DSI Summer Workshops Series

June 14, 2018

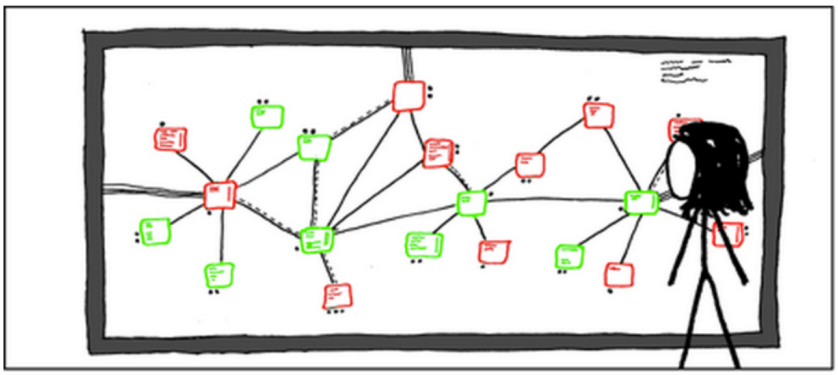
Peggy Lindner
Center for Advanced Computing & Data Science (CACDS)
Data Science Institute (DSI)
University of Houston
plindner@uh.edu

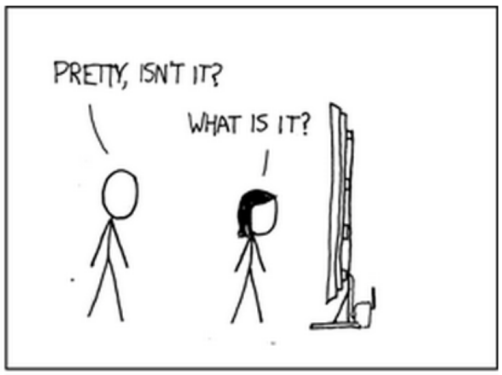
Please make sure you have Jupyterhub running with support for R and all the required packages installed. Data for this and other tutorials can be found in the github repsoitory for the Summer 2018 DSI Workshops
https://github.com/peggylind/Materials_Summer2018
(https://github.com/peggylind/Materials_Summer2018)

Network Graphs

Basis understanding of Network Analyis using R

Intro





net·work

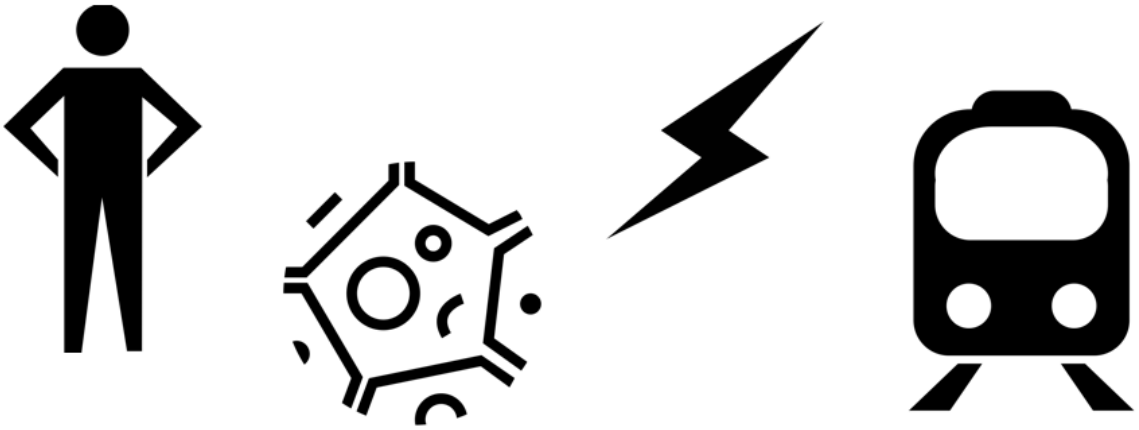
a group or system of interconnected people or things

a·nal·y·sis

detailed examination of the elements or structure of something, typically as a basis for discussion or interpretation

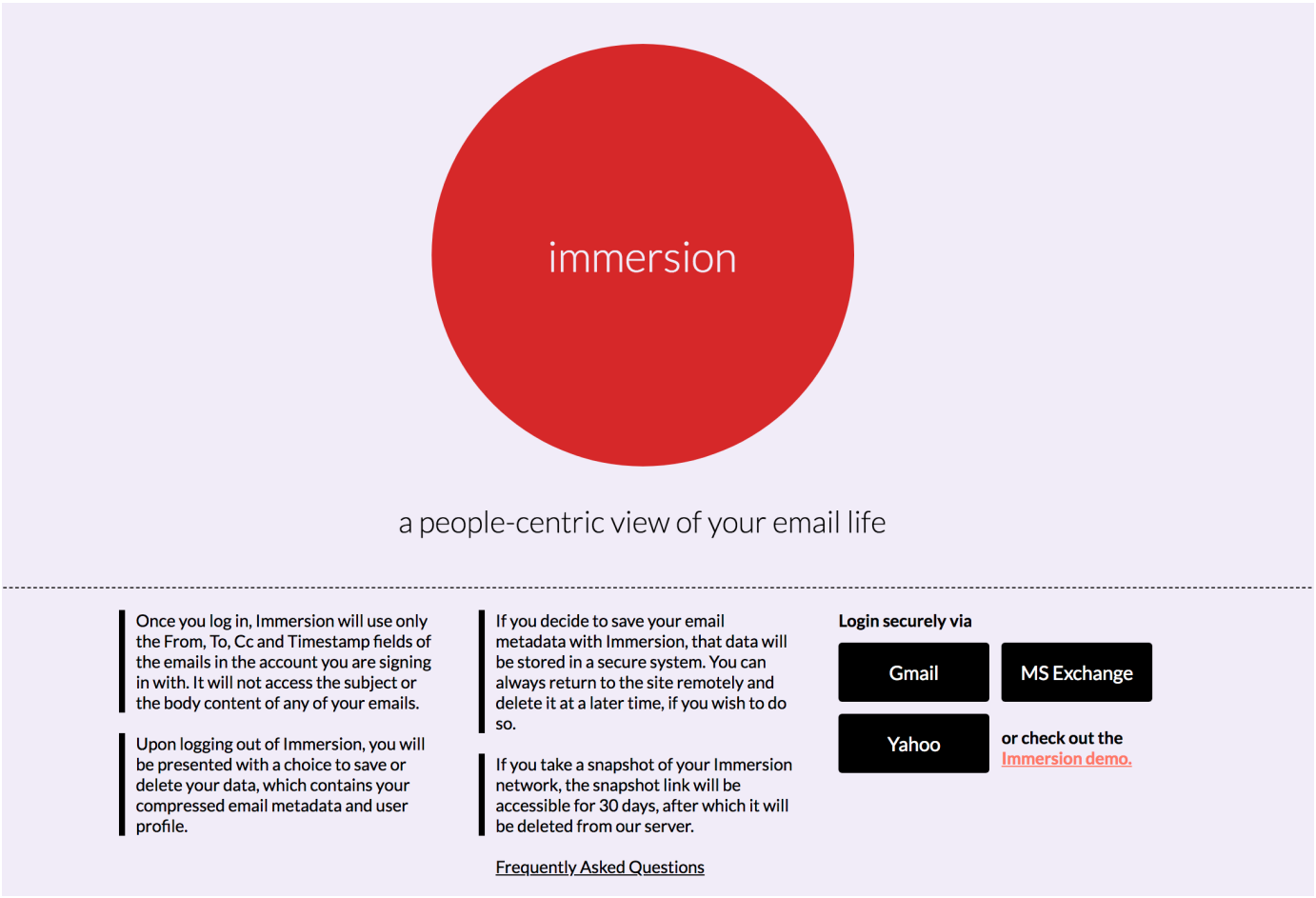
net·work a·nal·y·sis

Study of the structure of relationships between things and across things. Things include but are not limited to people, neural cells, power grids, and transportation hubs.



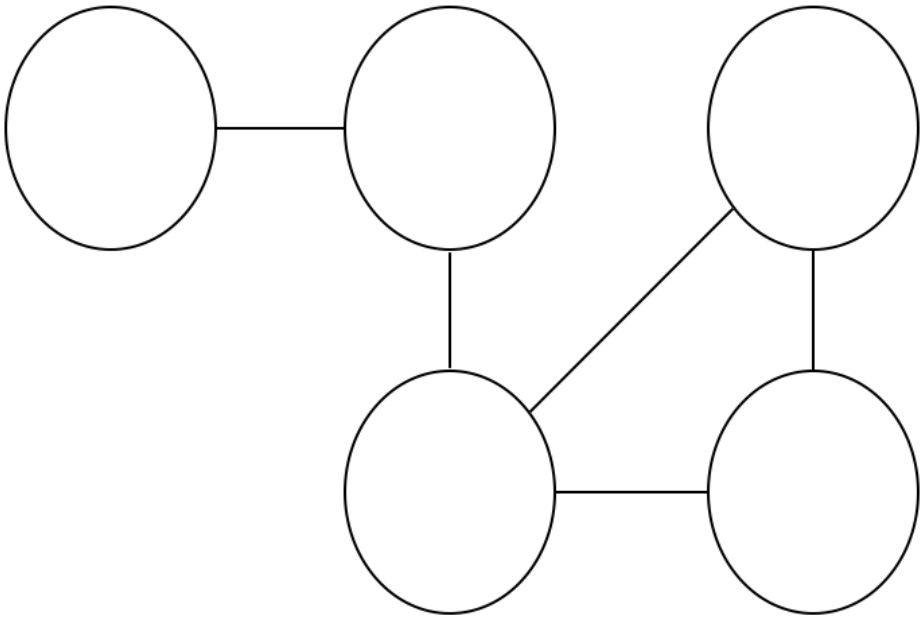
questions

What does the network look like?
How connected is the network?
Which are the key entities?
Which are key subgroups?
How does network structure affect function?

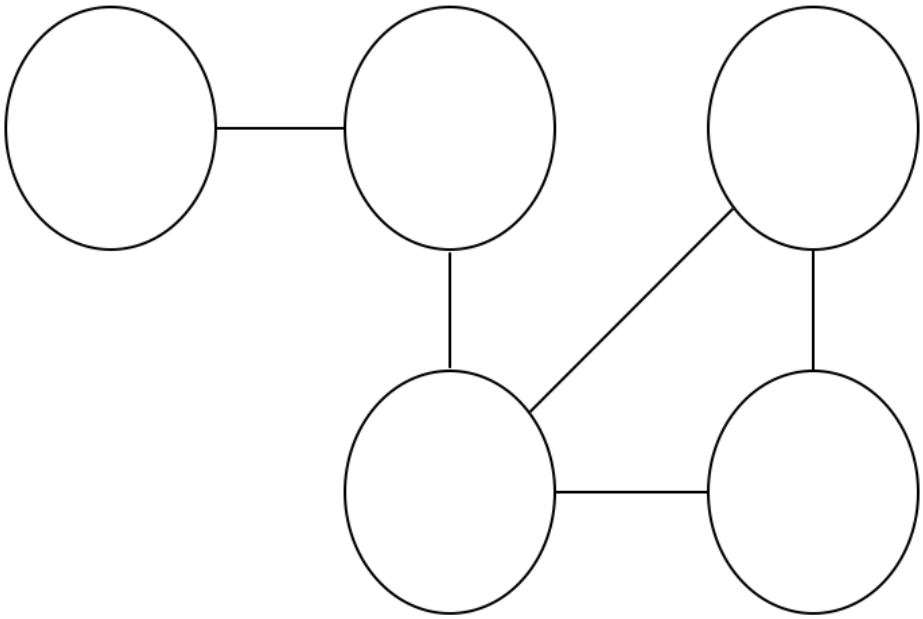


https://immersion.media.mit.edu/demo (https://immersion.media.mit.edu/demo)

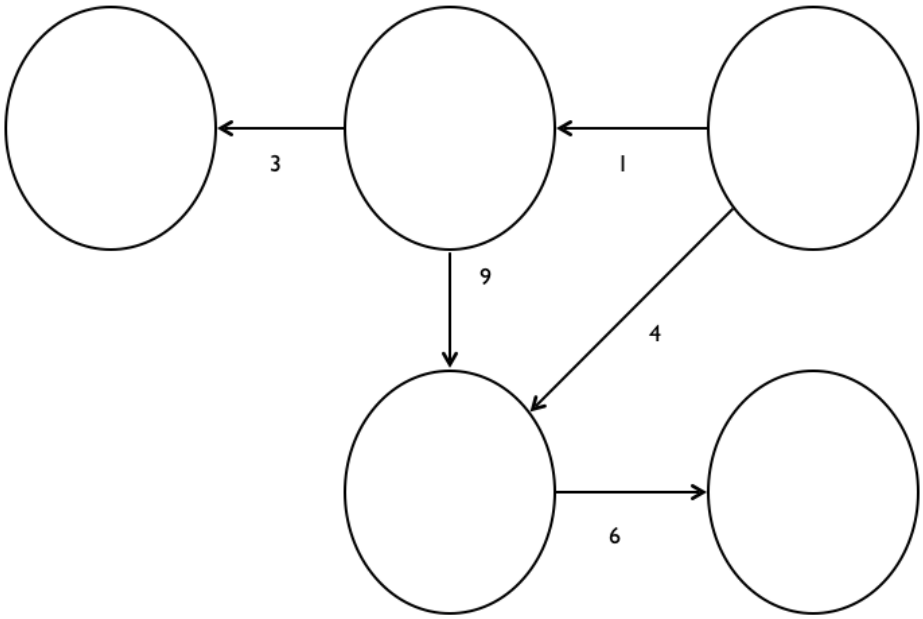
Graphs



nodes edges undirected

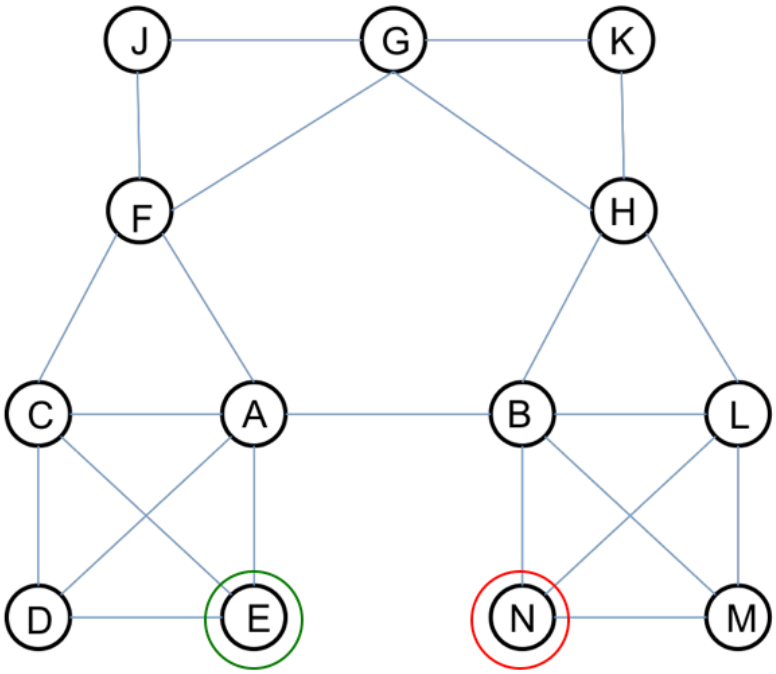


directed weighted



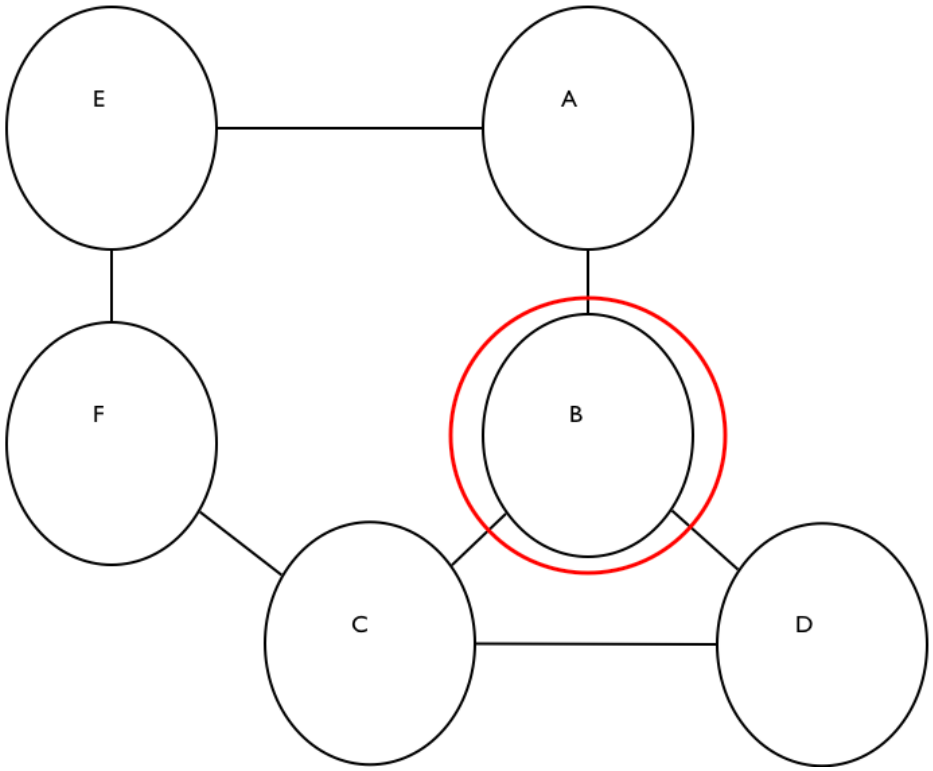
path

a sequence of nodes, where each node in the sequence is connected by an edge



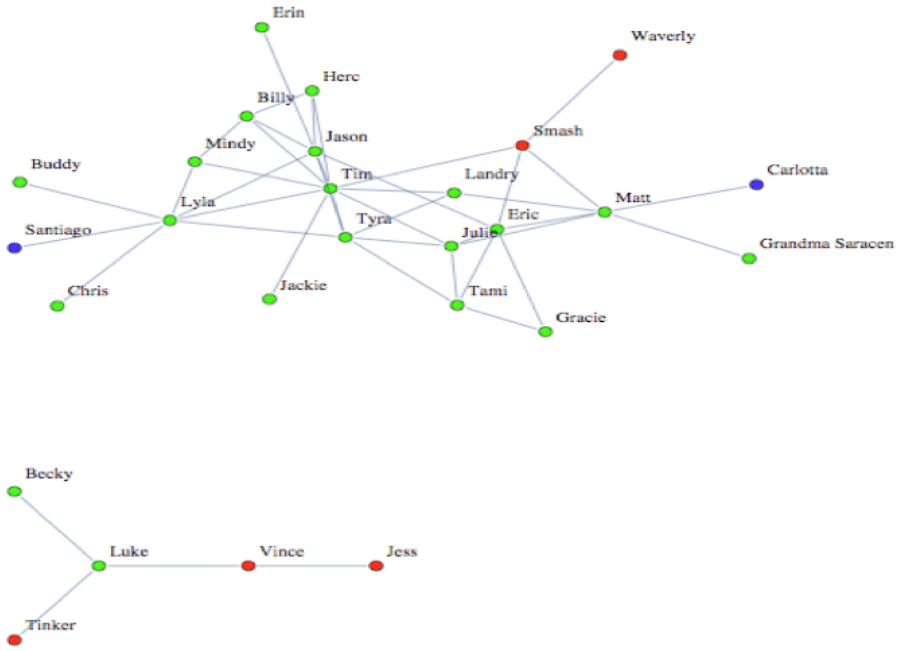
pivotal node

node is pivotal if it lies on shortest path between two pairs of nodes



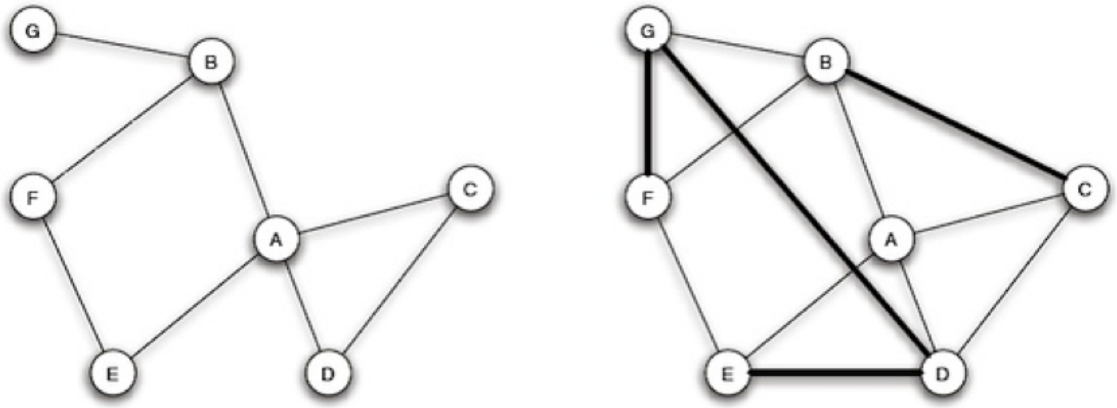
component

a subset of connected nodes



clustering coefficient

the probability that two randomly selected friends of a node are friends with each other

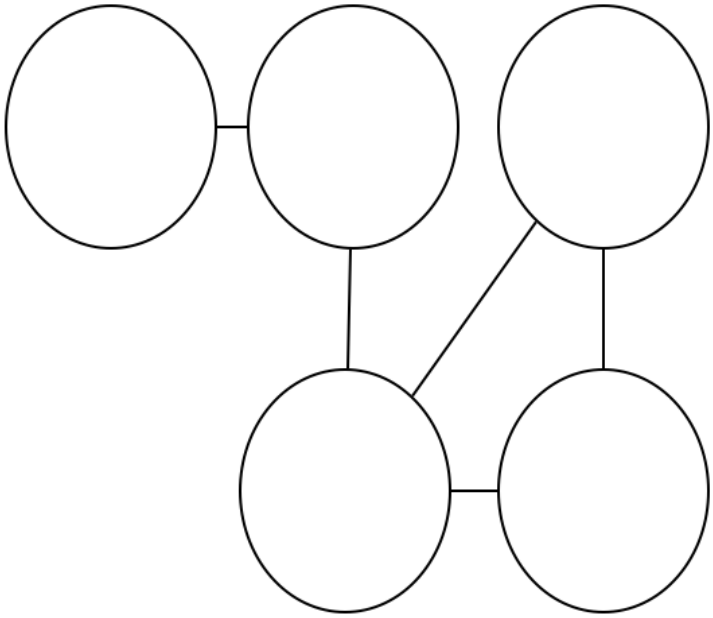


(a) *Before new edges form.*

(b) *After new edges form.*

degree centrality

measure of how connected a node is



degree – no. connections

in-degree – inbound connections

out-degree – outbound connections

basic process

Specify question
Find or create relational data
Specify nodes & edges
Explore & Analyze relational data
Interpret results
Repeat

Data format, size, and preparation

In this tutorial, we will work primarily with two small example data sets. Both contain data about media organizations. One involves a network of hyperlinks and mentions among news sources. The second is a network of links between media venues and consumers. While the example data used here is small, many of the ideas behind the visualizations we will generate apply to medium and large-scale networks. This is also the reason why we will rarely use certain visual properties such as the shape of the node symbols: those are impossible to distinguish in larger graph maps. In fact, when drawing very big networks we may even want to hide the network edges, and focus on identifying and visualizing communities of nodes. At this point, the size of the networks you can visualize in R is limited mainly by the RAM of your machine. One thing to emphasize though is that in many cases, visualizing larger networks as giant hairballs is less helpful than providing charts that show key characteristics of the graph.

