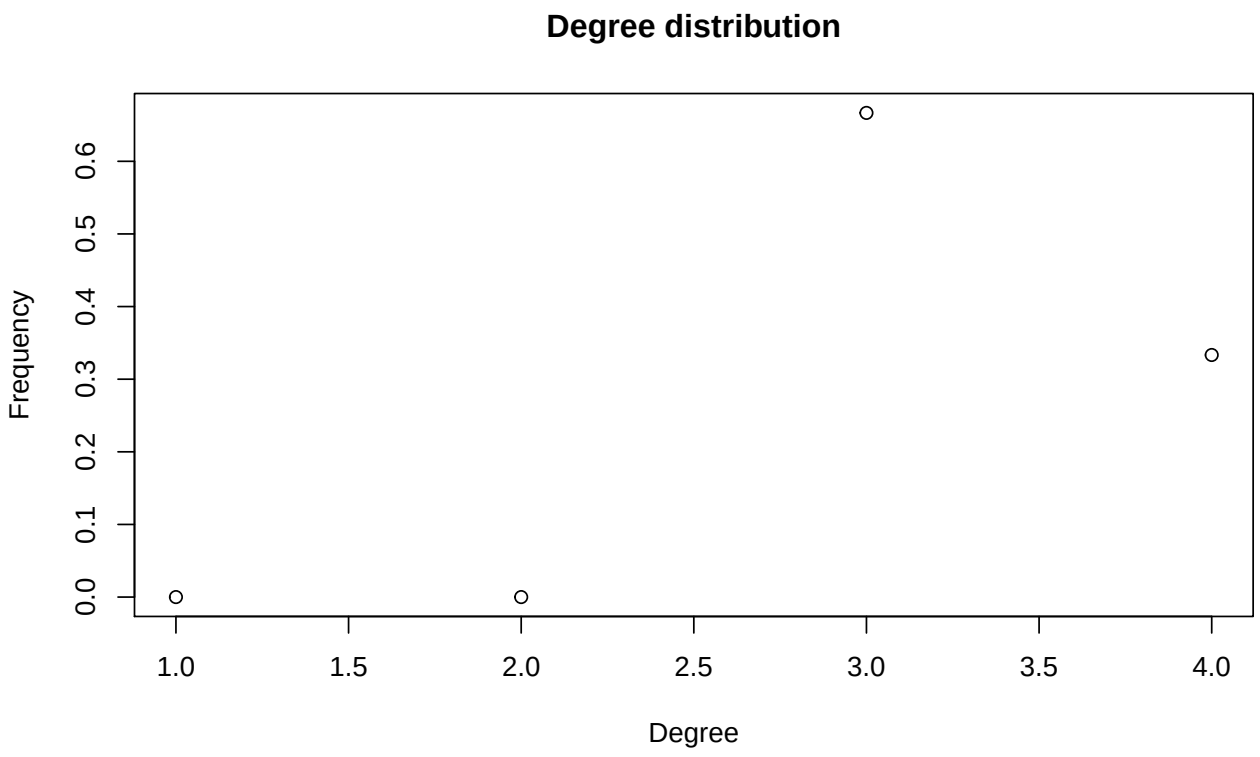
Plot the network in such way that that size of the nodes is propotional to the number of input connections

```
plot(g, layout=layout_nicely, vertex.size=degree(g, V(g), "in")*15+15,
     vertex.label.dist=0.5, edge.arrow.size=0.5)
```

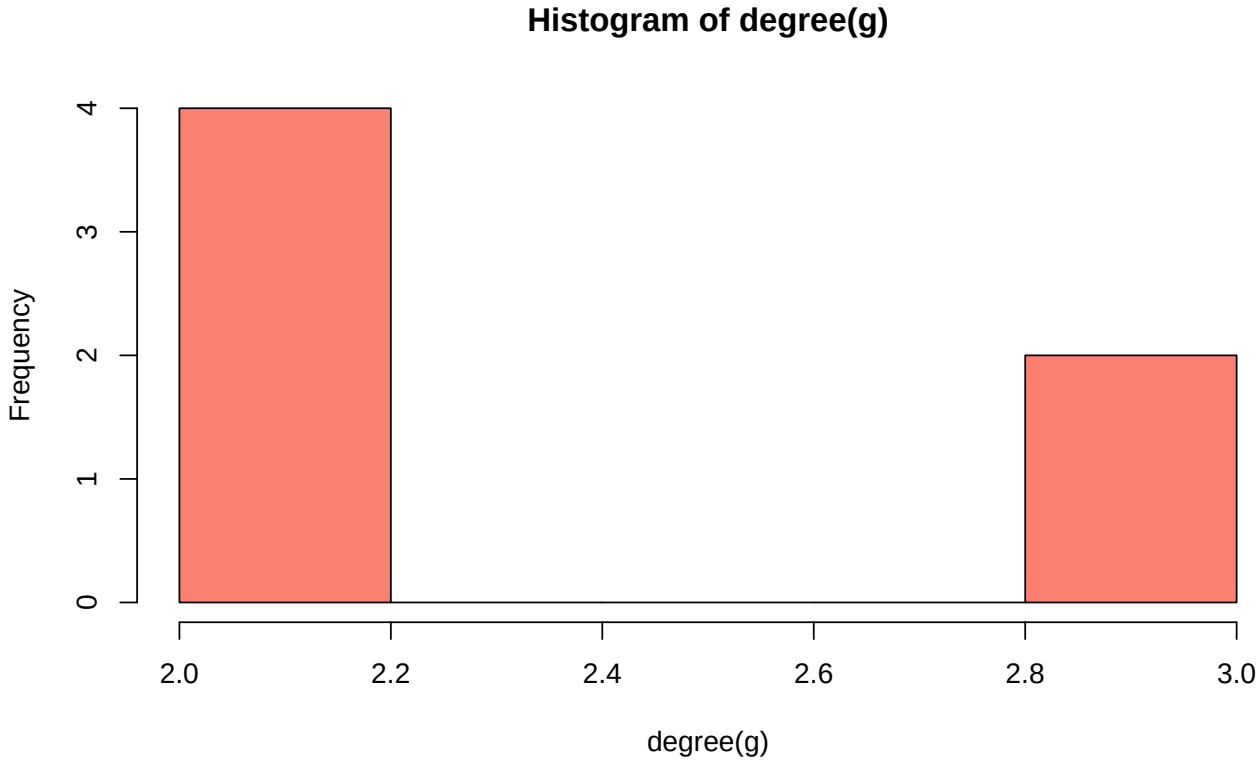


Find and plot the degree distribution

```
plot(degree_distribution(g), main="Degree distribution", xlab="Degree", ylab="Frequency")
```




```
hist(degree(g),col="salmon")
```