First we need load some packages that we need:

In [4]:

```
library(igraph)
library(network)
library(sna)
library(ndtv)
```

```
Loading required package: networkDynamic

networkDynamic: version 0.9.0, created on 2016-01-1
2
Copyright (c) 2016, Carter T. Butts, University of
California -- Irvine
                    Ayn Leslie-Cook, University of
Washington
                    Pavel N. Krivitsky, University
of Wollongong
                    Skye Bender-deMoll, University
of Washington
                    with contributions from
                    Zack Almquist, University of Ca
lifornia -- Irvine
                    David R. Hunter, Penn State Uni
versity
                    Li Wang
                    Kirk Li, University of Washingt
on
                    Steven M. Goodreau, University
of Washington
                    Jeffrey Horner
                    Martina Morris, University of W
ashington
Based on "statnet" project software (statnet.org).
For license and citation information see statnet.or
g/attribution
or type citation("networkDynamic").

Loading required package: animation

ndtv: version 0.12.2, created on 2018-5-9
Copyright (c) 2018, Skye Bender-deMoll, University
of Washington
                    with contributions from
                    Martina Morris, University of W
ashington
Based on "statnet" project software (statnet.org).
For license and citation information see statnet.or
g/attribution
or type citation("ndtv").
```

DATASET 1: edgelist

The first data set we are going to work with consists of two files, “Media-Example-NODES.csv” and “Media-Example-EDGES.csv”

In [5]:

```
nodes <- read.csv("dataJune14th/Dataset1-Media-Example-NODES.csv", header=T, as.is=T)
links <- read.csv("dataJune14th/Dataset1-Media-Example-EDGES.csv", header=T, as.is=T)
```

In [6]:

```
#Let's look at the data
head(nodes)
head(links)
nrow(nodes); length(unique(nodes$id))
nrow(links); nrow(unique(links[,c("from", "to")]))
```

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

from	to	weight	type
s01	s02	10	hyperlink
s01	s02	12	hyperlink
s01	s03	22	hyperlink
s01	s04	21	hyperlink
s04	s11	22	mention
s05	s15	21	mention

17

17

52

49

In [7]:

```
links <- aggregate(links[,3], links[,-3], sum)
links <- links[order(links$from, links$to),]
colnames(links)[4] <- "weight"
rownames(links) <- NULL
```

In [8]:

```
# Dataset 2
nodes2 <- read.csv("dataJune14th/Dataset2-Media-User-Example-N
ODES.csv", header=T, as.is=T)
links2 <- read.csv("dataJune14th/Dataset2-Media-User-Example-E
DGES.csv", header=T, row.names=1)
```

In [9]:

```
#Examine
head(nodes2)
head(links2)
```

id	media	media.type	media.name	audience.size
s01	NYT	1	Newspaper	20
s02	WaPo	1	Newspaper	25
s03	WSJ	1	Newspaper	30
s04	USAT	1	Newspaper	32
s05	LATimes	1	Newspaper	20
s06	CNN	2	TV	56

	U01	U02	U03	U04	U05	U06	U07	U08	U09	U10	U11	U
s01	1	1	1	0	0	0	0	0	0	0	0	0
s02	0	0	0	1	1	0	0	0	0	0	0	0
s03	0	0	0	0	0	1	1	1	1	0	0	0
s04	0	0	0	0	0	0	0	0	1	1	1	0
s05	0	0	0	0	0	0	0	0	0	0	1	1
s06	0	0	0	0	0	0	0	0	0	0	0	0

In [10]:

```
# We can see that links2 is an adjacency matrix for a two-mode
 network:


links2 <- as.matrix(links2)
dim(links2)
dim(nodes2)
```

10 20

30 5

Network visualization: first steps with igraph

We start by converting the raw data to an igraph network object. Here we use igraph's graph.data.frame function, which takes two data frames: d and vertices.

- d describes the edges of the network. Its first two columns are the IDs of the source and the target node for each edge. The following columns are edge attributes (weight, type, label, or anything else).
- vertices starts with a column of node IDs. Any following columns are interpreted as node attributes.

In [11]:

```
net <- graph.data.frame(links, nodes, directed=T)
net
```

```
IGRAPH 3532d6d 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 3532d6d (vertex names):
 [1] s01->s02 s01->s03 s01->s04 s01->s15 s02->s01 s
02->s03 s02->s09 s02->s10
 [9] s03->s01 s03->s04 s03->s05 s03->s08 s03->s10 s
03->s11 s03->s12 s04->s03
[17] s04->s06 s04->s11 s04->s12 s04->s17 s05->s01 s
05->s02 s05->s09 s05->s15
[25] s06->s06 s06->s16 s06->s17 s07->s03 s07->s08 s
07->s10 s07->s14 s08->s03
[33] s08->s07 s08->s09 s09->s10 s10->s03 s12->s06 s
12->s13 s12->s14 s13->s12
[41] s13->s17 s14->s11 s14->s13 s15->s01 s15->s04 s
15->s06 s16->s06 s16->s17
[49] s17->s04
```

The description of an igraph object starts with four letters:

1. D or U, for a directed or undirected graph
2. N for a named graph (where nodes have a name attribute)
3. W for a weighted graph (where edges have a weight attribute)
4. 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:

In [12]:

```
E(net)       # The edges of the "net" object
V(net)       # The vertices of the "net" object
E(net)$type  # Edge attribute "type"
V(net)$media # Vertex attribute "media"

# You can also manipulate the network matrix directly:
net[1,]
net[5,7]
```

```
+ 49/49 edges from 3532d6d (vertex names):
 [1] s01->s02 s01->s03 s01->s04 s01->s15 s02->s01 s
02->s03 s02->s09 s02->s10
 [9] s03->s01 s03->s04 s03->s05 s03->s08 s03->s10 s
03->s11 s03->s12 s04->s03
[17] s04->s06 s04->s11 s04->s12 s04->s17 s05->s01 s
05->s02 s05->s09 s05->s15
[25] s06->s06 s06->s16 s06->s17 s07->s03 s07->s08 s
07->s10 s07->s14 s08->s03
[33] s08->s07 s08->s09 s09->s10 s10->s03 s12->s06 s
12->s13 s12->s14 s13->s12
[41] s13->s17 s14->s11 s14->s13 s15->s01 s15->s04 s
15->s06 s16->s06 s16->s17
[49] s17->s04

+ 17/17 vertices, named, from 3532d6d:
 [1] s01 s02 s03 s04 s05 s06 s07 s08 s09 s10 s11 s1
2 s13 s14 s15 s16 s17
```

'hyperlink' 'hyperlink' 'hyperlink' 'mention'
'hyperlink' 'hyperlink' 'hyperlink' 'hyperlink'
'hyperlink' 'hyperlink' 'hyperlink' 'hyperlink'
'mention' 'hyperlink' 'hyperlink' 'hyperlink'
'mention' 'mention' 'hyperlink' 'mention' 'mention'
'hyperlink' 'hyperlink' 'mention' 'hyperlink'
'hyperlink' 'mention' 'mention' 'mention' 'hyperlink'
'mention' 'hyperlink' 'mention' 'mention' 'mention'
'hyperlink' 'mention' 'hyperlink' 'mention'
'hyperlink' 'mention' 'mention' 'mention' 'hyperlink'
'hyperlink' 'hyperlink' 'hyperlink' 'mention'
'hyperlink'

'NY Times' 'Washington Post' 'Wall Street Journal'
'USA Today' 'LA Times' 'New York Post' 'CNN'
'MSNBC' 'FOX News' 'ABC' 'BBC' 'Yahoo News'
'Google News' 'Reuters.com' 'NYTimes.com'
'WashingtonPost.com' 'AOL.com'

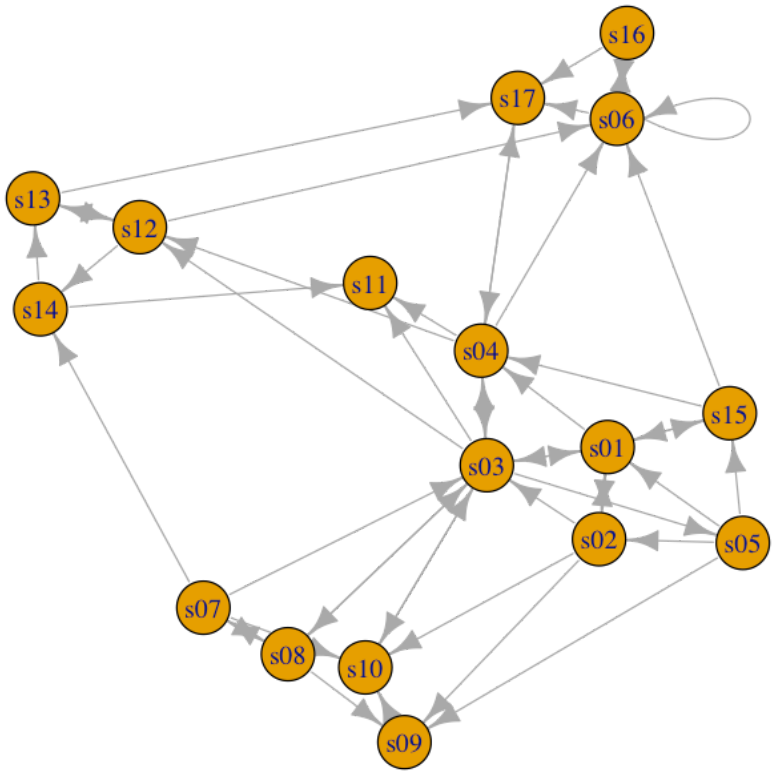
s01
0
s02
22
s03
22
s04
21
s05
0
s06
0
s07
0
s08
0
s09
0
s10
0
s11
0
s12
0
s13
0
s14
0
s15
20
s16
0
s17
0

0

First plot ...

In [13]:

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



In [14]:

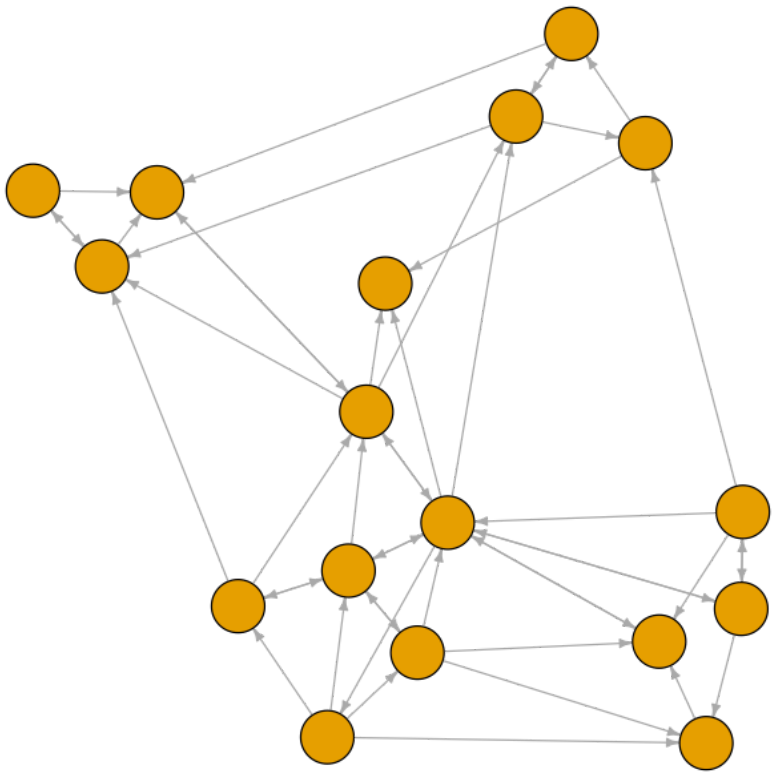
```
#That doesn't look very good. Let's start fixing things by rem
oving the loops in the graph.
net <- simplify(net, remove.multiple = F, remove.loops = T)
```

You might notice that could have used simplify to combine multiple edges by summing their weights with a command like simplify(net, edge.attr.comb=list(Weight="sum","ignore")). The problem is that this would also combine multiple edge types (in our data: “hyperlinks” and “mentions”).

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

In [15]:

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

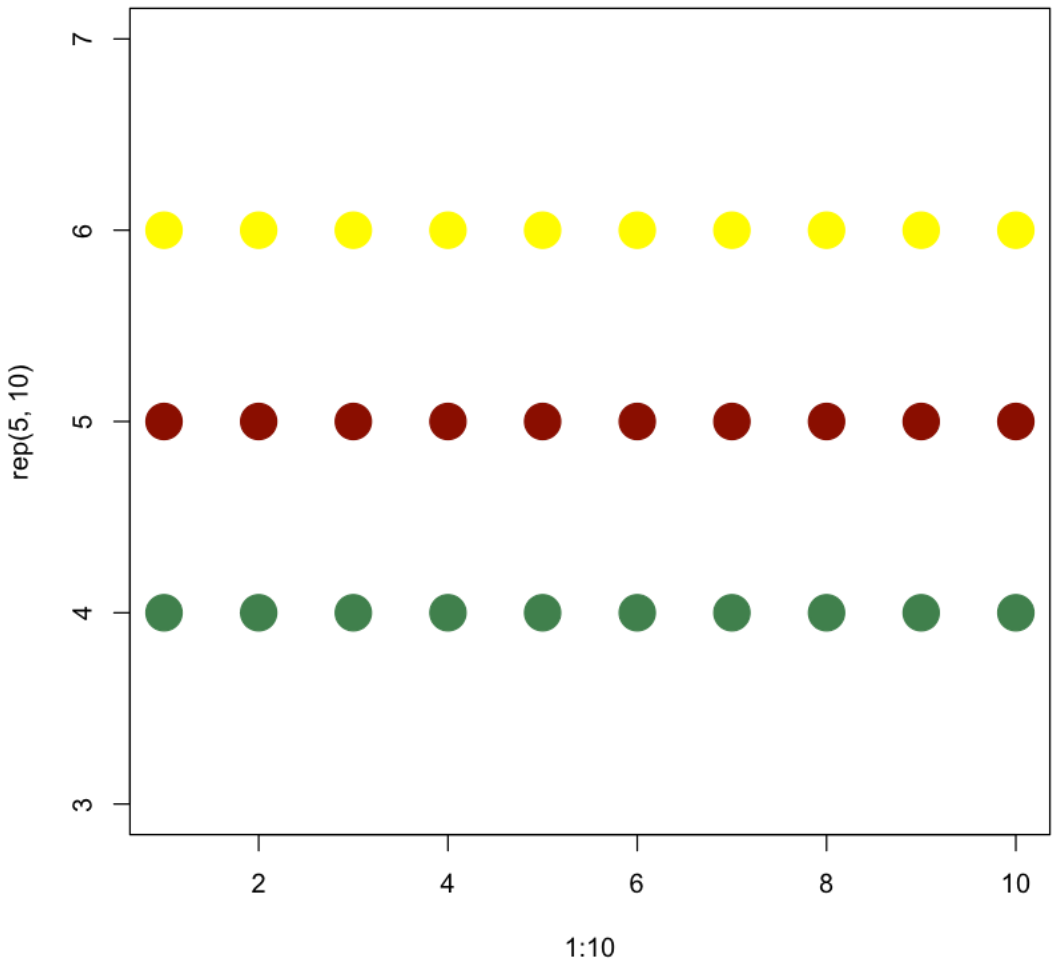


A brief detour I: Colors in R plots

Colors are pretty, but more importantly they help people differentiate between types of objects, or levels of an attribute. In most R functions, you can use named colors, hex, or RGB values. In the simple base R plot chart below, x and y are the point coordinates, pch is the point symbol shape, cex is the point size, and col is the color. To see the parameters for plotting in base R, check out ?par

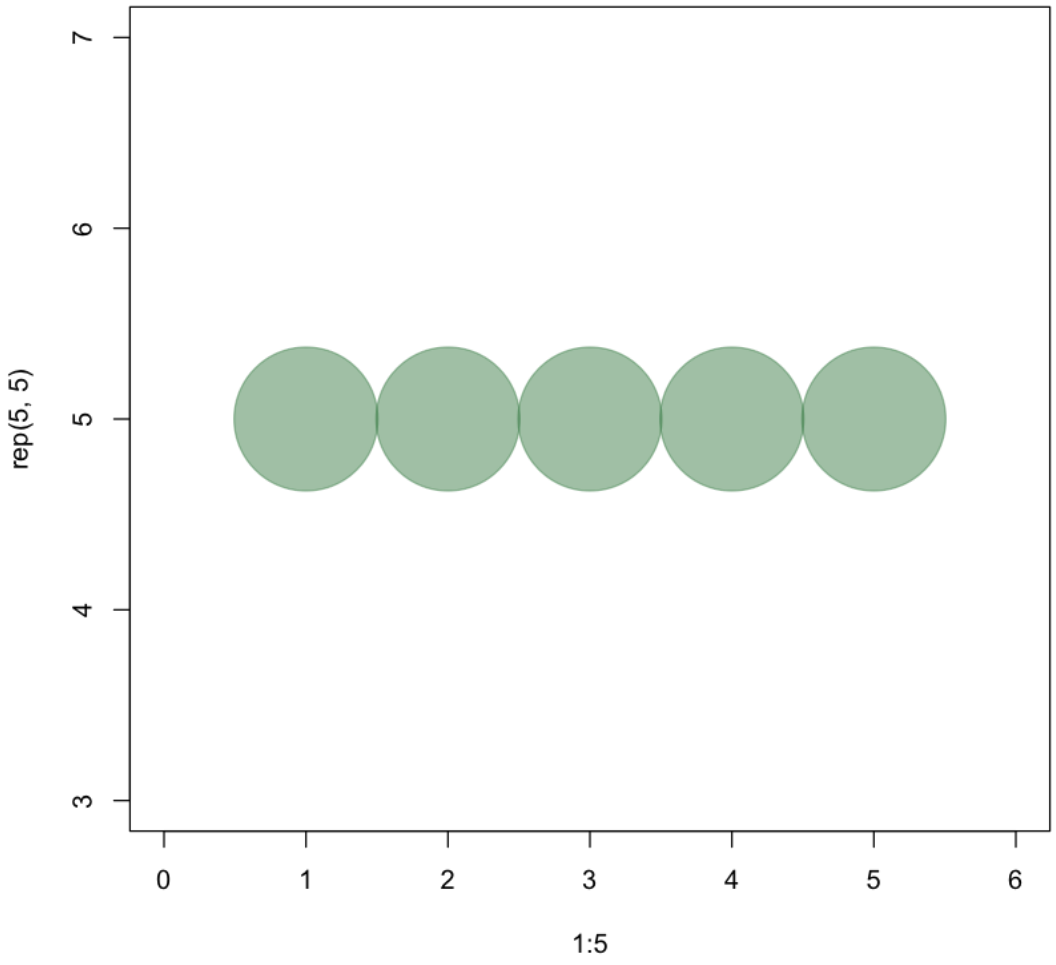
In [16]:

```
plot(x=1:10, y=rep(5,10), pch=19, cex=3, col="dark red")
points(x=1:10, y=rep(6, 10), pch=19, cex=3, col="557799")
points(x=1:10, y=rep(4, 10), pch=19, cex=3, col=rgb(.25, .5, .
3))
```



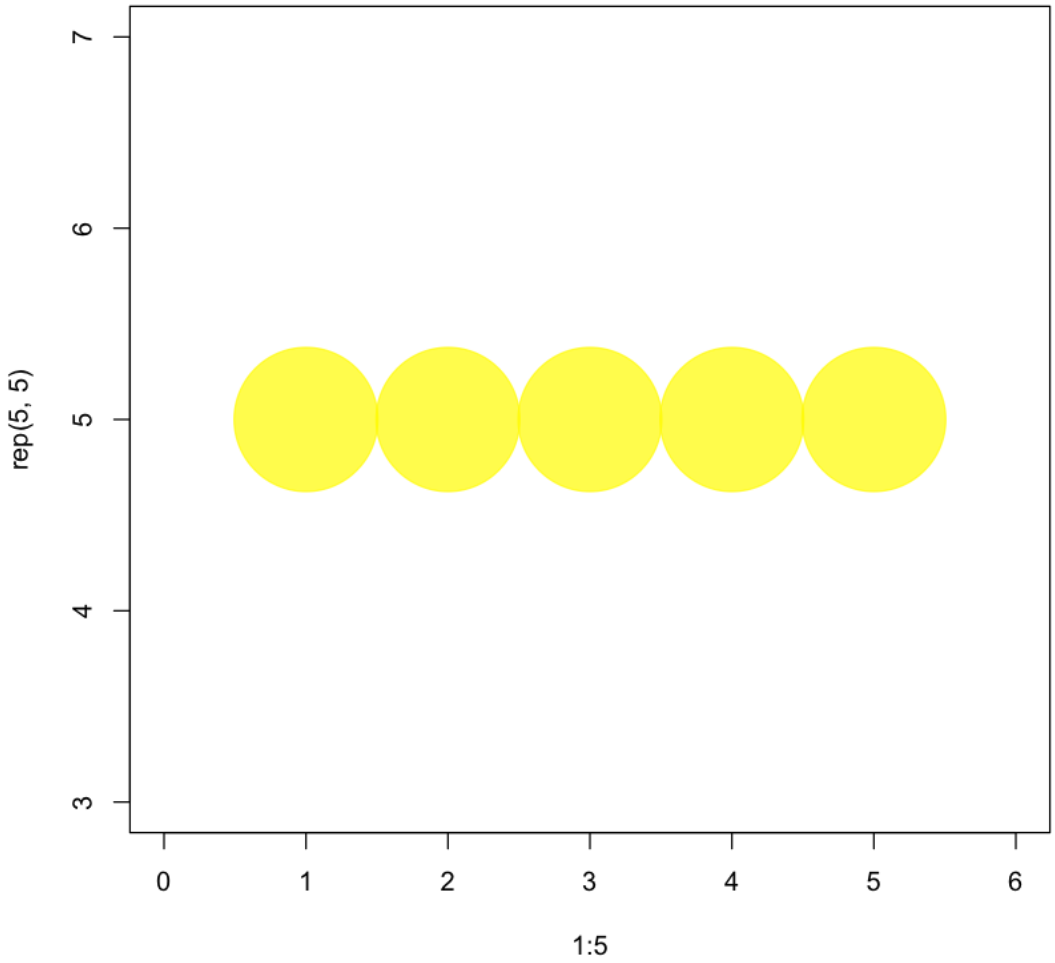
In [17]:

```
#You may notice that RGB here ranges from 0 to 1. While this i
s the R default,
#you can also set it for to the 0-255 range using something
#like rgb(10, 100, 100, maxColorValue=255).

# We can set the opacity/transparency of an element using the
 parameter alpha (range 0-1):
plot(x=1:5, y=rep(5,5), pch=19, cex=12, col=rgb(.25, .5, .3, a
lpha=.5), xlim=c(0,6))
```



In [18]:

```
#If we have a hex color representation, we can set the transpa
rency alpha
#using adjustcolor from package grDevices.
#For fun, let's also set the plot background to gray using
#the par() function for graphical parameters.

col.tr <- grDevices::adjustcolor("557799", alpha=0.7)
plot(x=1:5, y=rep(5,5), pch=19, cex=12, col=col.tr, xlim=c(0,6
))
```



In [19]:

```
colors()                              # List all named colors
grep("blue", colors(), value=T)   # Colors that have "blue" in
 the name
```

'white' 'aliceblue' 'antiquewhite' 'antiquewhite1'
'antiquewhite2' 'antiquewhite3' 'antiquewhite4'
'aquamarine' 'aquamarine1' 'aquamarine2'
'aquamarine3' 'aquamarine4' 'azure' 'azure1'
'azure2' 'azure3' 'azure4' 'beige' 'bisque'
'bisque1' 'bisque2' 'bisque3' 'bisque4' 'black'
'blanchedalmond' 'blue' 'blue1' 'blue2' 'blue3'
'blue4' 'blueviolet' 'brown' 'brown1' 'brown2'
'brown3' 'brown4' 'burlywood' 'burlywood1'
'burlywood2' 'burlywood3' 'burlywood4' 'cadetblue'
'cadetblue1' 'cadetblue2' 'cadetblue3' 'cadetblue4'
'chartreuse' 'chartreuse1' 'chartreuse2' 'chartreuse3'
'chartreuse4' 'chocolate' 'chocolate1' 'chocolate2'
'chocolate3' 'chocolate4' 'coral' 'coral1' 'coral2'
'coral3' 'coral4' 'cornflowerblue' 'cornsilk'
'cornsilk1' 'cornsilk2' 'cornsilk3' 'cornsilk4' 'cyan'
'cyan1' 'cyan2' 'cyan3' 'cyan4' 'darkblue'
'darkcyan' 'darkgoldenrod' 'darkgoldenrod1'
'darkgoldenrod2' 'darkgoldenrod3' 'darkgoldenrod4'
'darkgray' 'darkgreen' 'darkgrey' 'darkkhaki'
'darkmagenta' 'darkolivegreen' 'darkolivegreen1'
'darkolivegreen2' 'darkolivegreen3' 'darkolivegreen4'
'darkorange' 'darkorange1' 'darkorange2'
'darkorange3' 'darkorange4' 'darkorchid'
'darkorchid1' 'darkorchid2' 'darkorchid3'
'darkorchid4' 'darkred' 'darksalmon' 'darkseagreen'
'darkseagreen1' 'darkseagreen2' 'darkseagreen3'
'darkseagreen4' 'darkslateblue' 'darkslategray'
'darkslategray1' 'darkslategray2' 'darkslategray3'
'darkslategray4' 'darkslategrey' 'darkturquoise'
'darkviolet' 'deeppink' 'deeppink1' 'deeppink2'
'deeppink3' 'deeppink4' 'deepskyblue'
'deepskyblue1' 'deepskyblue2' 'deepskyblue3'
'deepskyblue4' 'dimgray' 'dimgrey' 'dodgerblue'
'dodgerblue1' 'dodgerblue2' 'dodgerblue3'
'dodgerblue4' 'firebrick' 'firebrick1' 'firebrick2'
'firebrick3' 'firebrick4' 'floralwhite' 'forestgreen'

'gainsboro' 'ghostwhite' 'gold' 'gold1' 'gold2'
'gold3' 'gold4' 'goldenrod' 'goldenrod1'
'goldenrod2' 'goldenrod3' 'goldenrod4' 'gray'
'gray0' 'gray1' 'gray2' 'gray3' 'gray4' 'gray5'
'gray6' 'gray7' 'gray8' 'gray9' 'gray10' 'gray11'
'gray12' 'gray13' 'gray14' 'gray15' 'gray16'
'gray17' 'gray18' 'gray19' 'gray20' 'gray21'
'gray22' 'gray23' 'gray24' 'gray25' 'gray26'
'gray27' 'gray28' 'gray29' 'gray30' 'gray31'
'gray32' 'gray33' 'gray34' 'gray35' 'gray36'
'gray37' 'gray38' 'gray39' 'gray40' 'gray41'
'gray42' 'gray43' 'gray44' 'gray45' 'gray46'
'gray47' 'gray48' 'gray49' 'gray50' 'gray51'
'gray52' 'gray53' 'gray54' 'gray55' 'gray56'
'gray57' 'gray58' 'gray59' 'gray60' 'gray61'
'gray62' 'gray63' 'gray64' 'gray65' 'gray66'
'gray67' 'gray68' 'gray69' 'gray70' 'gray71'
'gray72' 'gray73' 'gray74' 'gray75' 'gray76'
'gray77' 'gray78' 'gray79' 'gray80' 'gray81'
'gray82' 'gray83' 'gray84' 'gray85' 'gray86'
'gray87' 'gray88' 'gray89' 'gray90' 'gray91'
'gray92' 'gray93' 'gray94' 'gray95' 'gray96'
'gray97' 'gray98' 'gray99' 'gray100' 'green'
'green1' 'green2' 'green3' 'green4' 'greenyellow'
'grey' 'grey0' 'grey1' 'grey2' 'grey3' 'grey4'
'grey5' 'grey6' 'grey7' 'grey8' 'grey9' 'grey10'
'grey11' 'grey12' 'grey13' 'grey14' 'grey15'
'grey16' 'grey17' 'grey18' 'grey19' 'grey20'
'grey21' 'grey22' 'grey23' 'grey24' 'grey25'
'grey26' 'grey27' 'grey28' 'grey29' 'grey30'
'grey31' 'grey32' 'grey33' 'grey34' 'grey35'
'grey36' 'grey37' 'grey38' 'grey39' 'grey40'
'grey41' 'grey42' 'grey43' 'grey44' 'grey45'
'grey46' 'grey47' 'grey48' 'grey49' 'grey50'
'grey51' 'grey52' 'grey53' 'grey54' 'grey55'
'grey56' 'grey57' 'grey58' 'grey59' 'grey60'
'grey61' 'grey62' 'grey63' 'grey64' 'grey65'

'grey66' 'grey67' 'grey68' 'grey69' 'grey70'
'grey71' 'grey72' 'grey73' 'grey74' 'grey75'
'grey76' 'grey77' 'grey78' 'grey79' 'grey80'
'grey81' 'grey82' 'grey83' 'grey84' 'grey85'
'grey86' 'grey87' 'grey88' 'grey89' 'grey90'
'grey91' 'grey92' 'grey93' 'grey94' 'grey95'
'grey96' 'grey97' 'grey98' 'grey99' 'grey100'
'honeydew' 'honeydew1' 'honeydew2' 'honeydew3'
'honeydew4' 'hotpink' 'hotpink1' 'hotpink2'
'hotpink3' 'hotpink4' 'indianred' 'indianred1'
'indianred2' 'indianred3' 'indianred4' 'ivory' 'ivory1'
'ivory2' 'ivory3' 'ivory4' 'khaki' 'khaki1' 'khaki2'
'khaki3' 'khaki4' 'lavender' 'lavenderblush'
'lavenderblush1' 'lavenderblush2' 'lavenderblush3'
'lavenderblush4' 'lawngreen' 'lemonchiffon'
'lemonchiffon1' 'lemonchiffon2' 'lemonchiffon3'
'lemonchiffon4' 'lightblue' 'lightblue1' 'lightblue2'
'lightblue3' 'lightblue4' 'lightcoral' 'lightcyan'
'lightcyan1' 'lightcyan2' 'lightcyan3' 'lightcyan4'
'lightgoldenrod' 'lightgoldenrod1' 'lightgoldenrod2'
'lightgoldenrod3' 'lightgoldenrod4'
'lightgoldenrodyellow' 'lightgray' 'lightgreen'
'lightgrey' 'lightpink' 'lightpink1' 'lightpink2'
'lightpink3' 'lightpink4' 'lightsalmon' 'lightsalmon1'
'lightsalmon2' 'lightsalmon3' 'lightsalmon4'
'lightseagreen' 'lightskyblue' 'lightskyblue1'
'lightskyblue2' 'lightskyblue3' 'lightskyblue4'
'lightslateblue' 'lightslategray' 'lightslategrey'
'lightsteelblue' 'lightsteelblue1' 'lightsteelblue2'
'lightsteelblue3' 'lightsteelblue4' 'lightyellow'
'lightyellow1' 'lightyellow2' 'lightyellow3'
'lightyellow4' 'limegreen' 'linen' 'magenta'
'magenta1' 'magenta2' 'magenta3' 'magenta4'
'maroon' 'maroon1' 'maroon2' 'maroon3'
'maroon4' 'mediumaquamarine' 'mediumblue'
'mediumorchid' 'mediumorchid1' 'mediumorchid2'
'mediumorchid3' 'mediumorchid4' 'mediumpurple'

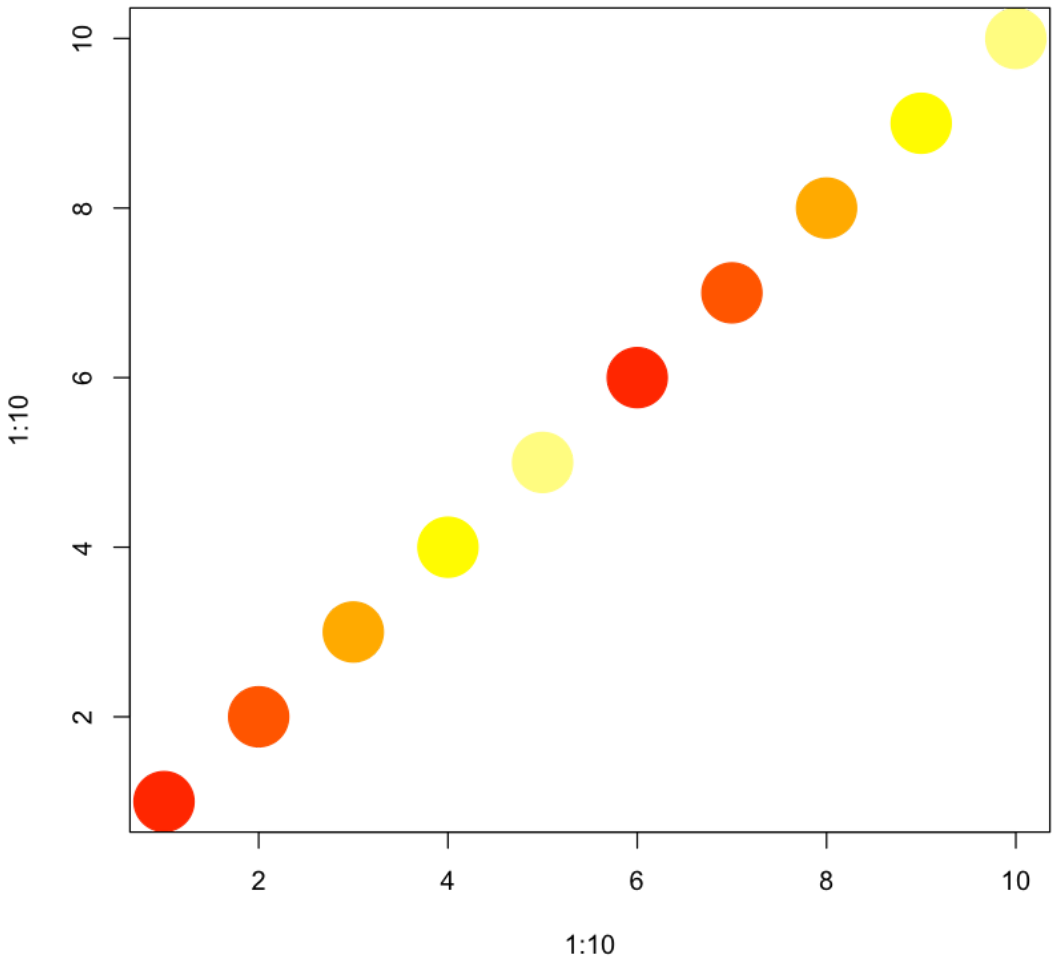
'mediumpurple1' 'mediumpurple2' 'mediumpurple3'
'mediumpurple4' 'mediumseagreen'
'mediumslateblue' 'mediumspringgreen'
'mediumturquoise' 'mediumvioletred' 'midnightblue'
'mintcream' 'mistyrose' 'mistyrose1' 'mistyrose2'
'mistyrose3' 'mistyrose4' 'moccasin' 'navajowhite'
'navajowhite1' 'navajowhite2' 'navajowhite3'
'navajowhite4' 'navy' 'navyblue' 'oldlace'
'olivedrab' 'olivedrab1' 'olivedrab2' 'olivedrab3'
'olivedrab4' 'orange' 'orange1' 'orange2' 'orange3'
'orange4' 'orangered' 'orangered1' 'orangered2'
'orangered3' 'orangered4' 'orchid' 'orchid1'
'orchid2' 'orchid3' 'orchid4' 'palegoldenrod'
'palegreen' 'palegreen1' 'palegreen2' 'palegreen3'
'palegreen4' 'paleturquoise' 'paleturquoise1'
'paleturquoise2' 'paleturquoise3' 'paleturquoise4'
'palevioletred' 'palevioletred1' 'palevioletred2'
'palevioletred3' 'palevioletred4' 'papayawhip'
'peachpuff' 'peachpuff1' 'peachpuff2' 'peachpuff3'
'peachpuff4' 'peru' 'pink' 'pink1' 'pink2' 'pink3'
'pink4' 'plum' 'plum1' 'plum2' 'plum3' 'plum4'
'powderblue' 'purple' 'purple1' 'purple2' 'purple3'
'purple4' 'red' 'red1' 'red2' 'red3' 'red4'
'rosybrown' 'rosybrown1' 'rosybrown2' 'rosybrown3'
'rosybrown4' 'royalblue' 'royalblue1' 'royalblue2'
'royalblue3' 'royalblue4' 'saddlebrown' 'salmon'
'salmon1' 'salmon2' 'salmon3' 'salmon4'
'sandybrown' 'seagreen' 'seagreen1' 'seagreen2'
'seagreen3' 'seagreen4' 'seashell' 'seashell1'
'seashell2' 'seashell3' 'seashell4' 'sienna' 'sienna1'
'sienna2' 'sienna3' 'sienna4' 'skyblue' 'skyblue1'
'skyblue2' 'skyblue3' 'skyblue4' 'slateblue'
'slateblue1' 'slateblue2' 'slateblue3' 'slateblue4'
'slategray' 'slategray1' 'slategray2' 'slategray3'
'slategray4' 'slategrey' 'snow' 'snow1' 'snow2'
'snow3' 'snow4' 'springgreen' 'springgreen1'
'springgreen2' 'springgreen3' 'springgreen4'

'steelblue' 'steelblue1' 'steelblue2' 'steelblue3'
'steelblue4' 'tan' 'tan1' 'tan2' 'tan3' 'tan4'
'thistle' 'thistle1' 'thistle2' 'thistle3' 'thistle4'
'tomato' 'tomato1' 'tomato2' 'tomato3' 'tomato4'
'turquoise' 'turquoise1' 'turquoise2' 'turquoise3'
'turquoise4' 'violet' 'violetred' 'violetred1'
'violetred2' 'violetred3' 'violetred4' 'wheat' 'wheat1'
'wheat2' 'wheat3' 'wheat4' 'whitesmoke' 'yellow'
'yellow1' 'yellow2' 'yellow3' 'yellow4' 'yellowgreen'

'aliceblue' 'blue' 'blue1' 'blue2' 'blue3' 'blue4'
'blueviolet' 'cadetblue' 'cadetblue1' 'cadetblue2'
'cadetblue3' 'cadetblue4' 'cornflowerblue' 'darkblue'
'darkslateblue' 'deepskyblue' 'deepskyblue1'
'deepskyblue2' 'deepskyblue3' 'deepskyblue4'
'dodgerblue' 'dodgerblue1' 'dodgerblue2'
'dodgerblue3' 'dodgerblue4' 'lightblue' 'lightblue1'
'lightblue2' 'lightblue3' 'lightblue4' 'lightskyblue'
'lightskyblue1' 'lightskyblue2' 'lightskyblue3'
'lightskyblue4' 'lightslateblue' 'lightsteelblue'
'lightsteelblue1' 'lightsteelblue2' 'lightsteelblue3'
'lightsteelblue4' 'mediumblue' 'mediumslateblue'
'midnightblue' 'navyblue' 'powderblue' 'royalblue'
'royalblue1' 'royalblue2' 'royalblue3' 'royalblue4'
'skyblue' 'skyblue1' 'skyblue2' 'skyblue3'
'skyblue4' 'slateblue' 'slateblue1' 'slateblue2'
'slateblue3' 'slateblue4' 'steelblue' 'steelblue1'
'steelblue2' 'steelblue3' 'steelblue4'