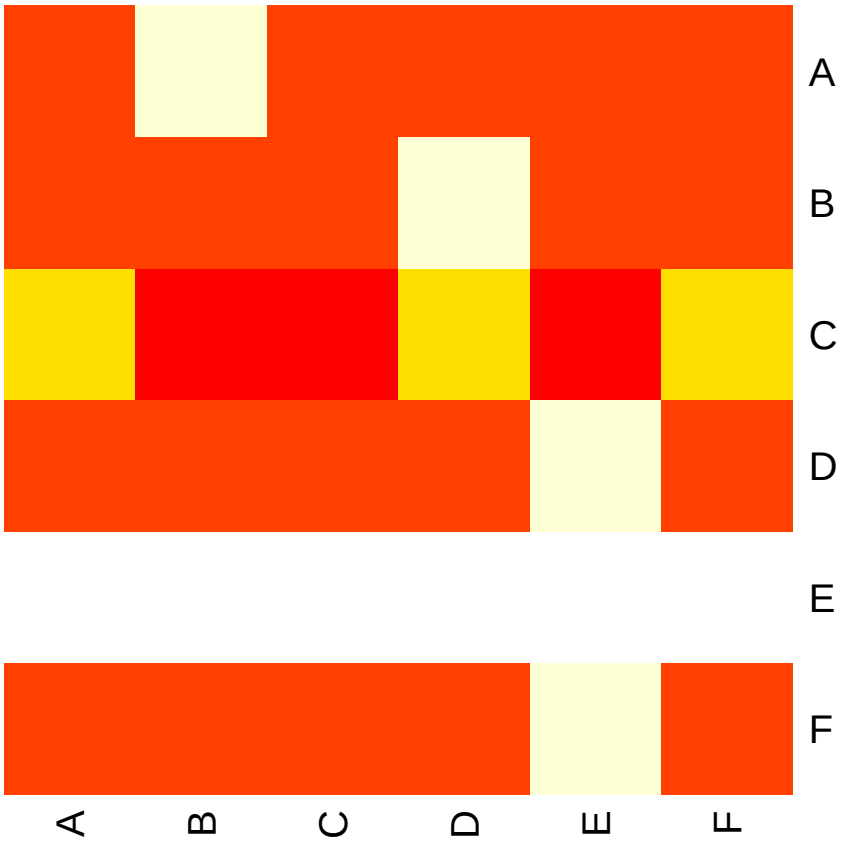
The adjacency matrix is obtained by **get.adjacency(*igraph_object*)**

```
adj_mat<-as.matrix(get.adjacency(g))
adj_mat
```

```
##   A B C D E F
## A 0 1 0 0 0 0
## B 0 0 0 1 0 0
## C 1 0 0 1 0 1
## D 0 0 0 0 1 0
## E 0 0 0 0 0 0
## F 0 0 0 0 1 0
```

Plot a heat map from this matrix

```
heatmap(adj_mat, Rowv=NA, Colv="Rowv")
```



Some special networks are available in igraph

- star (`make_star()`)
- ring (`make_ring()`)
- tree (`make_tree()`)
- lattice (`make_lattice()`)
- kautz (`make_kautz_graph()`)

Try to figure out what kind networks generate the previous commands before running them!

1.1 Data sets: edgelist

The first data set we are going to work with consists of two files, "Dataset1-Media-Example-NODES.csv" and "Dataset1-Media-Example-EDGES.csv" Download here .

```
complete_address<-"~/Dropbox/BioInfo2017/polnet2018/Data files/"
nodes <- read.csv(paste0(complete_address,"Dataset1-Media-Example-NODES.csv"), header=T, as.is=T)
```

```
links <- read.csv(paste0(complete_address,"Dataset1-Media-Example-EDGES.csv"), header=T, as.is=T)
```

Examine the data

```
head(nodes)
```

```
##    id               media media.type type.label audience.size
## 1 s01            NY Times          1  Newspaper            20
## 2 s02     Washington Post          1  Newspaper            25
## 3 s03 Wall Street Journal          1  Newspaper            30
## 4 s04           USA Today          1  Newspaper            32
## 5 s05            LA Times          1  Newspaper            20
## 6 s06       New York Post          1  Newspaper            50
```

```
head(links)
```

```
##   from  to      type weight
## 1  s01 s02 hyperlink     22
## 2  s01 s03 hyperlink     22
## 3  s01 s04 hyperlink     21
## 4  s01 s15   mention     20
## 5  s02 s01 hyperlink     23
## 6  s02 s03 hyperlink     21
```

or using View(object)

```
View(nodes)
View(links)
```

1.2 Creating an igraph object

Next we will convert the raw data to an igraph network object. To do that, we will use the graph_from_data_frame() function, which requires two data frames: nodes and vertices. nodes: describes the edges of the network. Its first two columns are the IDs of the source and the target node for each edge. links: describes the vertices which starts with a column of node IDs. Any following columns are interpreted as node attributes. This parameter can be omitted - in that case the igraph object will be generated based solely on the link structure in nodes.

```
net <- graph_from_data_frame(d=links, vertices=nodes, directed=T)
net
```

```
## IGRAPH fb802f8 DNW- 17 49 --
## + attr: name (v/c), media (v/c), media.type (v/n), type.label
## | (v/c), audience.size (v/n), type (e/c), weight (e/n)
## + edges from fb802f8 (vertex names):
##  [1] s01->s02 s01->s03 s01->s04 s01->s15 s02->s01 s02->s03 s02->s09
##  [8] s02->s10 s03->s01 s03->s04 s03->s05 s03->s08 s03->s10 s03->s11
## [15] s03->s12 s04->s03 s04->s06 s04->s11 s04->s12 s04->s17 s05->s01
## [22] s05->s02 s05->s09 s05->s15 s06->s06 s06->s16 s06->s17 s07->s03
## [29] s07->s08 s07->s10 s07->s14 s08->s03 s08->s07 s08->s09 s09->s10
## [36] s10->s03 s12->s06 s12->s13 s12->s14 s13->s12 s13->s17 s14->s11
## [43] s14->s13 s15->s01 s15->s04 s15->s06 s16->s06 s16->s17 s17->s04
```

The description of an igraph object starts with four letters:

```
D or U, for a directed or undirected graph
N for a named graph (where nodes have a name attribute)
```

```
W for a weighted graph (where edges have a weight attribute)
B for a bipartite (two-mode) graph (where nodes have a type attribute)
```