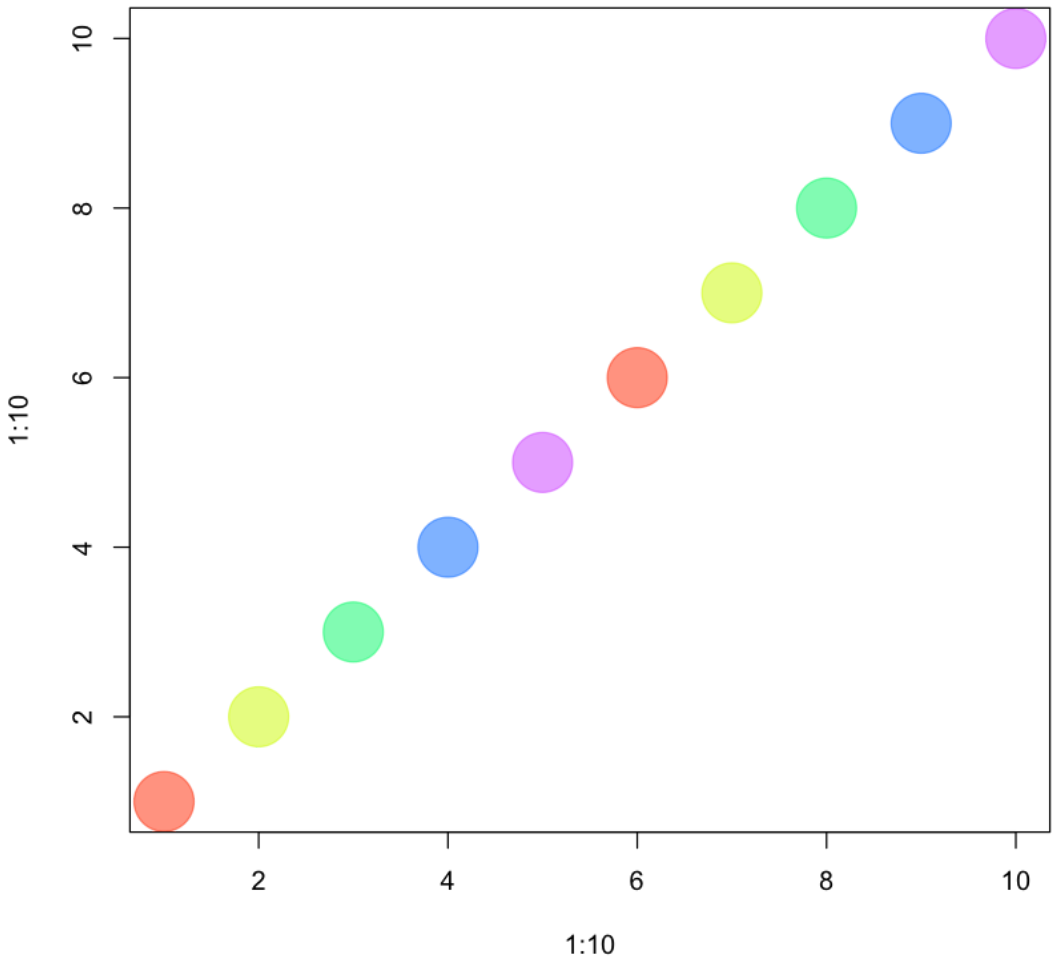
In [20]:

```
pal1 <- heat.colors(5, alpha=1)   #  5 colors from the heat pa
lette, opaque
pal2 <- rainbow(5, alpha=.5)      #  5 colors from the heat pa
lette, transparent
plot(x=1:10, y=1:10, pch=19, cex=5, col=pal1)
```



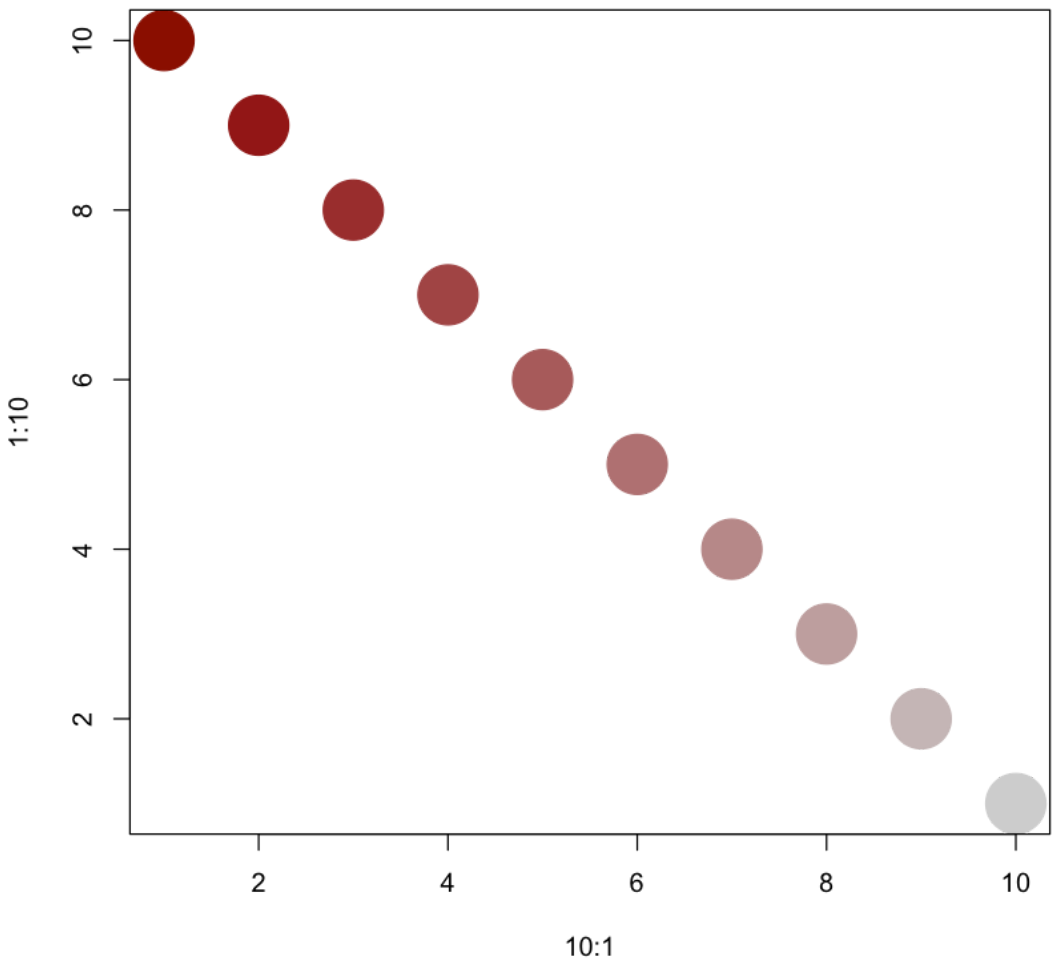
In [21]:

```
plot(x=1:10, y=1:10, pch=19, cex=5, col=pal2)
```



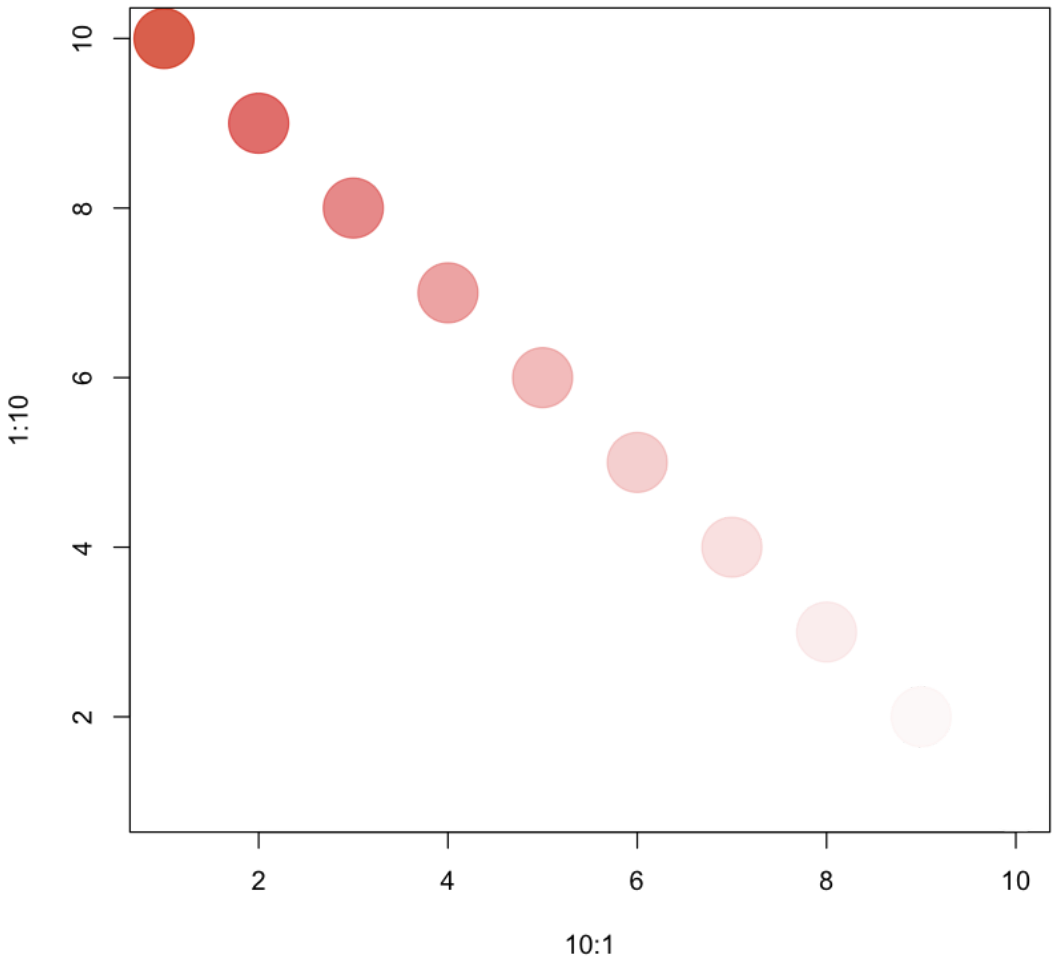
In [22]:

```
palf <- colorRampPalette(c("gray80", "dark red"))
plot(x=10:1, y=1:10, pch=19, cex=5, col=palf(10))
```



In [23]:

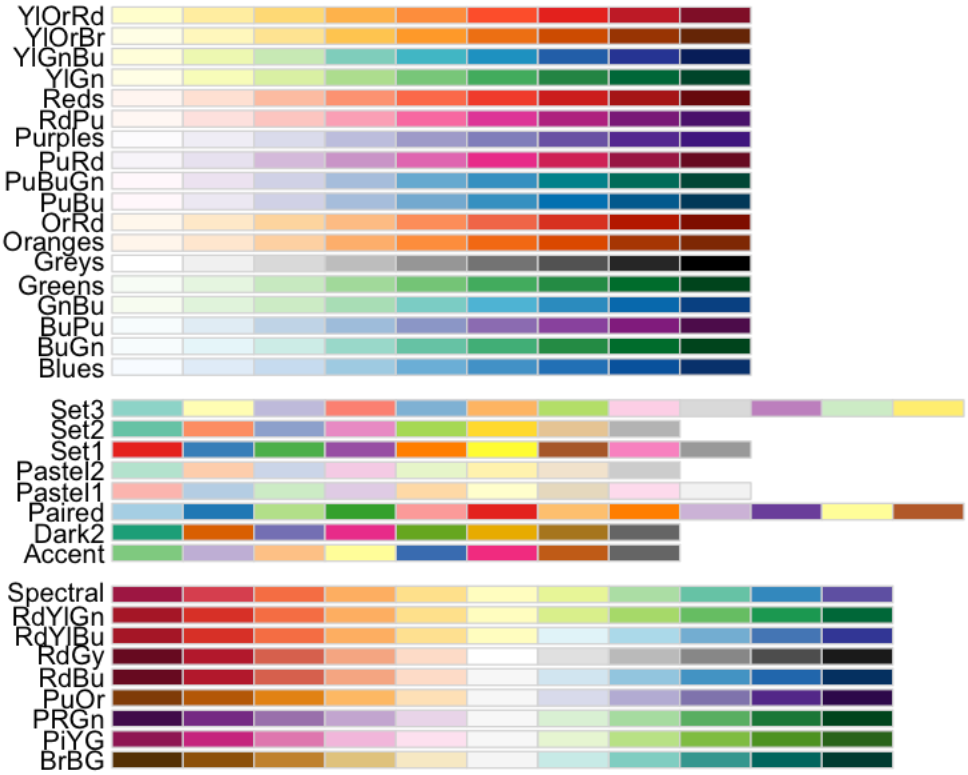
```
palf <- colorRampPalette(c(rgb(1,1,1, .2),rgb(.8,0,0, .7)), al
pha=TRUE)
plot(x=10:1, y=1:10, pch=19, cex=5, col=palf(10))
```



In [24]:

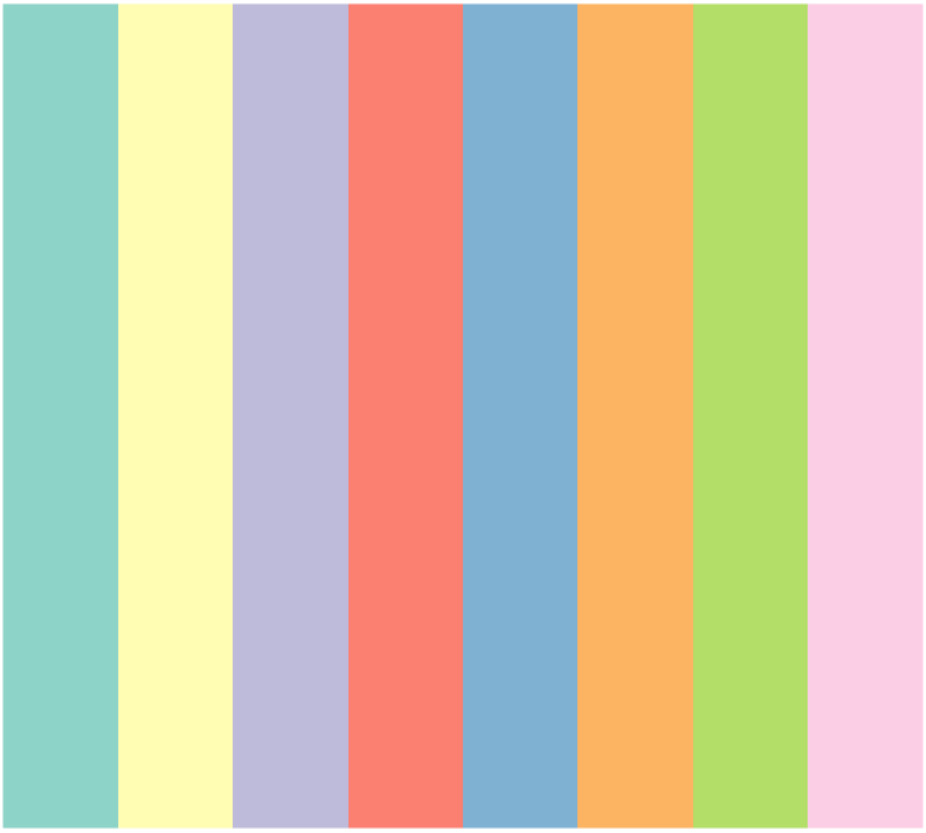
```
# If you don't have R ColorBrewer already, you will need to in
stall it:
install.packages("RColorBrewer")
library(RColorBrewer)
display.brewer.all()
```

```
The downloaded binary packages are in
        /var/folders/jw/knt_b30n31xgtwmrfn00sctm000
0gn/T//RtmpHjraEZ/downloaded_packages
```



In [25]:

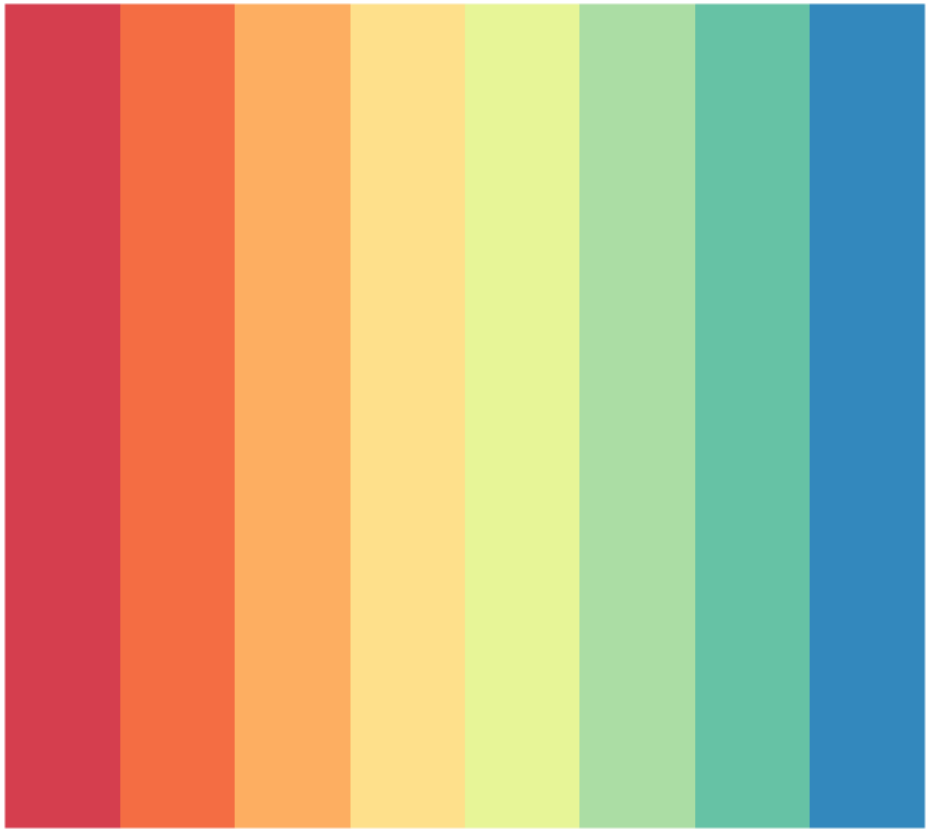
```
display.brewer.pal(8, "Set3")
```



Set3 (qualitative)

In [26]:

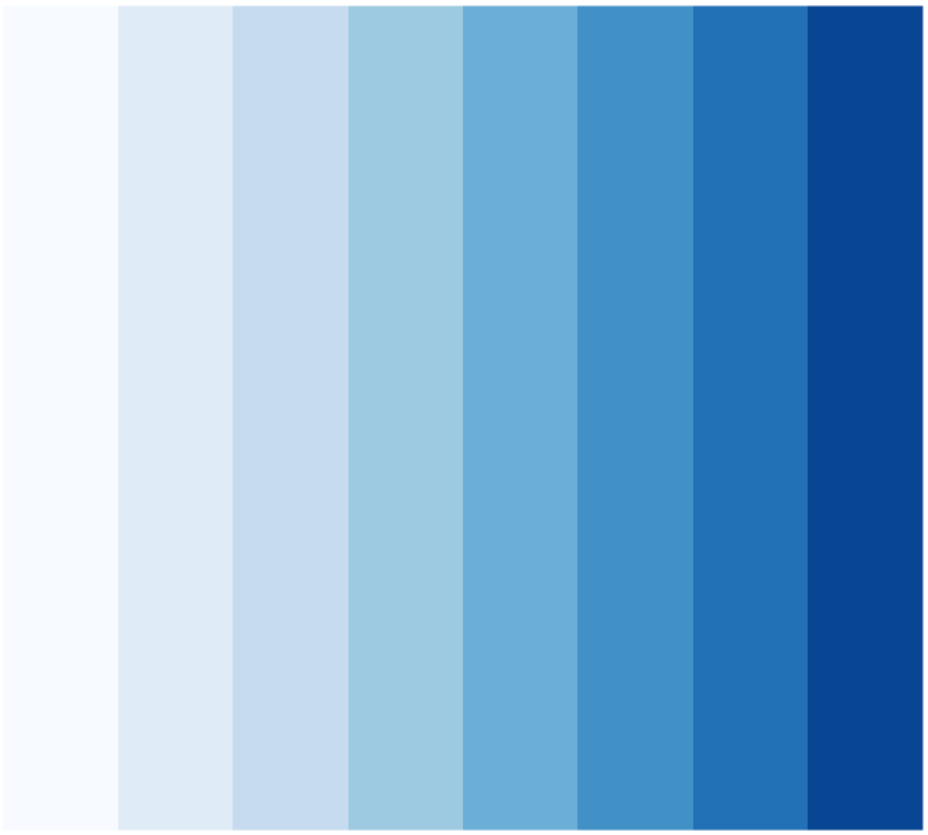
```
display.brewer.pal(8, "Spectral")
```



Spectral (divergent)

In [27]:

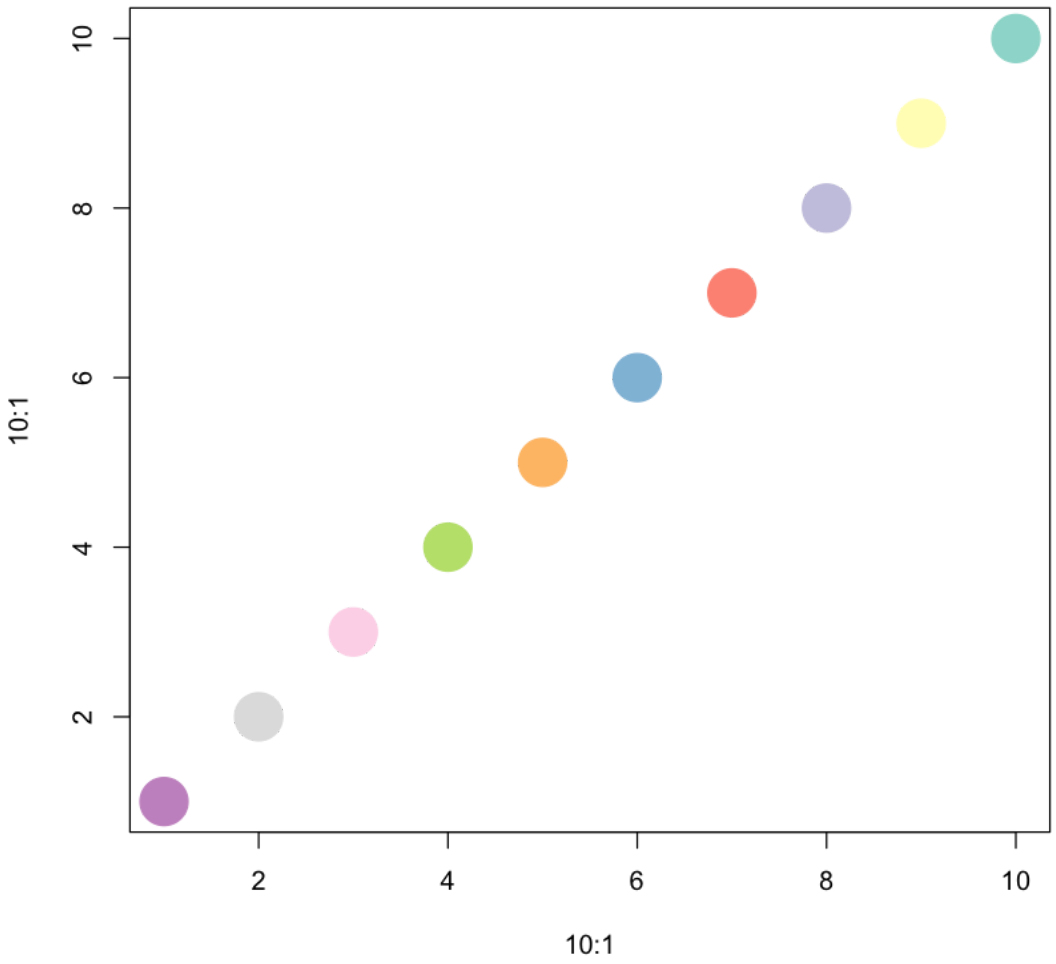
```
display.brewer.pal(8, "Blues")
```



Blues (sequential)

In [28]:

```
pal3 <- brewer.pal(10, "Set3")
plot(x=10:1, y=10:1, pch=19, cex=4, col=pal3)
```



Back to our main plot line: plotting networks

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.

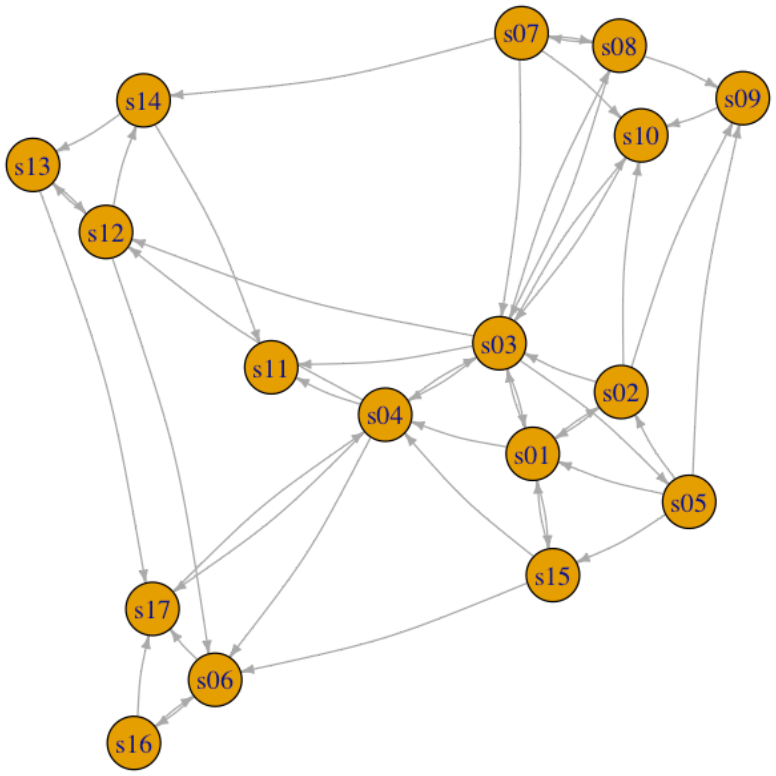
The igraph plotting parameters include (among others):

Plotting parameters

NODES	
vertex.color	Node color
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 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

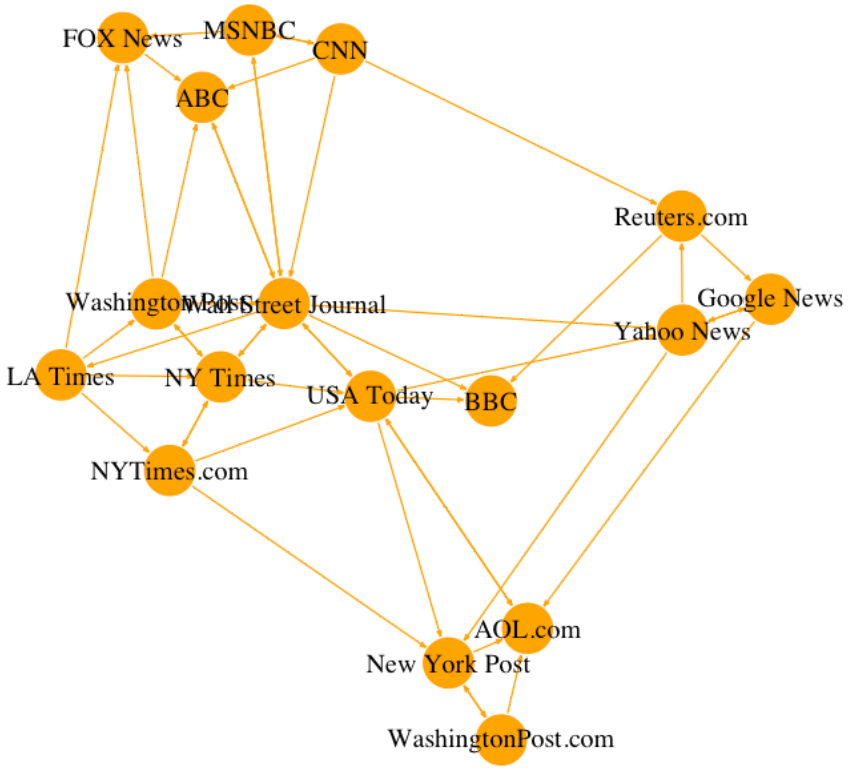
In [29]:

```
# Plot with curved edges (edge.curved=.1) and reduce arrow siz
e:
plot(net, edge.arrow.size=.4, edge.curved=.1)
```



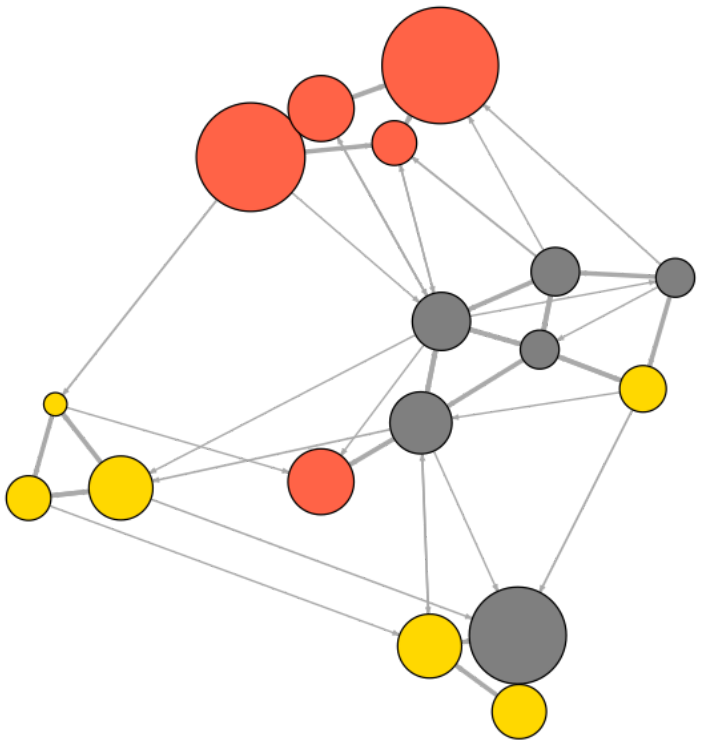
In [30]:

```
# Set edge color to light gray, the node & border color to ora
nge
# Replace the vertex label with the node names stored in "medi
a"
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")
```



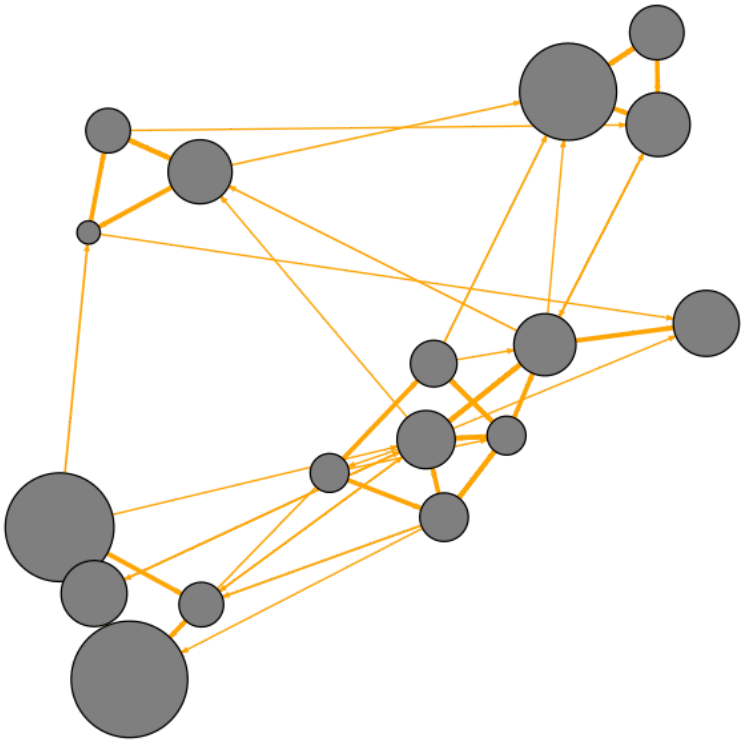
In [31]:

```
# Generate colors base 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 <- igraph::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"
E(net)$width <- 1+E(net)$weight/12
plot(net)
```



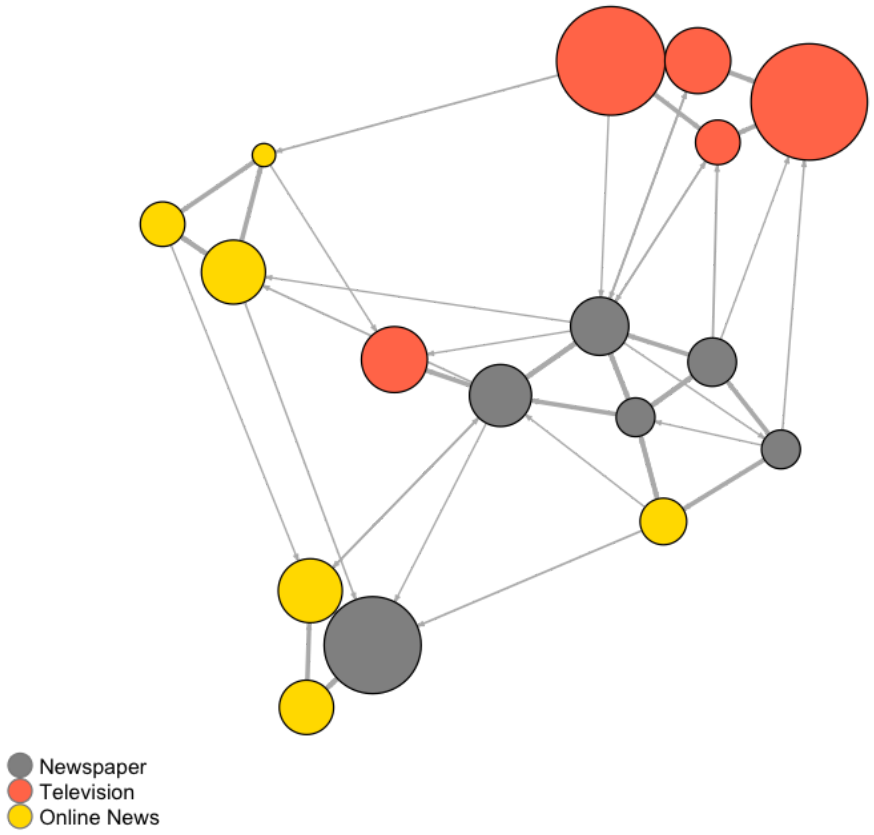
In [32]:

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



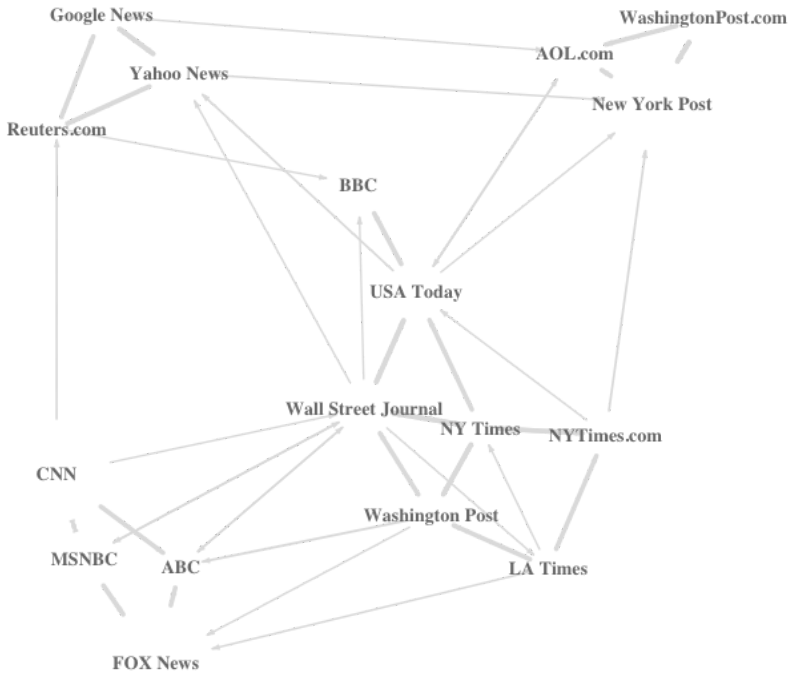
In [33]:

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



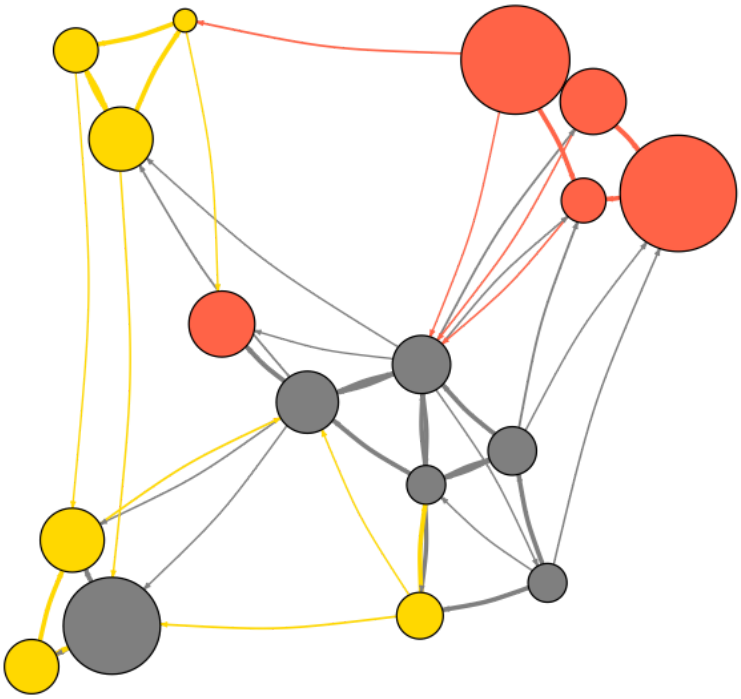
In [34]:

```
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")
```



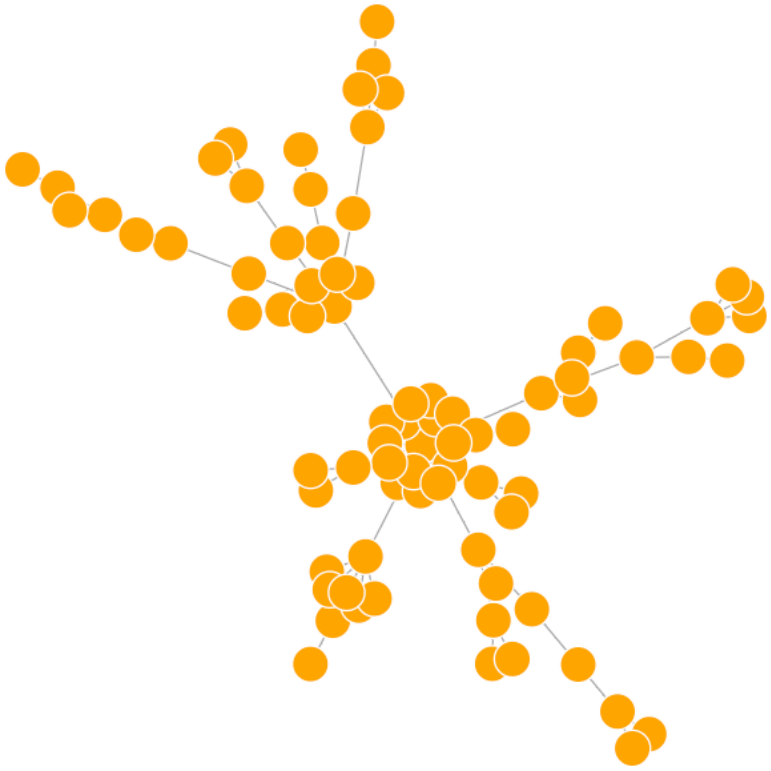
In [35]:

```
edge.start <- igraph::get.edges(net, 1:ecount(net))[,1]
edge.col <- V(net)$color[edge.start]

plot(net, edge.color=edge.col, edge.curved=.1)
```



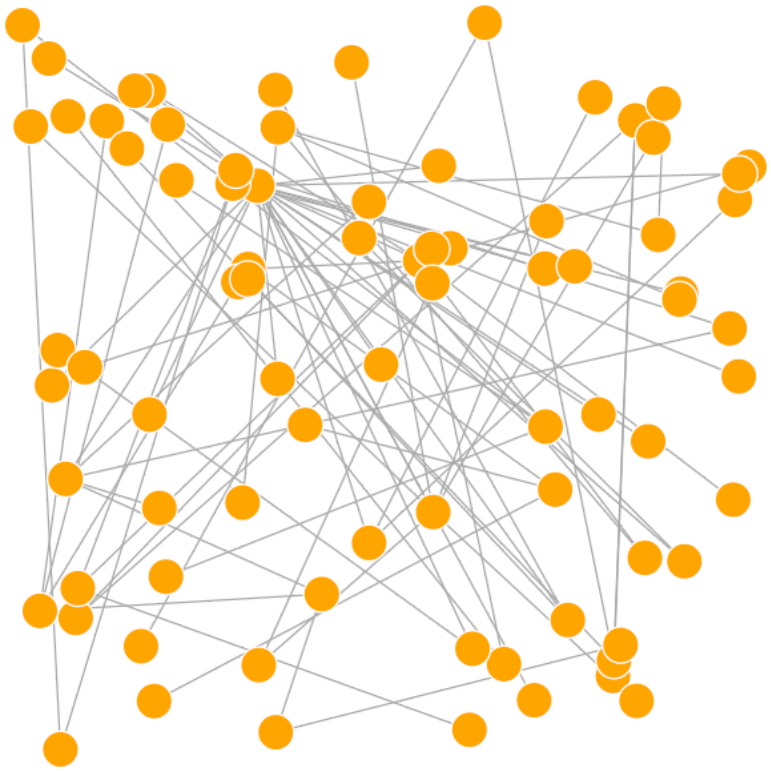
In [36]:

```
net.bg <- barabasi.game(80)
V(net.bg)$frame.color <- "white"
V(net.bg)$color <- "orange"
V(net.bg)$label <- ""
V(net.bg)$size <- 10
E(net.bg)$arrow.mode <- 0
plot(net.bg)
```



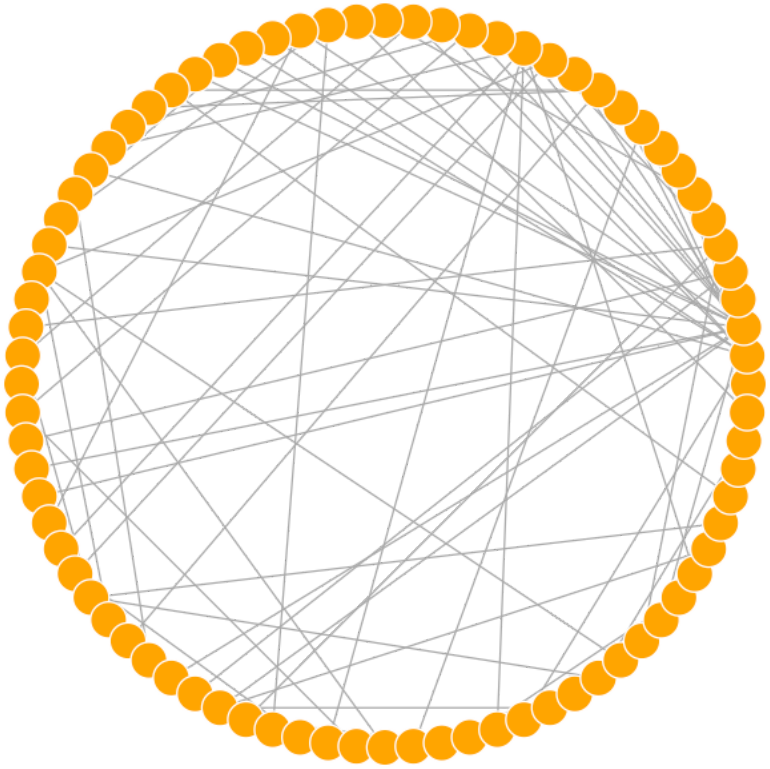
In [37]:

```
plot(net.bg, layout=layout.random)
```