The two numbers that follow (17 49) refer to the number of nodes and edges in the graph. The description also lists node & edge attributes, for example:

```
(g/c) - graph-level character attribute
(v/c) - vertex-level character attribute
(e/n) - edge-level numeric attribute
```

We also have easy access to nodes, edges, and their attributes with:

```
E(net)       # The edges of the "net" object
```

```
## + 49/49 edges from fb802f8 (vertex names):
##  [1] s01->s02 s01->s03 s01->s04 s01->s15 s02->s01 s02->s03 s02->s09
##  [8] s02->s10 s03->s01 s03->s04 s03->s05 s03->s08 s03->s10 s03->s11
## [15] s03->s12 s04->s03 s04->s06 s04->s11 s04->s12 s04->s17 s05->s01
## [22] s05->s02 s05->s09 s05->s15 s06->s06 s06->s16 s06->s17 s07->s03
## [29] s07->s08 s07->s10 s07->s14 s08->s03 s08->s07 s08->s09 s09->s10
## [36] s10->s03 s12->s06 s12->s13 s12->s14 s13->s12 s13->s17 s14->s11
## [43] s14->s13 s15->s01 s15->s04 s15->s06 s16->s06 s16->s17 s17->s04
```

```
V(net)       # The vertices of the "net" object
```

```
## + 17/17 vertices, named, from fb802f8:
##  [1] s01 s02 s03 s04 s05 s06 s07 s08 s09 s10 s11 s12 s13 s14 s15 s16 s17
```

```
E(net)$type  # Edge attribute "type"
```

```
##  [1] "hyperlink" "hyperlink" "hyperlink" "mention"   "hyperlink"
##  [6] "hyperlink" "hyperlink" "hyperlink" "hyperlink" "hyperlink"
## [11] "hyperlink" "hyperlink" "mention"   "hyperlink" "hyperlink"
## [16] "hyperlink" "mention"   "mention"   "hyperlink" "mention"
## [21] "mention"   "hyperlink" "hyperlink" "mention"   "hyperlink"
## [26] "hyperlink" "mention"   "mention"   "mention"   "hyperlink"
## [31] "mention"   "hyperlink" "mention"   "mention"   "mention"
## [36] "hyperlink" "mention"   "hyperlink" "mention"   "hyperlink"
## [41] "mention"   "mention"   "mention"   "hyperlink" "hyperlink"
## [46] "hyperlink" "hyperlink" "mention"   "hyperlink"
```

```
V(net)$media # Vertex attribute "media"
```

```
##  [1] "NY Times"            "Washington Post"     "Wall Street Journal"
##  [4] "USA Today"           "LA Times"            "New York Post"
##  [7] "CNN"                 "MSNBC"               "FOX News"
## [10] "ABC"                 "BBC"                 "Yahoo News"
## [13] "Google News"         "Reuters.com"         "NYTimes.com"
## [16] "WashingtonPost.com"  "AOL.com"
```

```
# Find nodes and edges by attribute:
# (that returns oblects of type vertex sequence/edge sequence)
V(net)[media=="BBC"]
```

```
## + 1/17 vertex, named, from fb802f8:
## [1] s11
```

```
E(net)[type=="mention"]
```

```
## + 20/49 edges from fb802f8 (vertex names):
```

```
##  [1] s01->s15 s03->s10 s04->s06 s04->s11 s04->s17 s05->s01 s05->s15
##  [8] s06->s17 s07->s03 s07->s08 s07->s14 s08->s07 s08->s09 s09->s10
## [15] s12->s06 s12->s14 s13->s17 s14->s11 s14->s13 s16->s17
```

```
# You can also accesss the network matrix directly:
net[1,]
```

```
## s01 s02 s03 s04 s05 s06 s07 s08 s09 s10 s11 s12 s13 s14 s15 s16 s17
##   0  22  22  21   0   0   0   0   0   0   0   0   0   0  20   0   0
```

```
net[5,7]
```