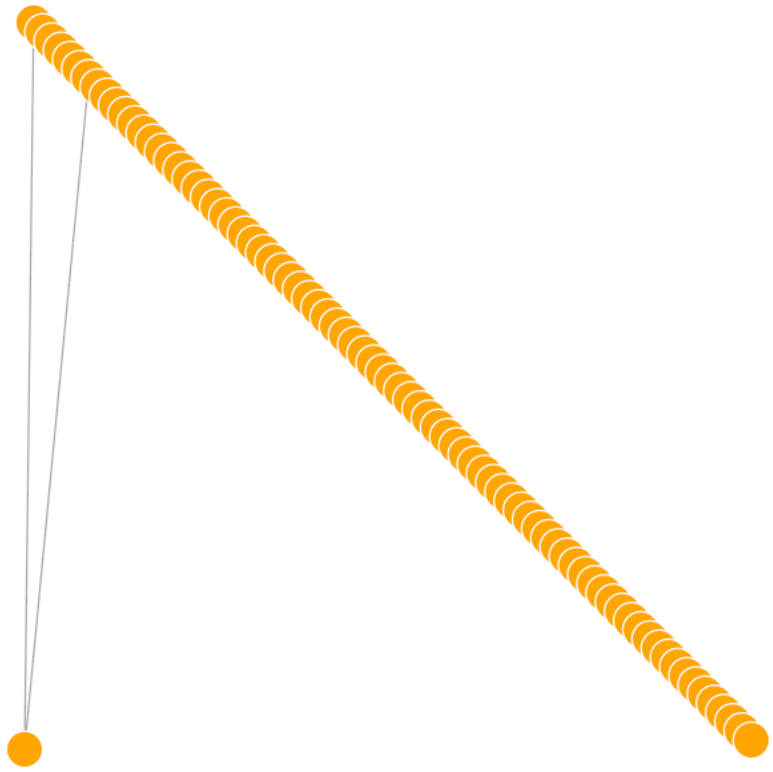
In [38]:

```
l <- layout.circle(net.bg)
plot(net.bg, layout=l)
```



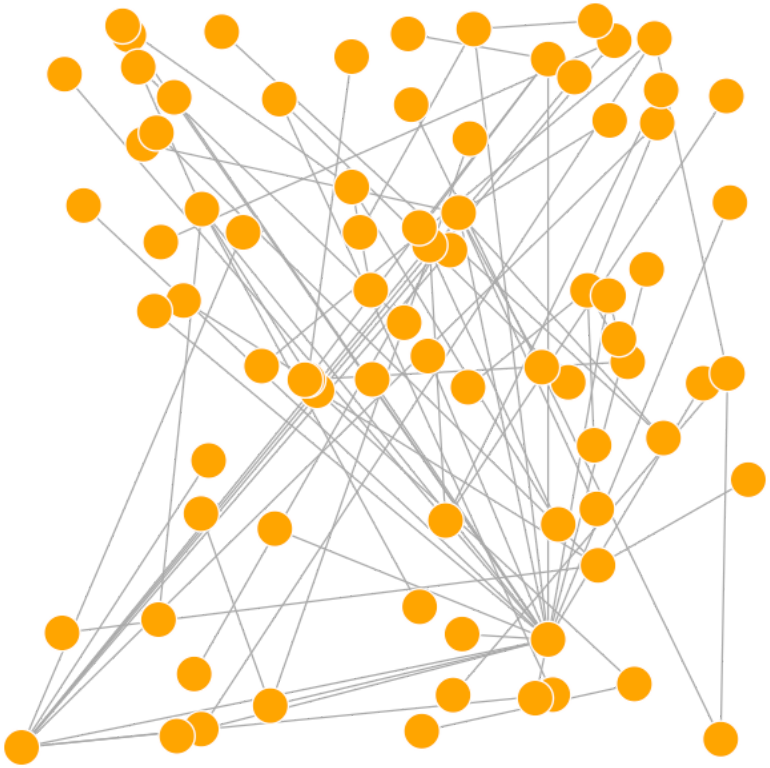
In [39]:

```
l <- matrix(c(1:vcount(net.bg), c(1, vcount(net.bg):2)), vcoun
t(net.bg), 2)
plot(net.bg, layout=l)
```



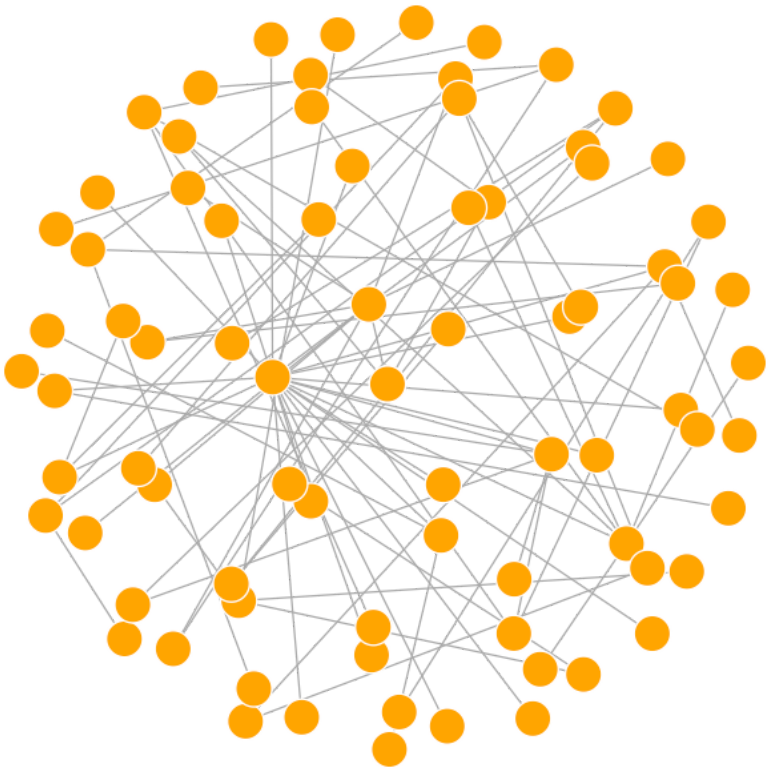
In [40]:

```
l <- layout.random(net.bg)
plot(net.bg, layout=l)
```



In [41]:

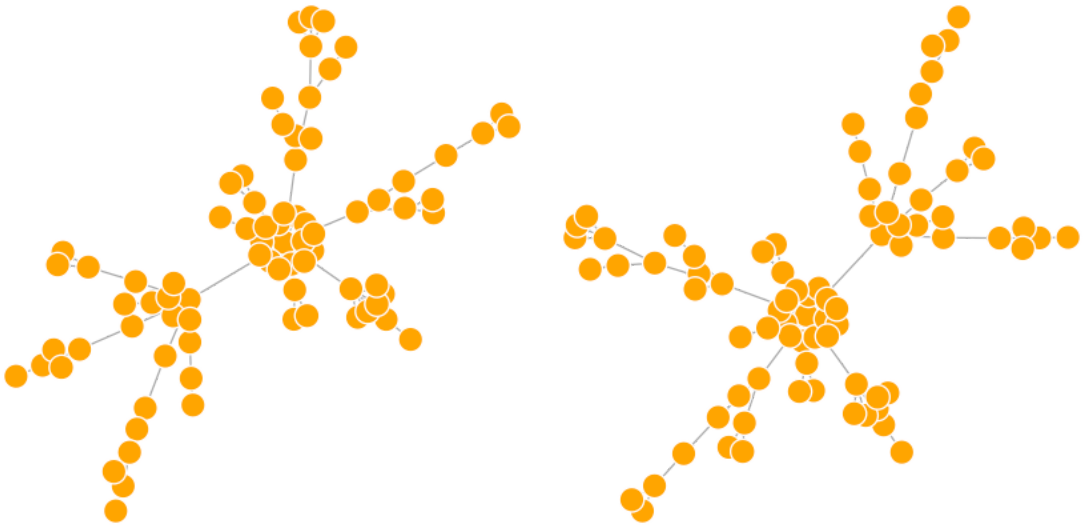
```
# 3D sphere layout
l <- layout.sphere(net.bg)
plot(net.bg, layout=l)
```



In [42]:

```
l <- layout.fruchterman.reingold(net.bg, repulserad=vcount(ne
t.bg)^3,
                                 area=vcount(net.bg)^2.4)
par(mfrow=c(1,2),  mar=c(0,0,0,0)) # plot two figures - 1 row,
 2 columns
plot(net.bg, layout=layout.fruchterman.reingold)
plot(net.bg, layout=l)
```

```
Warning message in layout_with_fr(structure(list(8
0, TRUE, c(1, 2, 3, 4, 5, 6, 7, :
“Argument `area' is deprecated and has no effect”Wa
rning message in layout_with_fr(structure(list(80,
TRUE, c(1, 2, 3, 4, 5, 6, 7, :
“Argument `repulserad' is deprecated and has no eff
ect”
```



In [43]:

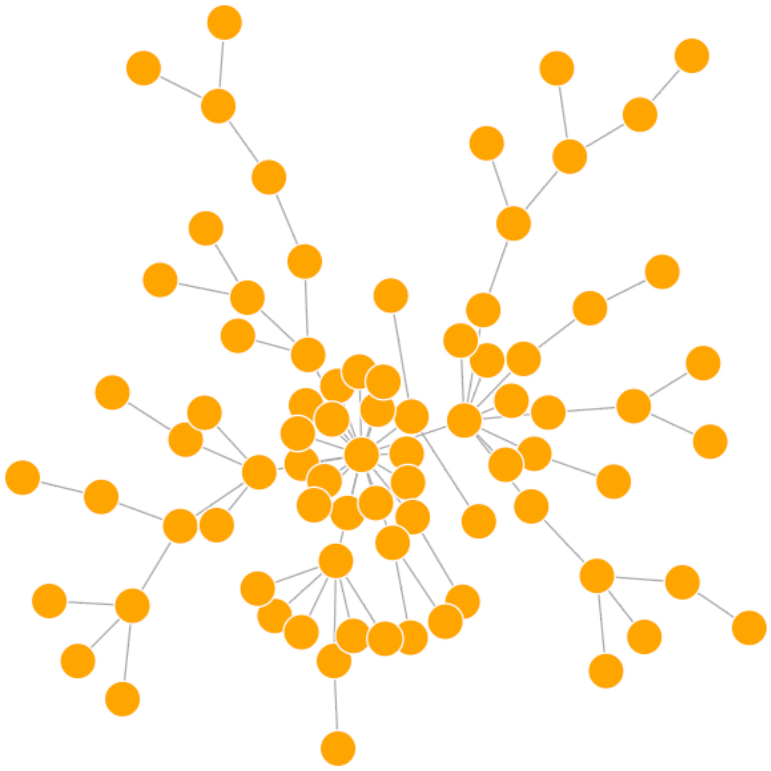
```
dev.off() # shut off the  graphic device to clear the two-figu
re configuration.
```

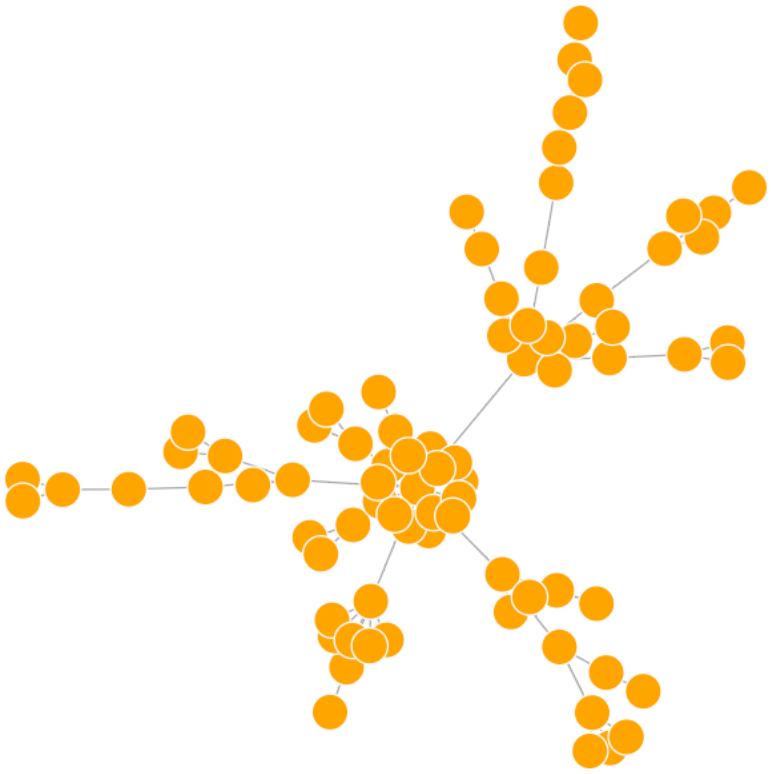
null device: 1

In [44]:

```
l <- layout.kamada.kawai(net.bg)
plot(net.bg, layout=l)

l <- layout.spring(net.bg, mass=.5)
plot(net.bg, layout=l)
```

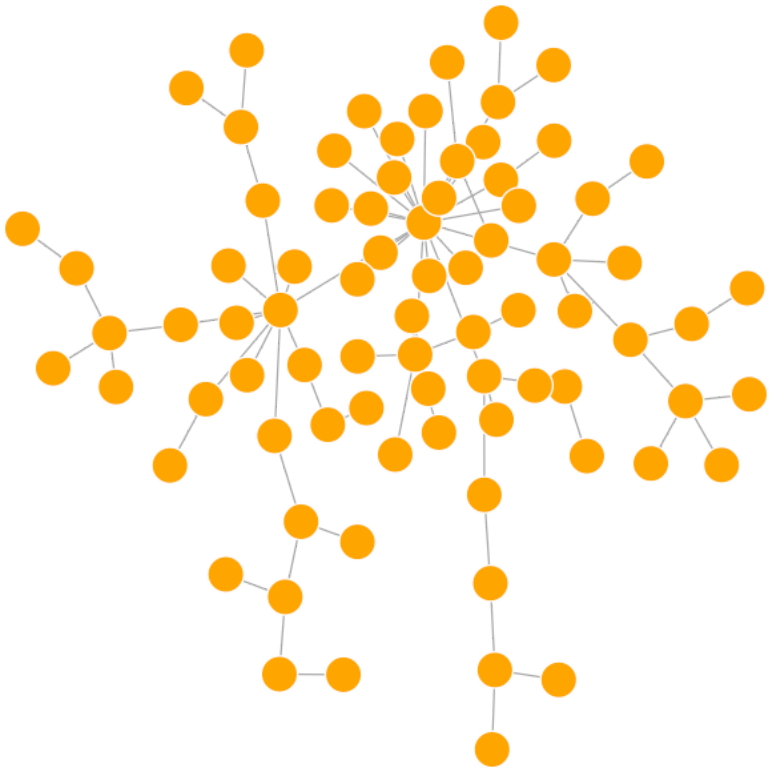
```
Warning message in layout.spring(net.bg, mass = 0.
5):
“Spring layout was removed, we use Fruchterman-Rein
gold instead.”
```





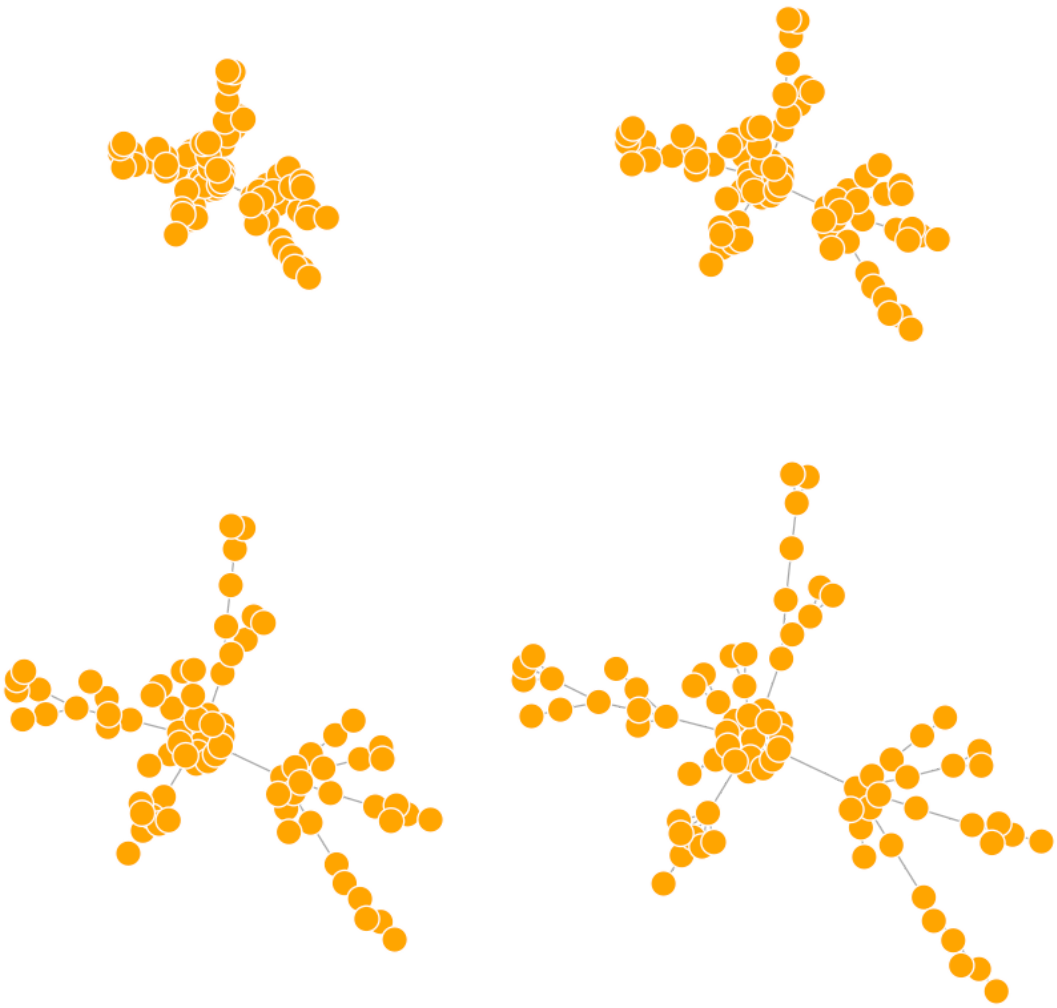
In [45]:

```
plot(net.bg, layout=layout.lgl)
```



In [46]:

```
l <- layout.fruchterman.reingold(net.bg)
l <- layout.norm(l, ymin=-1, ymax=1, xmin=-1, xmax=1)

par(mfrow=c(2,2), mar=c(0,0,0,0))
plot(net.bg, rescale=F, layout=l*0.4)
plot(net.bg, rescale=F, layout=l*0.6)
plot(net.bg, rescale=F, layout=l*0.8)
plot(net.bg, rescale=F, layout=l*1.0)
```

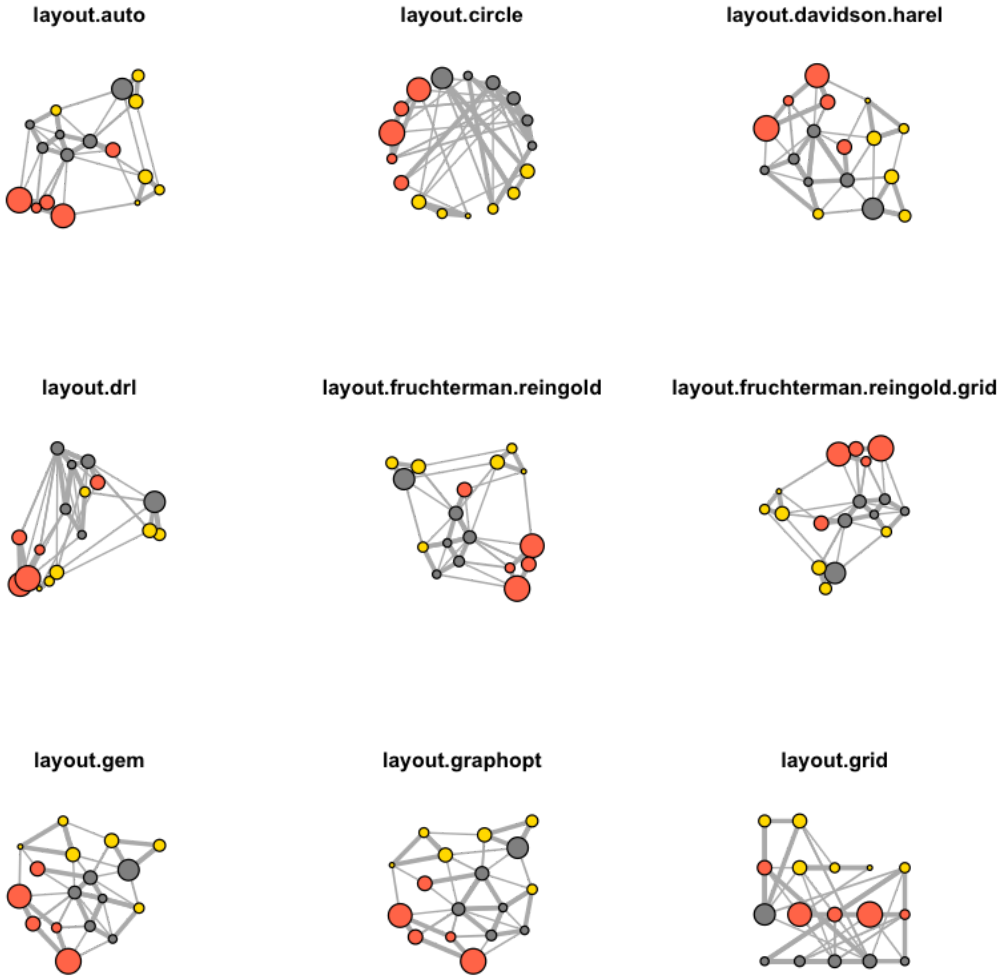


In [47]:

```
layouts <- grep("^layout\\.", ls("package:igraph"), value=TRUE
)
# Remove layouts that do not apply to our graph.
layouts <- layouts[!grepl("bipartite|merge|norm|sugiyama", lay
outs)]

par(mfrow=c(3,3))

for (layout in layouts) {
  print(layout)
  l <- do.call(layout, list(net))
  plot(net, edge.arrow.mode=0, layout=l, main=layout) }

dev.off()
```