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

It is also easy to extract an edge list or matrix back from the igraph network:

```
# Get an edge list or a matrix:
as_edgelist(net, names=T)
```

```
##       [,1]  [,2]
##  [1,] "s01" "s02"
##  [2,] "s01" "s03"
##  [3,] "s01" "s04"
##  [4,] "s01" "s15"
##  [5,] "s02" "s01"
##  [6,] "s02" "s03"
##  [7,] "s02" "s09"
##  [8,] "s02" "s10"
##  [9,] "s03" "s01"
## [10,] "s03" "s04"
## [11,] "s03" "s05"
## [12,] "s03" "s08"
## [13,] "s03" "s10"
## [14,] "s03" "s11"
## [15,] "s03" "s12"
## [16,] "s04" "s03"
## [17,] "s04" "s06"
## [18,] "s04" "s11"
## [19,] "s04" "s12"
## [20,] "s04" "s17"
## [21,] "s05" "s01"
## [22,] "s05" "s02"
## [23,] "s05" "s09"
## [24,] "s05" "s15"
## [25,] "s06" "s06"
## [26,] "s06" "s16"
## [27,] "s06" "s17"
## [28,] "s07" "s03"
## [29,] "s07" "s08"
## [30,] "s07" "s10"
## [31,] "s07" "s14"
## [32,] "s08" "s03"
## [33,] "s08" "s07"
## [34,] "s08" "s09"
## [35,] "s09" "s10"
## [36,] "s10" "s03"
## [37,] "s12" "s06"
```

```
## [38,] "s12" "s13"
## [39,] "s12" "s14"
## [40,] "s13" "s12"
## [41,] "s13" "s17"
## [42,] "s14" "s11"
## [43,] "s14" "s13"
## [44,] "s15" "s01"
## [45,] "s15" "s04"
## [46,] "s15" "s06"
## [47,] "s16" "s06"
## [48,] "s16" "s17"
## [49,] "s17" "s04"
```

```
as_adjacency_matrix(net, attr="weight")
```

```
## 17 x 17 sparse Matrix of class "dgCMatrix"
##    [[ suppressing 17 column names 's01', 's02', 's03' ... ]]
##
## s01  . 22 22 21 .  .  .  .  .  .  .  .  .  . 20  .  .
## s02 23  . 21  . .  .  .  .  1  5  .  .  .  .  .  .  .
## s03 21  .  . 22 1  .  .  4  .  2  1  1  .  .  .  .  .
## s04  .  . 23  . .  1  .  .  .  . 22  3  .  .  .  .  2
## s05  1 21  .  . .  .  .  .  2  .  .  .  .  . 21  .  .
## s06  .  .  .  . .  1  .  .  .  .  .  .  .  .  . 21 21
## s07  .  .  1  . .  .  . 22  . 21  .  .  .  4  .  .  .
## s08  .  .  2  . .  . 21  . 23  .  .  .  .  .  .  .  .
## s09  .  .  .  . .  .  .  .  . 21  .  .  .  .  .  .  .
## s10  .  .  2  . .  .  .  .  .  .  .  .  .  .  .  .  .
## s11  .  .  .  . .  .  .  .  .  .  .  .  .  .  .  .  .
## s12  .  .  .  . .  2  .  .  .  .  .  . 22 22  .  .  .
## s13  .  .  .  . .  .  .  .  .  .  . 21  .  .  .  .  1
## s14  .  .  .  . .  .  .  .  .  .  1  . 21  .  .  .  .
## s15 22  .  .  1 .  4  .  .  .  .  .  .  .  .  .  .  .
## s16  .  .  .  . . 23  .  .  .  .  .  .  .  .  .  . 21
## s17  .  .  .  4 .  .  .  .  .  .  .  .  .  .  .  .  .
```

```
# Or data frames describing nodes and edges:
as_data_frame(net, what="edges")
```

```
##    from  to      type weight
## 1   s01 s02 hyperlink     22
## 2   s01 s03 hyperlink     22
## 3   s01 s04 hyperlink     21
## 4   s01 s15   mention     20
## 5   s02 s01 hyperlink     23
## 6   s02 s03 hyperlink     21
## 7   s02 s09 hyperlink      1
## 8   s02 s10 hyperlink      5
## 9   s03 s01 hyperlink     21
## 10  s03 s04 hyperlink     22
## 11  s03 s05 hyperlink      1
## 12  s03 s08 hyperlink      4
## 13  s03 s10   mention      2
## 14  s03 s11 hyperlink      1
```

```
## 15  s03 s12 hyperlink      1
## 16  s04 s03 hyperlink     23
## 17  s04 s06   mention      1
## 18  s04 s11   mention     22
## 19  s04 s12 hyperlink      3
## 20  s04 s17   mention      2
## 21  s05 s01   mention      1
## 22  s05 s02 hyperlink     21
## 23  s05 s09 hyperlink      2
## 24  s05 s15   mention     21
## 25  s06 s06 hyperlink      1
## 26  s06 s16 hyperlink     21
## 27  s06 s17   mention     21
## 28  s07 s03   mention      1
## 29  s07 s08   mention     22
## 30  s07 s10 hyperlink     21
## 31  s07 s14   mention      4
## 32  s08 s03 hyperlink      2
## 33  s08 s07   mention     21
## 34  s08 s09   mention     23
## 35  s09 s10   mention     21
## 36  s10 s03 hyperlink      2
## 37  s12 s06   mention      2
## 38  s12 s13 hyperlink     22
## 39  s12 s14   mention     22
## 40  s13 s12 hyperlink     21
## 41  s13 s17   mention      1
## 42  s14 s11   mention      1
## 43  s14 s13   mention     21
## 44  s15 s01 hyperlink     22
## 45  s15 s04 hyperlink      1
## 46  s15 s06 hyperlink      4
## 47  s16 s06 hyperlink     23
## 48  s16 s17   mention     21
## 49  s17 s04 hyperlink      4
```

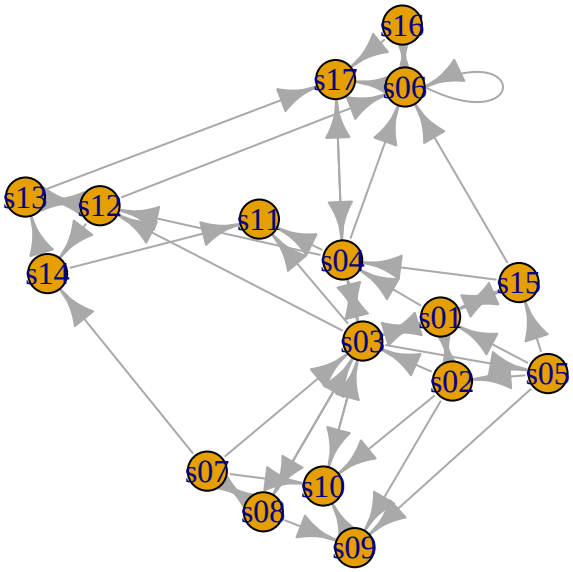
```
as_data_frame(net, what="vertices")
```

```
##     name               media media.type type.label audience.size
## s01  s01            NY Times          1  Newspaper            20
## s02  s02     Washington Post          1  Newspaper            25
## s03  s03 Wall Street Journal          1  Newspaper            30
## s04  s04           USA Today          1  Newspaper            32
## s05  s05            LA Times          1  Newspaper            20
## s06  s06       New York Post          1  Newspaper            50
## s07  s07                 CNN          2         TV            56
## s08  s08               MSNBC          2         TV            34
## s09  s09           FOX News           2         TV            60
## s10  s10                 ABC          2         TV            23
## s11  s11                 BBC          2         TV            34
## s12  s12          Yahoo News          3     Online            33
## s13  s13         Google News          3     Online            23
## s14  s14         Reuters.com          3     Online            12
## s15  s15         NYTimes.com          3     Online            24
## s16  s16  WashingtonPost.com          3     Online            28
```

```
## s17  s17              AOL.com          3      Online             33
```

Now that we have our igraph network object, let's make a first attempt to plot it.

```
plot(net) # not a pretty picture!
```

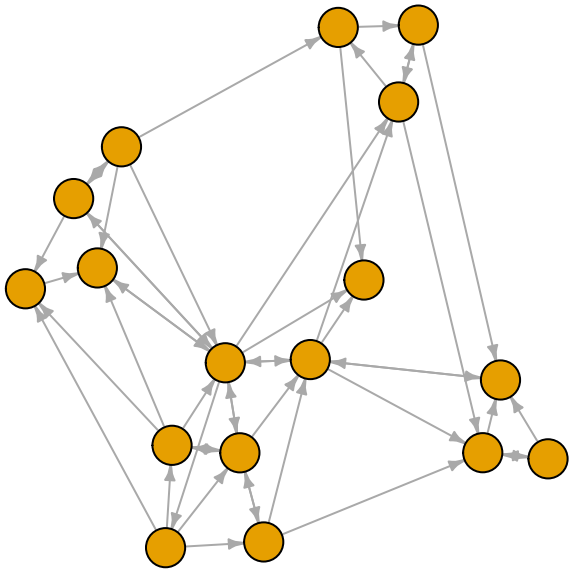


That doesn't look very good. Let's start fixing things by removing the loops in the graph.

```
net <- simplify(net, remove.multiple = F, remove.loops = T)
```

Let's and reduce the arrow size and remove the labels (we do that by setting them to NA):

```
plot(net, edge.arrow.size=.4,vertex.label=NA)
```



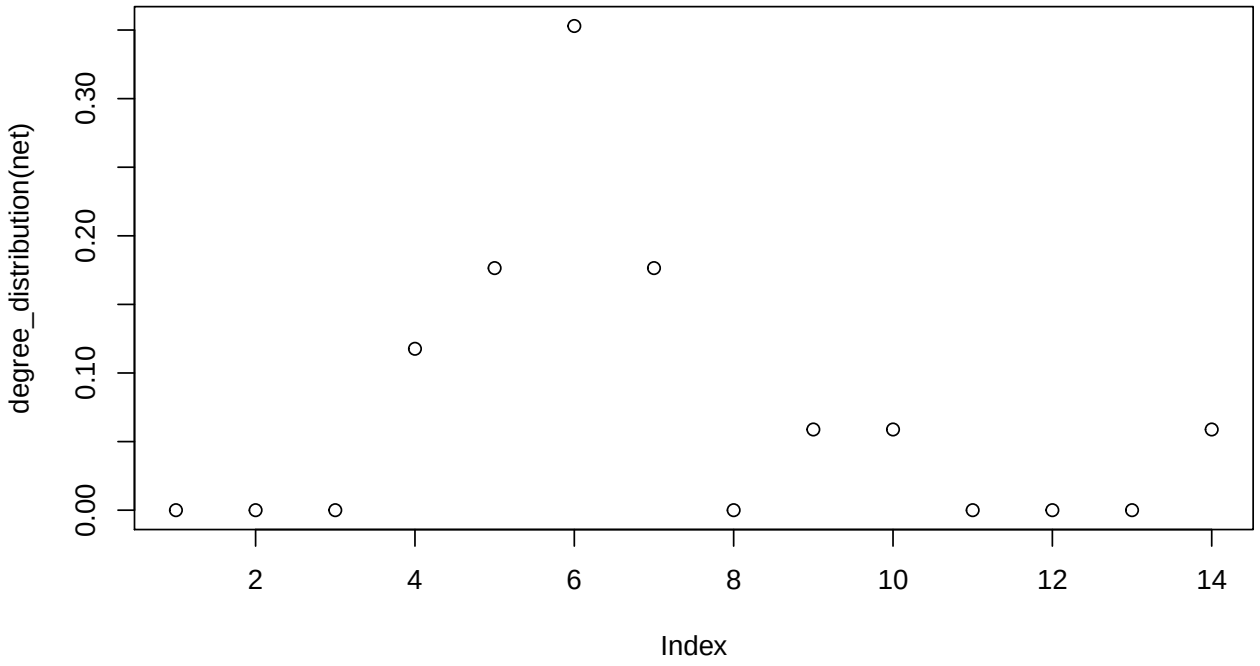
1.2.1 Topological properties

```
degree(net)
```

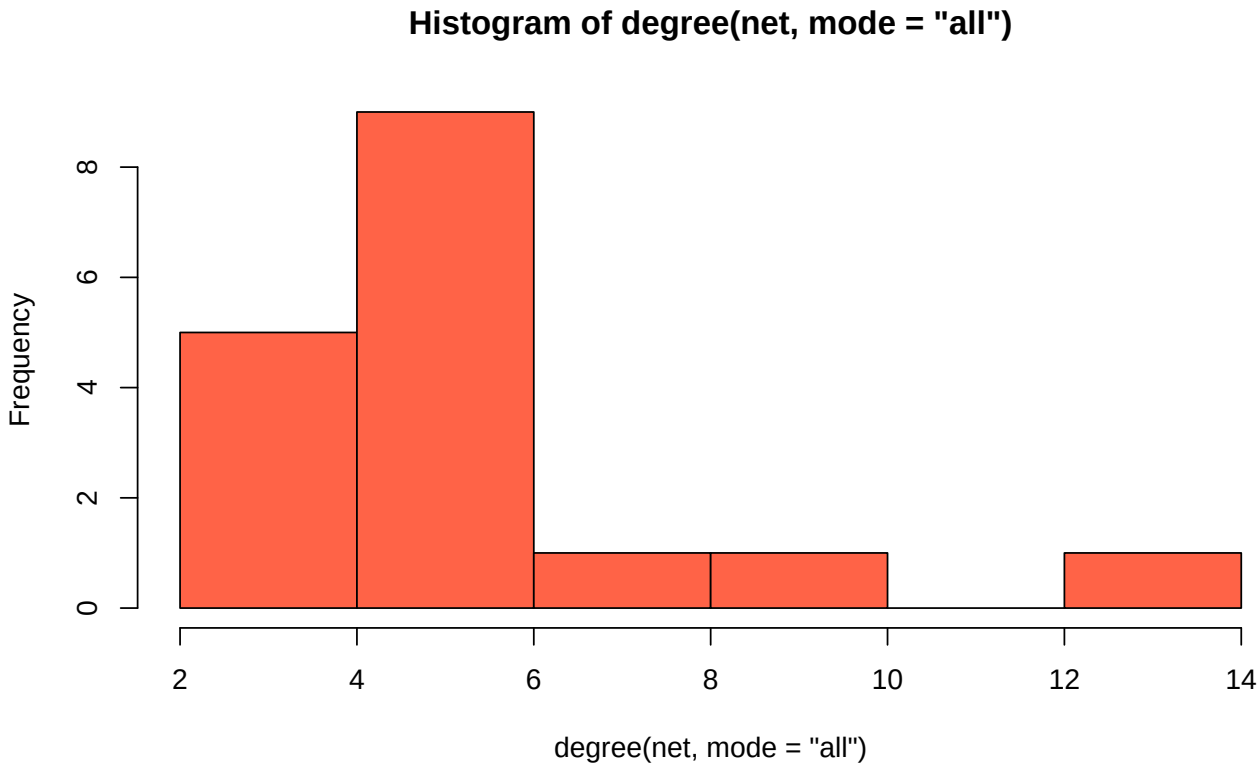
```
## s01 s02 s03 s04 s05 s06 s07 s08 s09 s10 s11 s12 s13 s14 s15 s16 s17
```

```
##  8  6 13  9  5  6  5  5  4  5  3  6  4  4  5  3  5
```