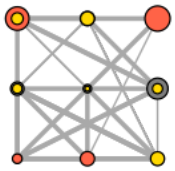
```
[1] "layout.auto"
[1] "layout.circle"
[1] "layout.davidson.harel"
[1] "layout.drl"
[1] "layout.fruchterman.reingold"
[1] "layout.fruchterman.reingold.grid"

Warning message in layout.fruchterman.reingold.grid
(structure(list(17, TRUE, c(0, :
“Grid Fruchterman-Reingold layout was removed,
we use Fruchterman-Reingold instead.”

[1] "layout.gem"
[1] "layout.graphopt"
[1] "layout.grid"
[1] "layout.grid.3d"

Warning message:
“layout.grid.3d is deprecated from
igraph 0.8.0, please use layout_on_grid instead”

[1] "layout.kamada.kawai"
[1] "layout.lgl"
[1] "layout.mds"
[1] "layout.random"
[1] "layout.reingold.tilford"

Warning message in layout_as_tree(structure(list(1
7, TRUE, c(0, 0, 0, 0, 1, 1, 1, :
“At structural_properties.c:3346 :graph contains a
cycle, partial result is returned”

[1] "layout.sphere"
[1] "layout.spring"

Warning message in layout.spring(structure(list(17,
TRUE, c(0, 0, 0, 0, 1, 1, 1, :
“Spring layout was removed, we use Fruchterman-Rein
gold instead.”

[1] "layout.star"
[1] "layout.svd"
```

```
Warning message in layout.svd(structure(list(17, TR
UE, c(0, 0, 0, 0, 1, 1, 1, 1, :
“SVD layout was removed, we use Fruchterman-Reingol
d instead.”
```



null device: 1

layout.grid.3d

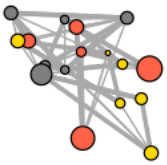


layout.kamada.kawai

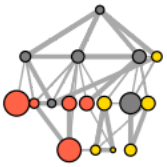
layout.lgl

layout.mds

layout.random

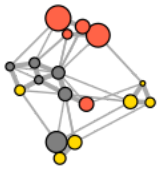


layout.reingold.tilford



layout.sphere

layout.spring



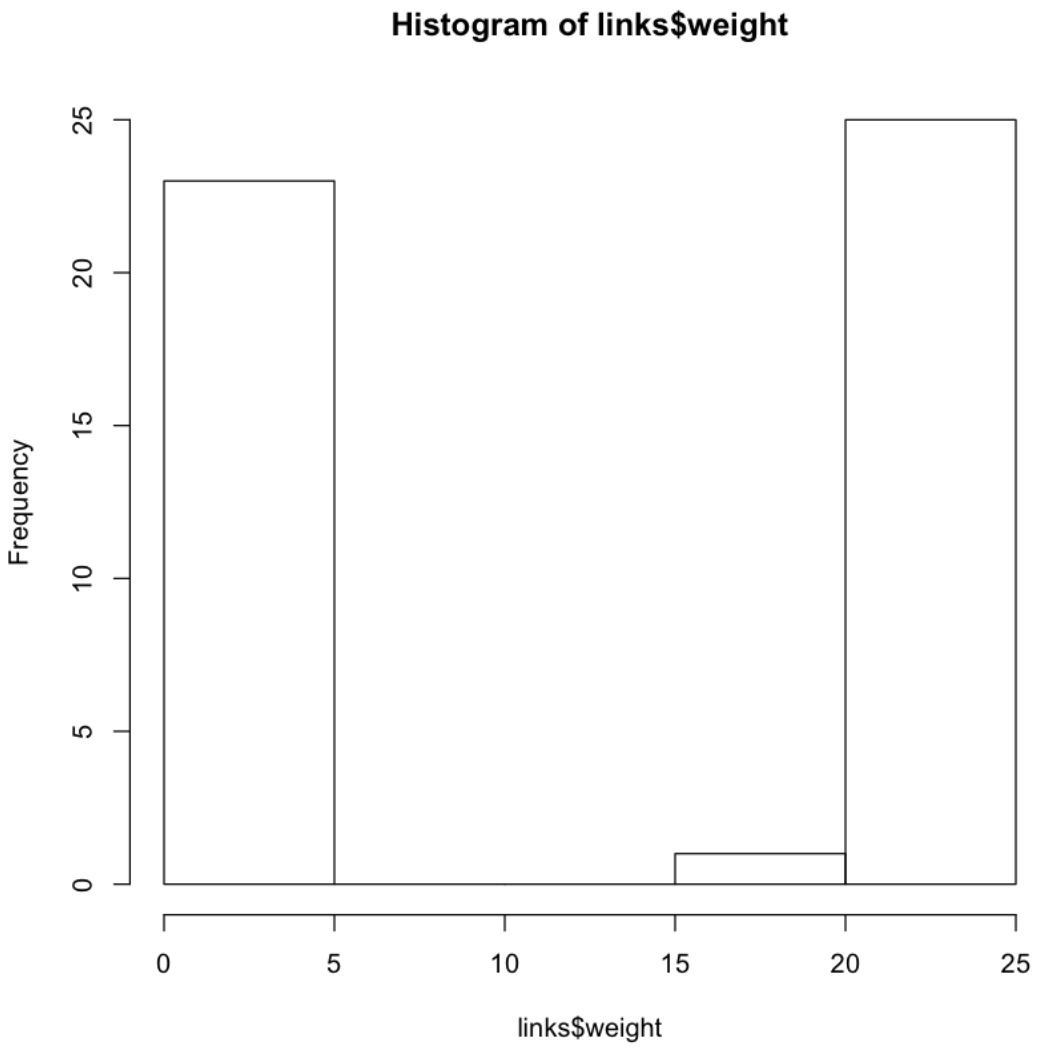
layout.star

In [48]:

```
hist(links$weight)
mean(links$weight)
sd(links$weight)
```

12.4081632653061

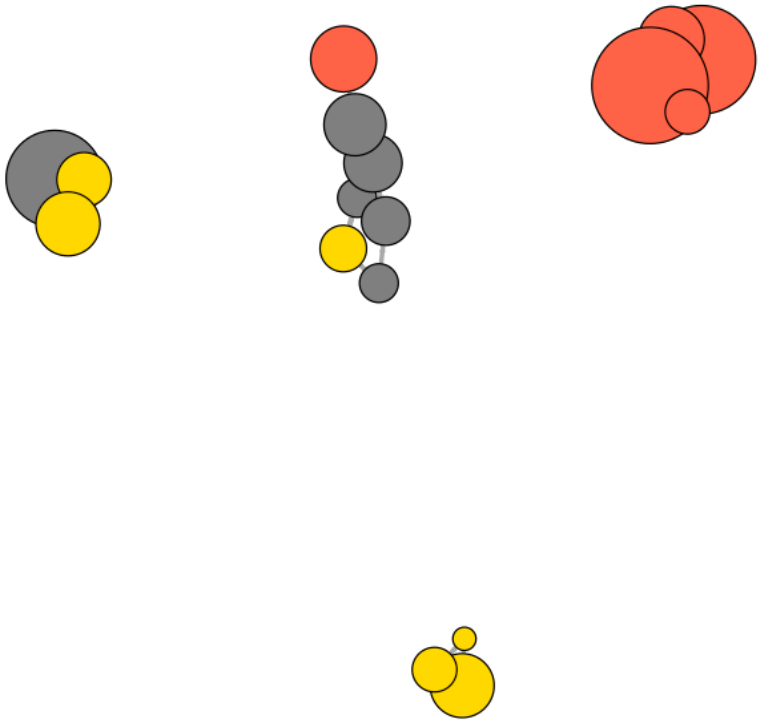
9.90563469139943



In [49]:

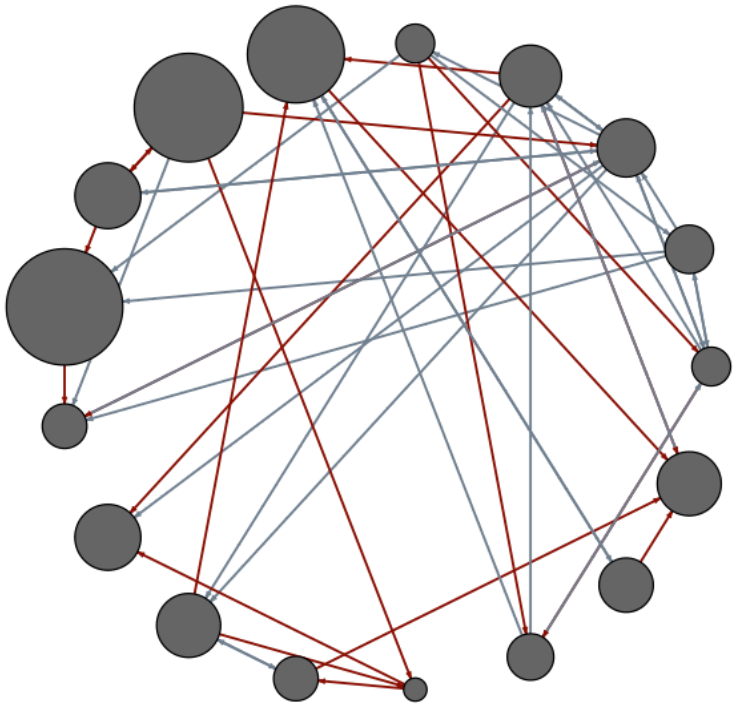
```
cut.off <- mean(links$weight)
net.sp <- igraph::delete.edges(net, E(net)[weight<cut.off])
l <- layout.fruchterman.reingold(net.sp, repulserad=vcount(net
)^2.1)
plot(net.sp, layout=l)
```

```
Warning message in layout_with_fr(structure(list(1
7, TRUE, c(0, 0, 0, 0, 1, 1, 2, :
“Argument `repulserad' is deprecated and has no eff
ect”
```



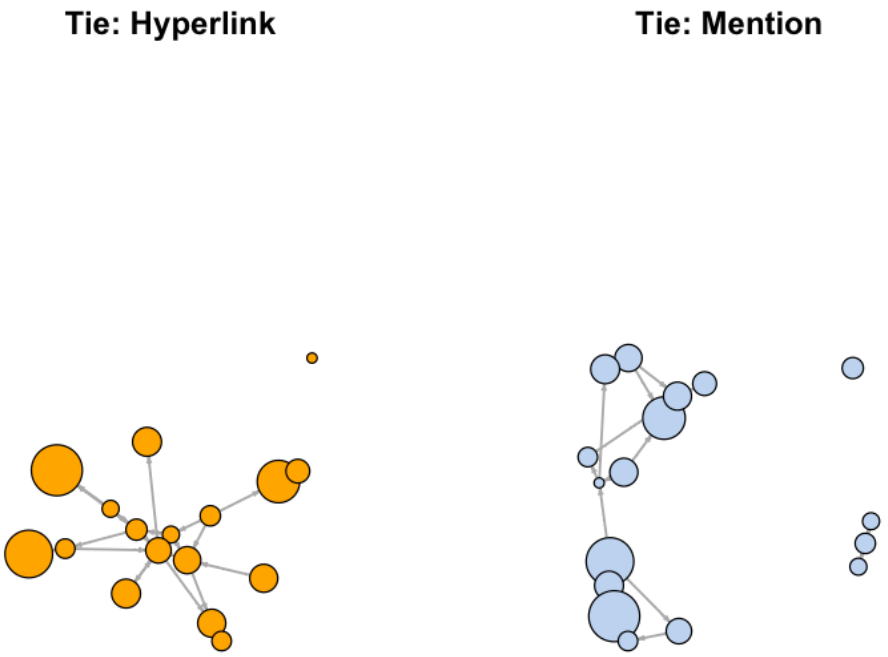
In [50]:

```
E(net)$width <- 1.5
plot(net, edge.color=c("dark red", "slategrey")[(E(net)$type==
"hyperlink")+1],
      vertex.color="gray40", layout=layout.circle)
```



In [51]:

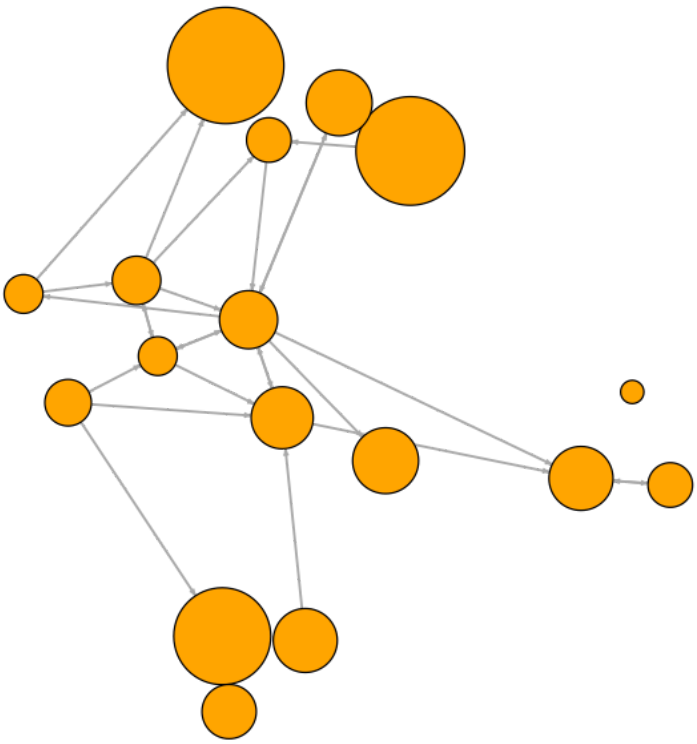
```
net.m <- net - E(net)[E(net)$type=="hyperlink"] # another way
 to delete edges
net.h <- net - E(net)[E(net)$type=="mention"]

par(mfrow=c(1,2))
plot(net.h, vertex.color="orange", main="Tie: Hyperlink")
plot(net.m, vertex.color="lightsteelblue2", main="Tie: Mentio
n")
```



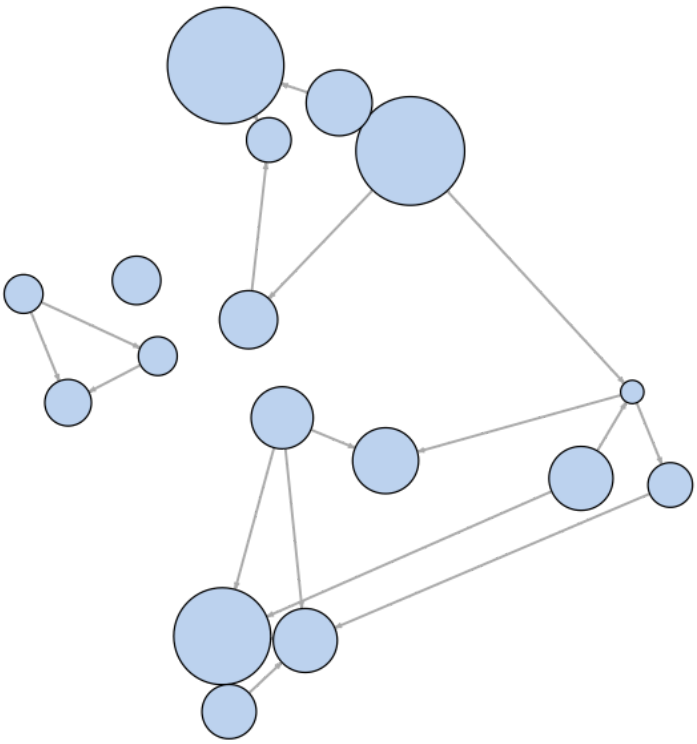
In [52]:

```
l <- layout.fruchterman.reingold(net)
plot(net.h, vertex.color="orange", layout=l, main="Tie: Hyperl
ink")
plot(net.m, vertex.color="lightsteelblue2", layout=l, main="Ti
e: Mention")
```

Tie: Hyperlink



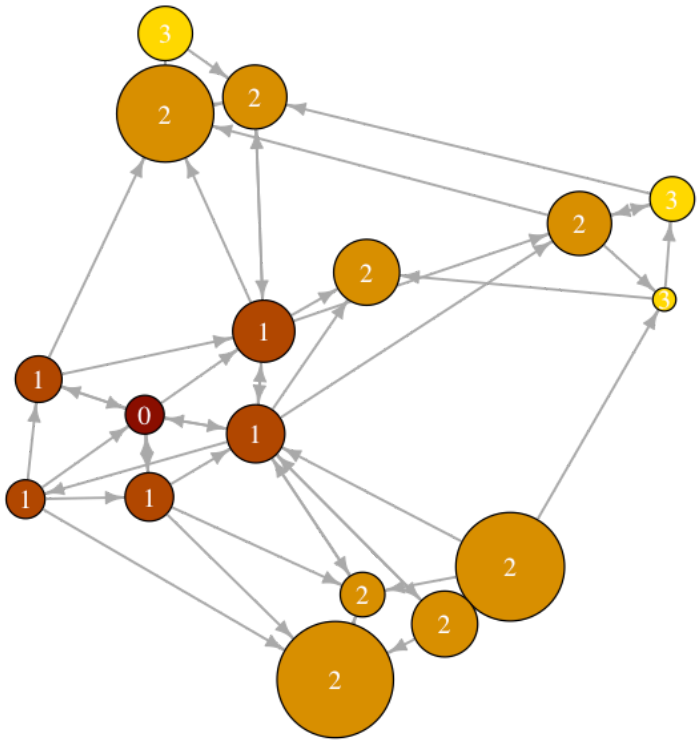
Tie: Mention



In [53]:

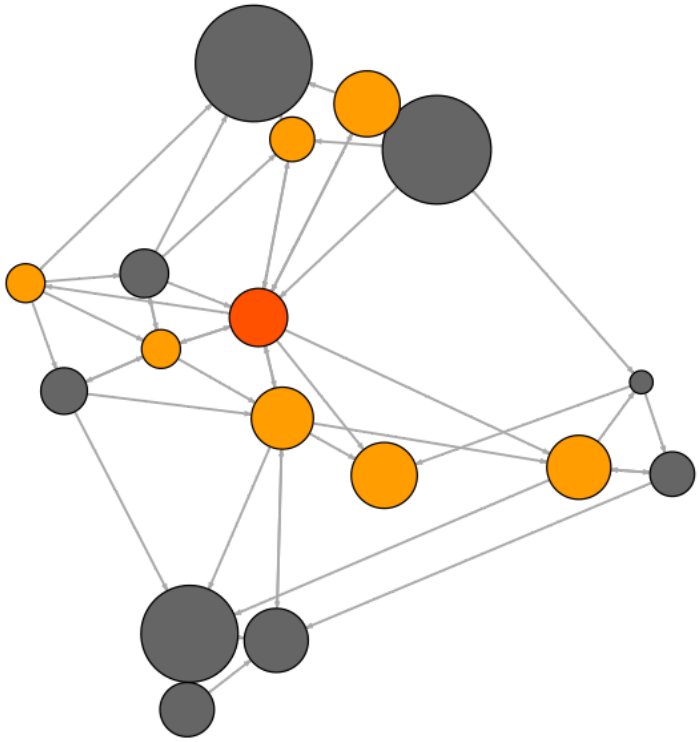
```
dist.from.NYT <- shortest.paths(net, algorithm="unweighted")[1
,]
oranges <- colorRampPalette(c("dark red", "gold"))
col <- oranges(max(dist.from.NYT)+1)[dist.from.NYT+1]

plot(net, vertex.color=col, vertex.label=dist.from.NYT, edge.a
rrow.size=.6,
     vertex.label.color="white")
```

```
Warning message in shortest.paths(net, algorithm =
"unweighted"):
“Unweighted algorithm chosen, weights ignored”
```



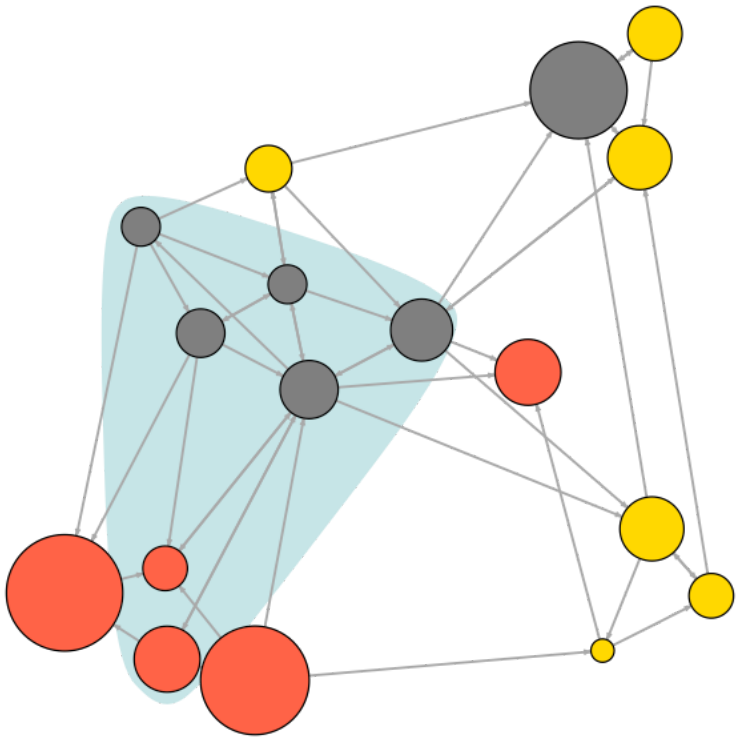
In [54]:

```
col <- rep("grey40", vcount(net))
col[V(net)$media=="Wall Street Journal"] <- "#ff5100"

neigh.nodes <- neighbors(net, V(net)[media=="Wall Street Journ
al"], mode="out")

col[neigh.nodes] <- "#ff9d00"
plot(net, vertex.color=col)
```