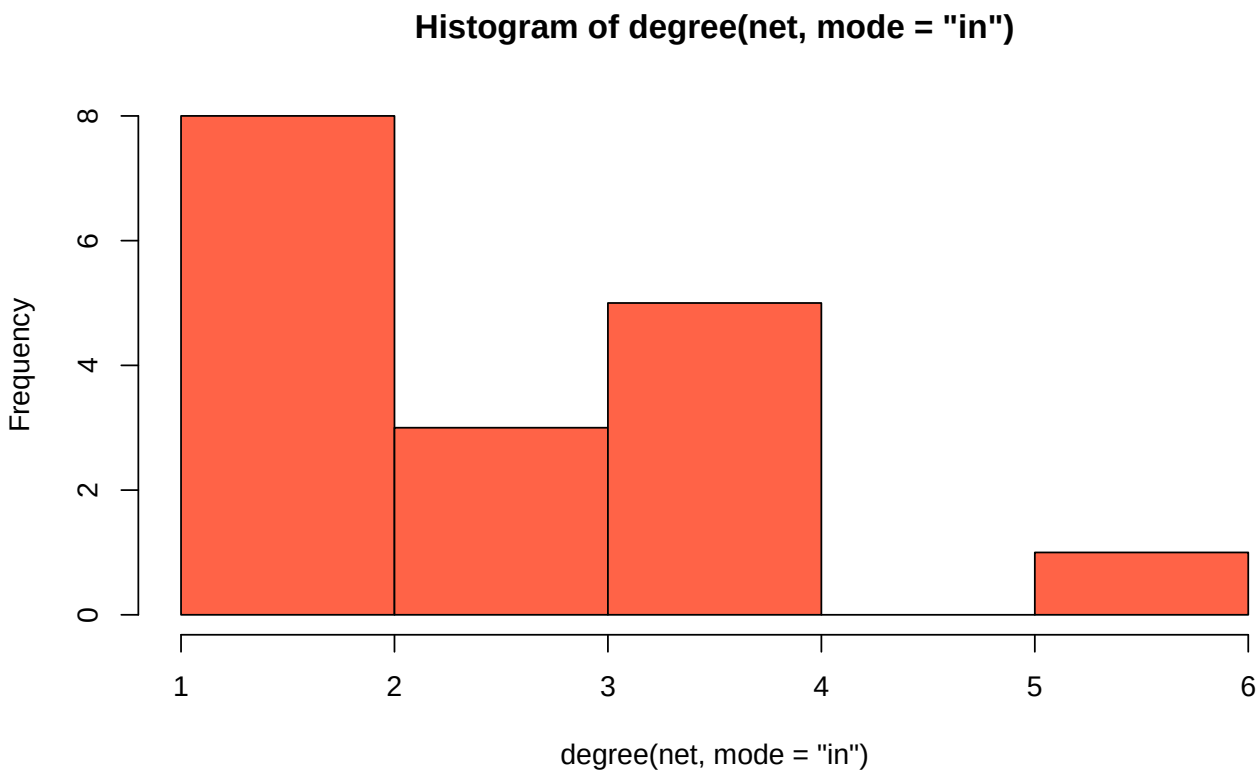
```
plot(degree_distribution(net))
```



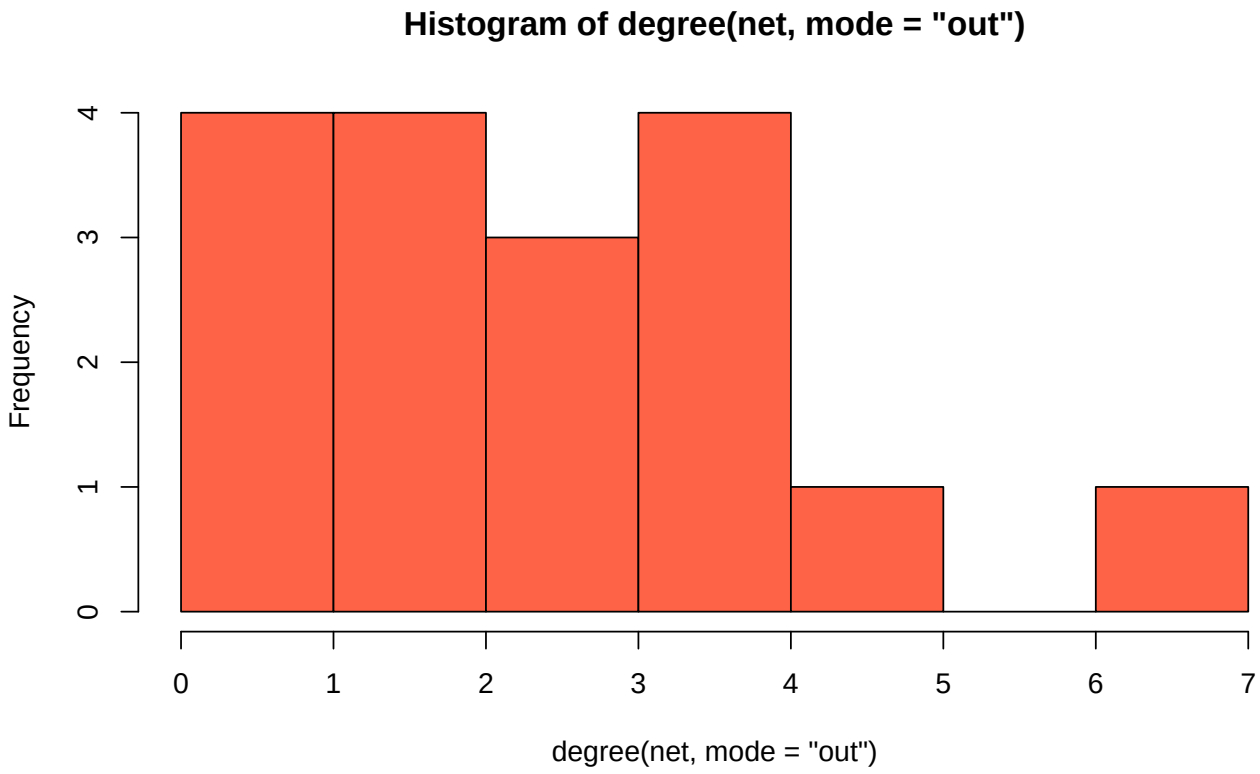

```
hist(degree(net,mode="all"),col = "tomato")
```




```
hist(degree(net,mode="in"),col = "tomato")
```




```
hist(degree(net,mode="out"),col = "tomato")
```



2 Plotting networks with igraph

2.1 Plotting parameters

Plotting with igraph: the network plots have a wide set of parameters you can set. Those include node options (starting with vertex.) and edge options (starting with edge.). A list of selected options is included below, but you can also check out ?igraph.plotting for more information.

The igraph plotting parameters include (among others):

NODES	
vertex.color	Node color (Use colors() to determine which colors are available in R
vertex.frame.color	Node border color
vertex.shape	One of “none”, “circle”, “square”, “csquare”, “rectangle”, “crectangle”, “vrectangle”, “pie”, “raster”, or “sphere”
vertex.size	Size of the node (default is 15)
vertex.size2	The second size of the node (e.g. for a rectangle)
vertex.label	Character vector used to label the nodes
vertex.label.family	Font family of the label (e.g.“Times”, “Helvetica”)
vertex.label.font	Font: 1 plain, 2 bold, 3, italic, 4 bold italic, 5 symbol
vertex.label.cex	Font size (multiplication factor, device-dependent)
vertex.label.dist	Distance between the label and the vertex
vertex.label.degree	The position of the label in relation to the vertex, where 0 is right, “pi” is left, “pi/2” is below, and “-pi/2” is above

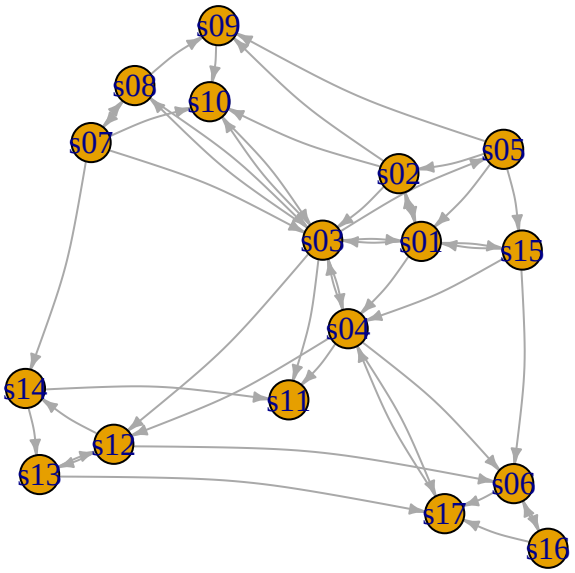
EDGES	
edge.color	Edge color
edge.width	Edge width, defaults to 1
edge.arrow.size	Arrow size, defaults to 1
edge.arrow.width	Arrow width, defaults to 1
edge.lty	Line type, could be 0 or “blank”, 1 or “solid”, 2 or “dashed”, 3 or “dotted”, 4 or “dotdash”, 5 or “longdash”, 6 or “twodash”
edge.label	Character vector used to label edges
edge.label.family	Font family of the label (e.g.“Times”, “Helvetica”)
edge.label.font	Font: 1 plain, 2 bold, 3, italic, 4 bold italic, 5 symbol
edge.label.cex	Font size for edge labels
edge.curved	Edge curvature, range 0-1 (FALSE sets it to 0, TRUE to 0.5)
arrow.mode	Vector specifying whether edges should have arrows, possible values: 0 no arrow, 1 back, 2 forward, 3 both

OTHER

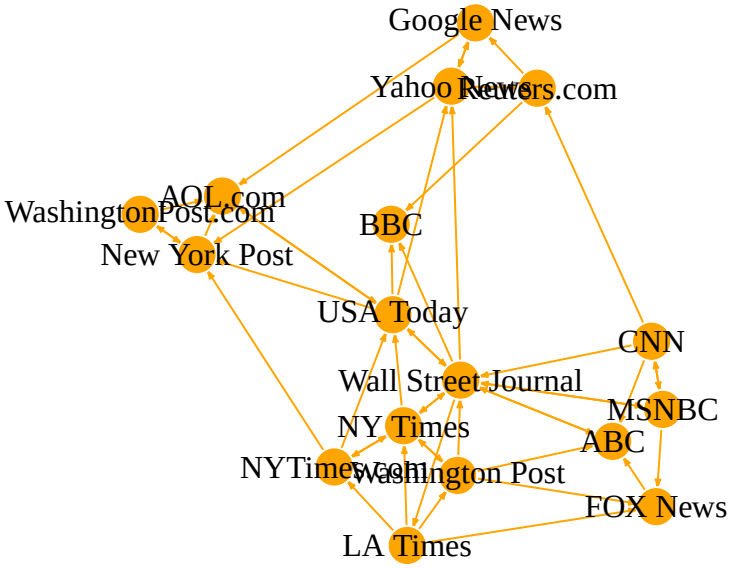
```
margin   Empty space margins around the plot, vector with length 4
frame    if TRUE, the plot will be framed
main     If set, adds a title to the plot
sub      If set, adds a subtitle to the plot
asp       Numeric, the aspect ratio of a plot (y/x).
palette      A color palette to use for vertex color
rescale      Whether to rescale coordinates to [-1,1]. Default is TRUE.
```

We can set the node & edge options in two ways - the first one is to specify them in the plot() function, as we are doing below.

```
# Plot with curved edges (edge.curved=.1) and reduce arrow size:
# Note that using curved edges will allow you to see multiple links
# between two nodes (e.g. links going in either direction, or multiplex links)
plot(net, edge.arrow.size=.4, edge.curved=.1)
```

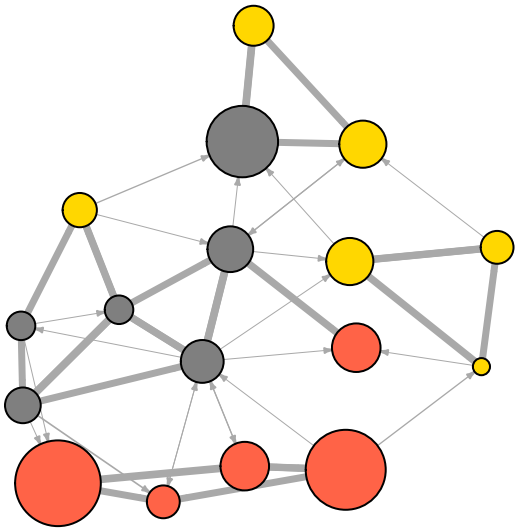



```
# Set edge color to light gray, the node & border color to orange
# Replace the vertex label with the node names stored in "media"
plot(net, edge.arrow.size=.2, edge.color="orange",
     vertex.color="orange", vertex.frame.color="#ffffff",
     vertex.label=V(net)$media, vertex.label.color="black")
```



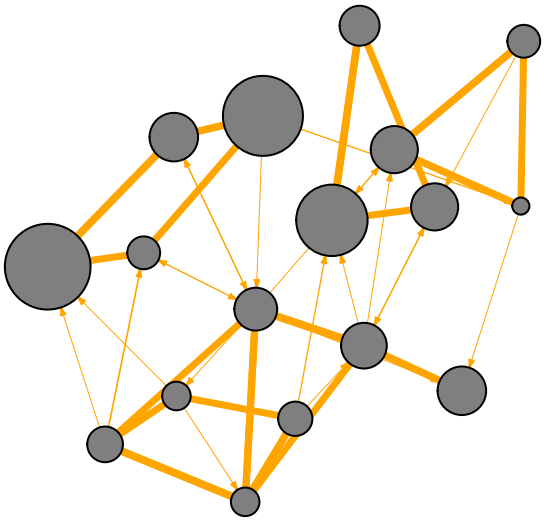
The second way to set attributes is to add them to the igraph object. Let's say we want to color our network nodes based on type of media, and size them based on degree centrality (more links -> larger node) We will also change the width of the edges based on their weight.

```
# Generate colors based on media type:
colrs <- c("gray50", "tomato", "gold")
V(net)$color <- colrs[V(net)$media.type]
# Compute node degrees (#links) and use that to set node size:
deg <- degree(net, mode="all")
V(net)$size <- deg*3
# We could also use the audience size value:
V(net)$size <- V(net)$audience.size*0.6

# The labels are currently node IDs.
# Setting them to NA will render no labels:
V(net)$label <- NA

# Set edge width based on weight:
E(net)$width <- E(net)$weight/6

#change arrow size and edge color:
E(net)$arrow.size <- .2
E(net)$edge.color <- "gray80"

# We can even set the network layout:
graph_attr(net, "layout") <- layout_with_lgl
plot(net)
```



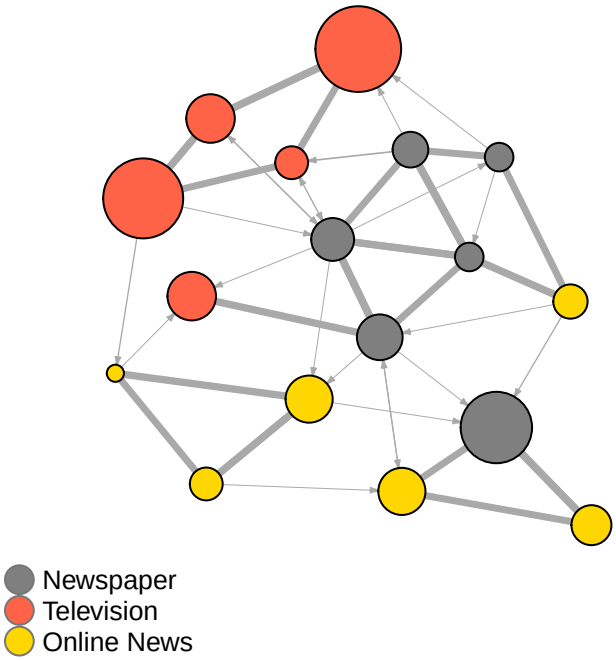
We can also override the attributes explicitly in the plot:

```
plot(net, edge.color="orange", vertex.color="gray50")
```



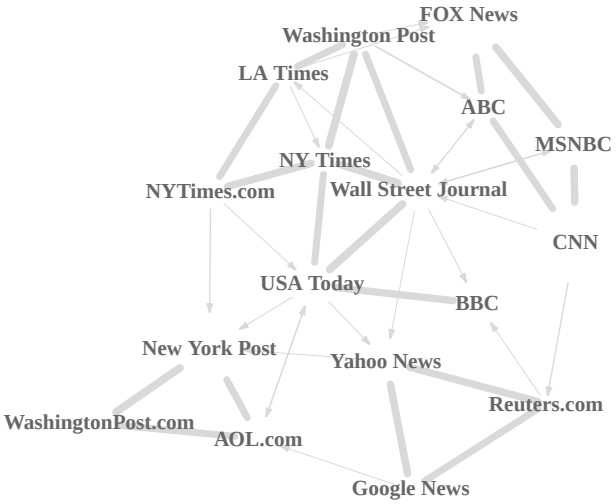
It helps to add a legend explaining the meaning of the colors we used:

```
plot(net)
legend(x=-1.5, y=-1.1, c("Newspaper","Television", "Online News"), pch=21,
       col="#777777", pt.bg=colrs, pt.cex=2, cex=.8, bty="n", ncol=1)
```



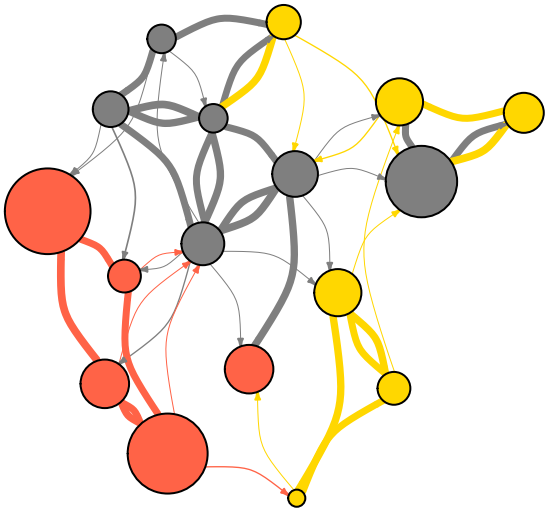
Sometimes, especially with semantic networks, we may be interested in plotting only the labels of the nodes:

```
plot(net, vertex.shape="none", vertex.label=V(net)$media,
     vertex.label.font=2, vertex.label.color="gray40",
     vertex.label.cex=.7, edge.color="gray85")
```



Let's color the edges of the graph based on their source node color. We can get the starting node for each edge with the ends() igraph function. It returns the start and end vertex for edges listed in the es parameter. The names parameter control whether the function returns edge names or IDs.

```
edge.start <- ends(net, es=E(net), names=F)[,1]
edge.col <- V(net)$color[edge.start]

plot(net, edge.color=edge.col, edge.curved=.4)
```



2.1.1 Exercises

1. Calculate the degree,average degree, plot the degree distribution, obtain the adjacency matrix from the next networks

```
g1<-barabasi.game(100,directed = FALSE)
g2<-random.graph.game(100,0.5)
g3<-sample_smallworld(1,100,p=0.2,nei=3)
```

2. Identify the vertex whose degree is the maximum
3. Find the 10 most connected vertices.
4. Build a network from vertex = people form this classroom, rule of connection= two people are connected if they were born in a neighbor state or in the same state. Use colors, names and sizes that reflects the totpological properties of the network.