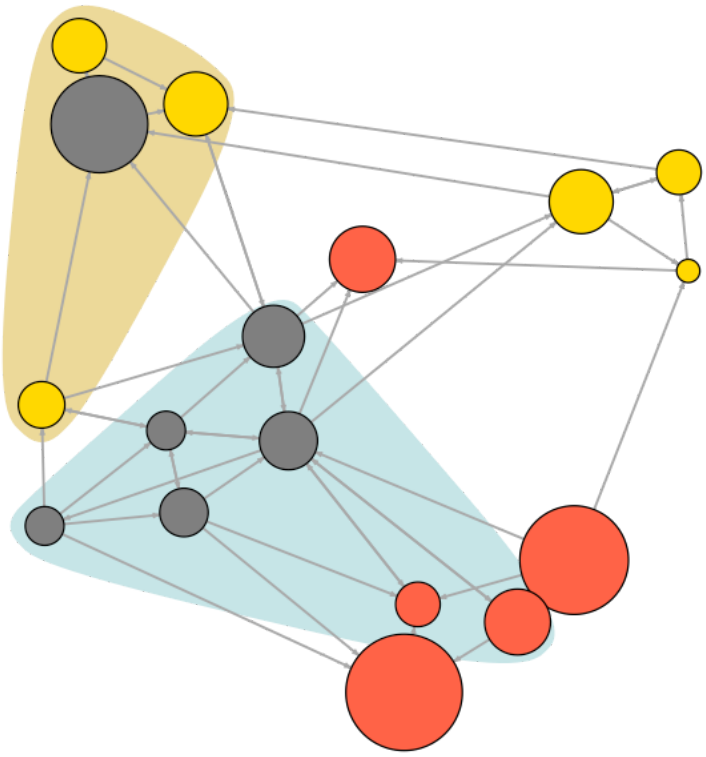
In [55]:

```
plot(net, mark.groups=c(1,4,5,8), mark.col="#C5E5E7", mark.bor
der=NA)
```



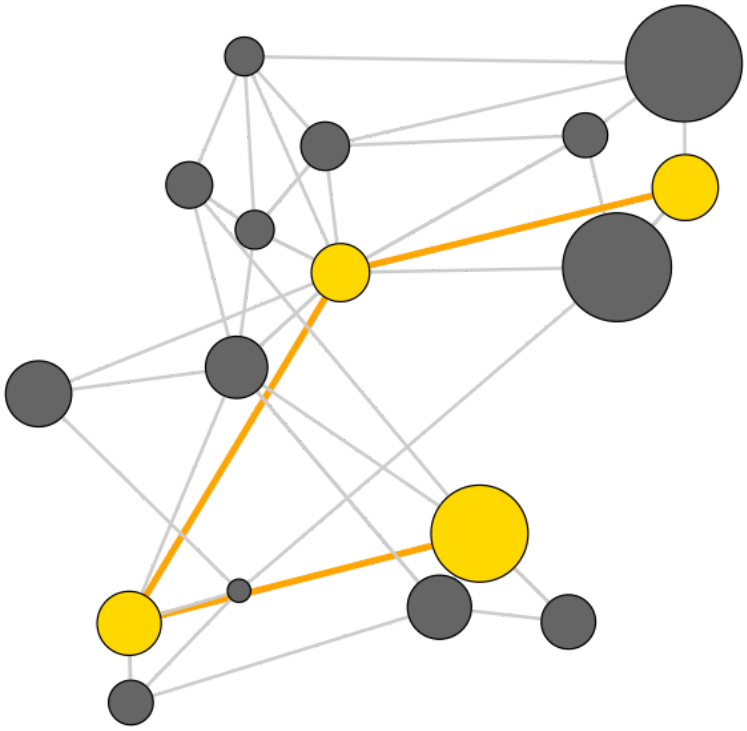
In [56]:

```
# Mark multiple groups:
plot(net, mark.groups=list(c(1,4,5,8), c(15:17)),
          mark.col=c("#C5E5E7","#ECD89A"), mark.border=NA)
```



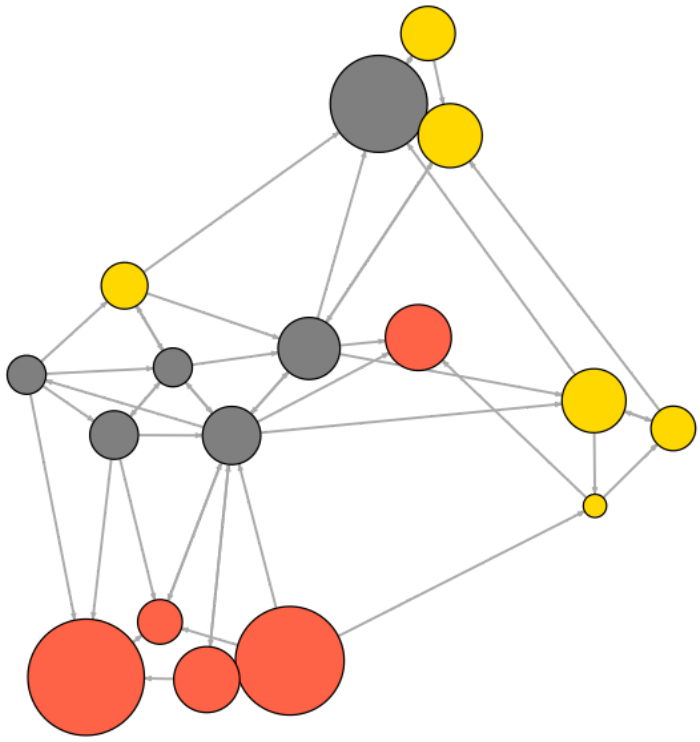
In [57]:

```
news.path <- get.shortest.paths(net, V(net)[media=="MSNBC"],
                                V(net)[media=="New York Post"
],
                                mode="all", output="both")


# Generate edge color variable:
ecol <- rep("gray80", ecount(net))
ecol[unlist(news.path$epath)] <- "orange"

# Generate edge width variable:
ew <- rep(2, ecount(net))
ew[unlist(news.path$epath)] <- 4

# Generate node color variable:
vcol <- rep("gray40", vcount(net))
vcol[unlist(news.path$vpath)] <- "gold"

plot(net, vertex.color=vcol, edge.color=ecol,
     edge.width=ew, edge.arrow.mode=0)
```



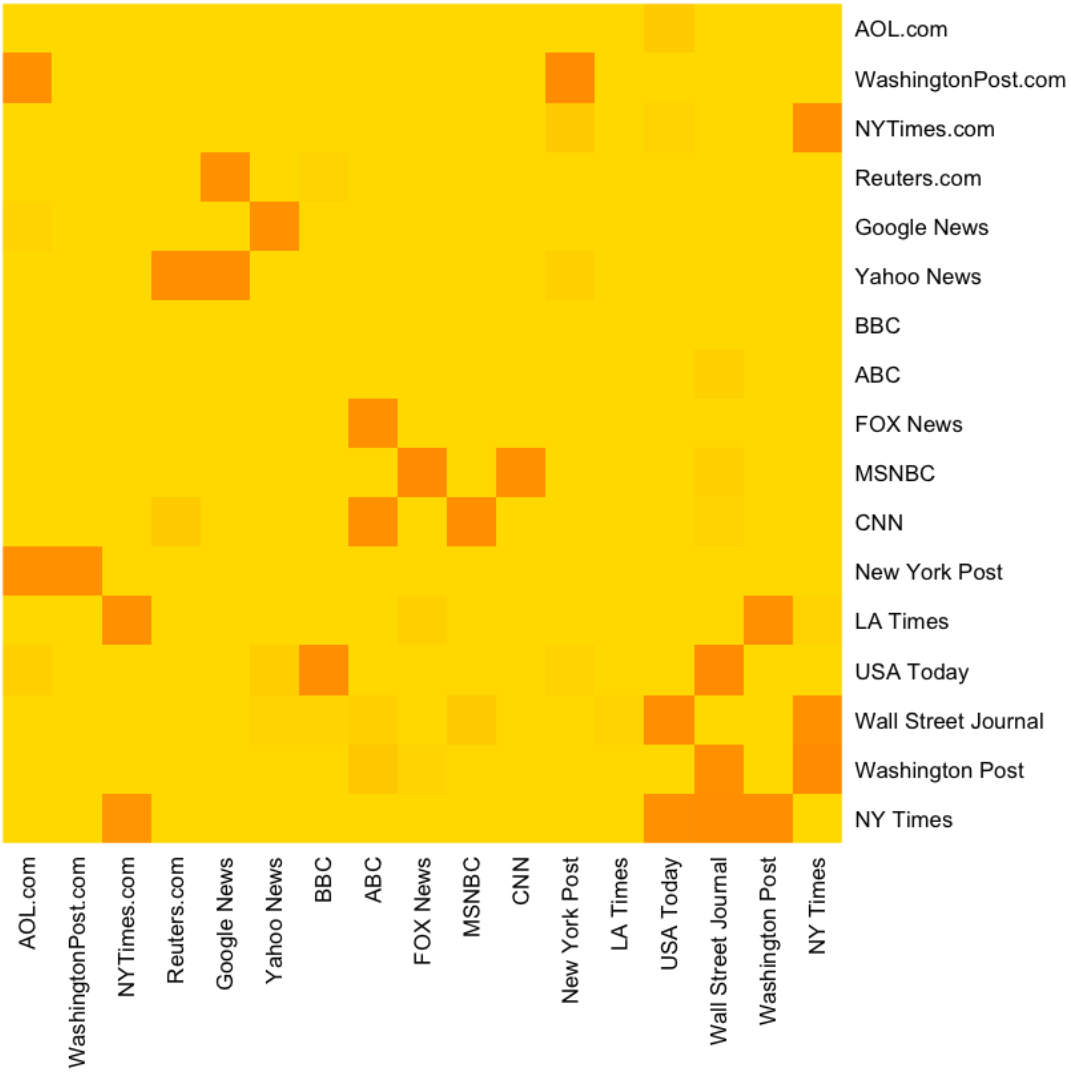
In [58]:

```
tkid <- tkplot(net) #tkid is the id of the tkplot that will op
en
l <- tkplot.getcoords(tkid) # grab the coordinates from tkplot
plot(net, layout=l)
```



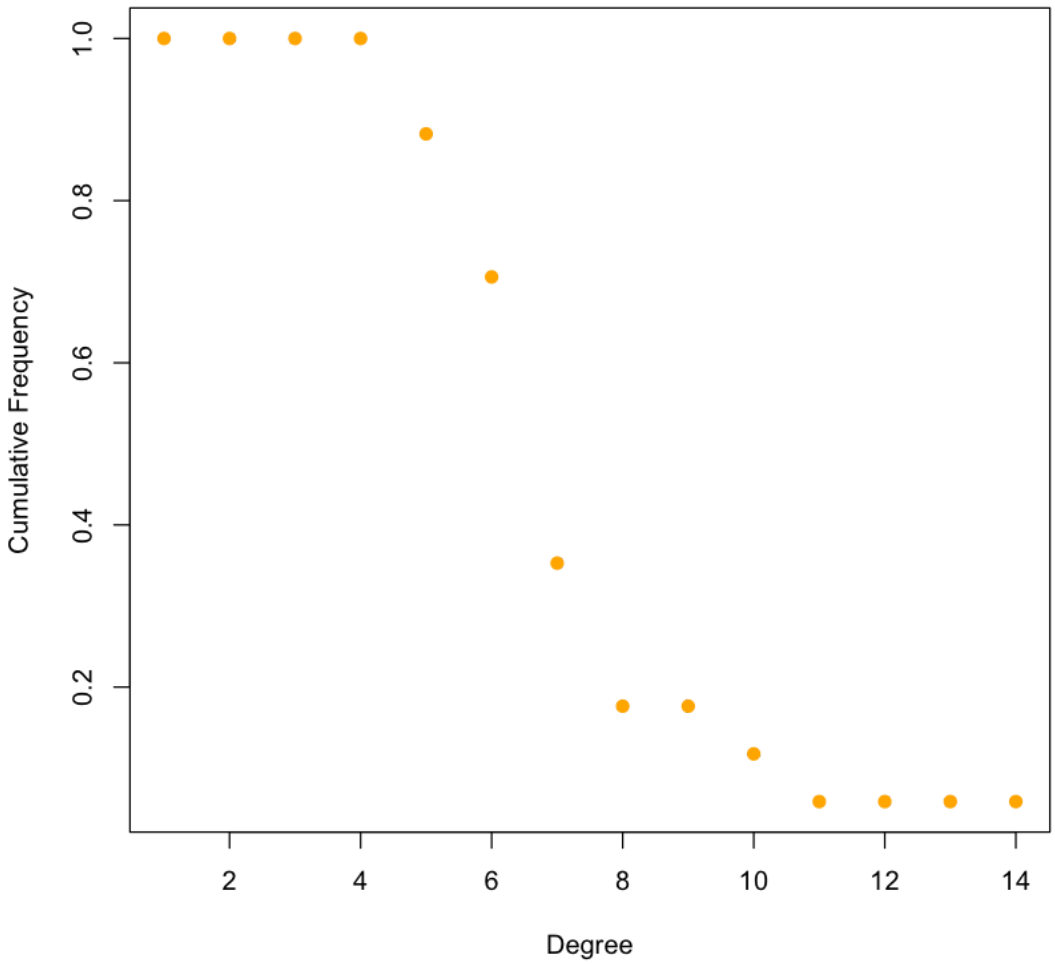
In [59]:

```
netm <- get.adjacency(net, attr="weight", sparse=F)
colnames(netm) <- V(net)$media
rownames(netm) <- V(net)$media

palf <- colorRampPalette(c("gold", "dark orange"))
heatmap(netm[,17:1], Rowv = NA, Colv = NA, col = palf(100),
        scale="none", margins=c(10,10) )
```



In [60]:

```
dd <- degree.distribution(net, cumulative=T, mode="all")
plot(dd, pch=19, cex=1, col="orange", xlab="Degree", ylab="Cum
ulative Frequency")
```



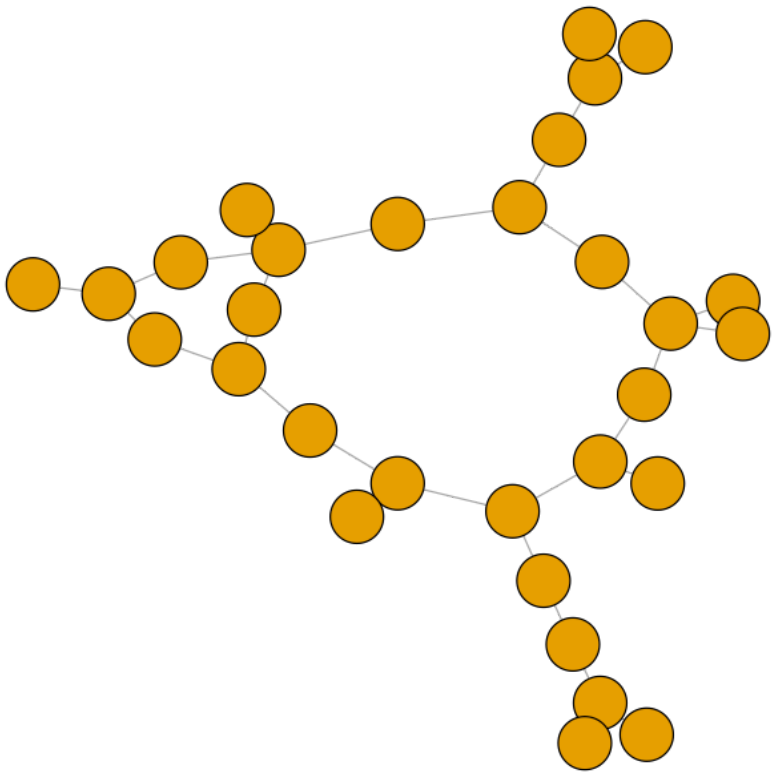
In [61]:

```
head(nodes2)
head(links2)

net2 <- graph.incidence(links2)
table(E(net2)$type)

plot(net2, vertex.label=NA)
```

id	media	media.type	media.name	audience.size
s01	NYT	1	Newspaper	20
s02	WaPo	1	Newspaper	25
s03	WSJ	1	Newspaper	30
s04	USAT	1	Newspaper	32
s05	LATimes	1	Newspaper	20
s06	CNN	2	TV	56

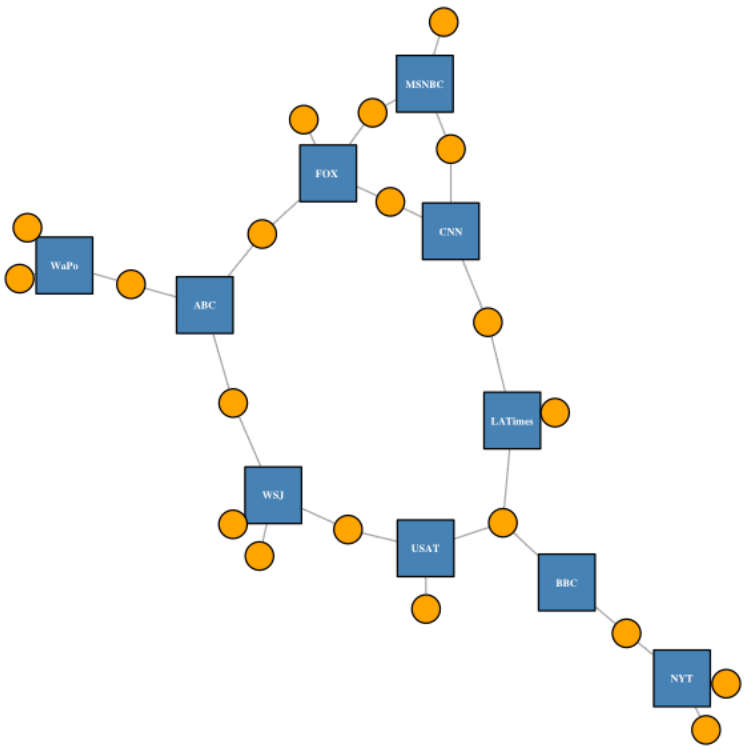
	U01	U02	U03	U04	U05	U06	U07	U08	U09	U10	U11	U
s01	1	1	1	0	0	0	0	0	0	0	0	0
s02	0	0	0	1	1	0	0	0	0	0	0	0
s03	0	0	0	0	0	1	1	1	1	0	0	0
s04	0	0	0	0	0	0	0	0	1	1	1	0
s05	0	0	0	0	0	0	0	0	0	0	1	1
s06	0	0	0	0	0	0	0	0	0	0	0	0

< table of extent 0 >



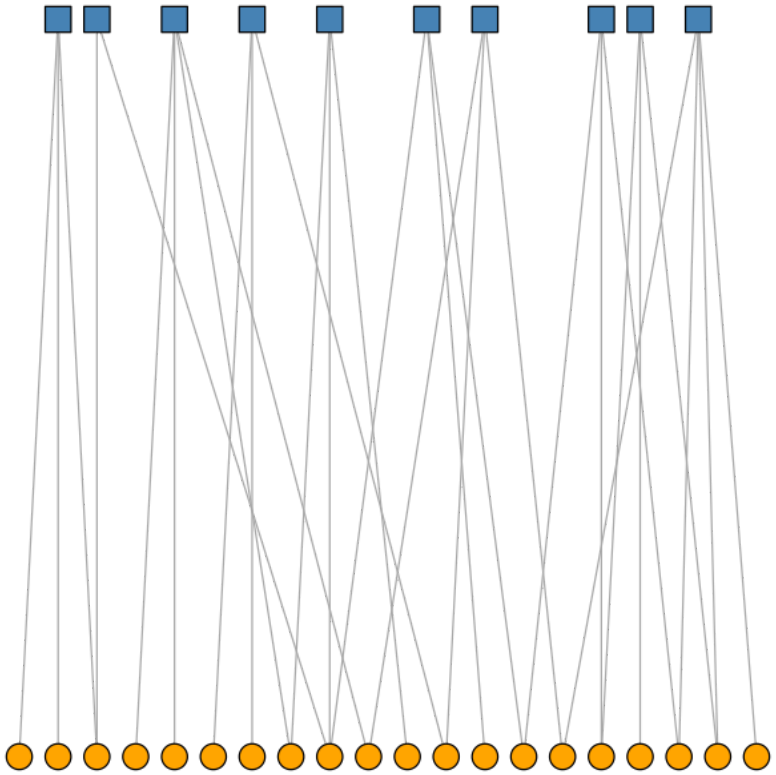
In [62]:

```
V(net2)$color <- c("steel blue", "orange")[V(net2)$type+1]
V(net2)$shape <- c("square", "circle")[V(net2)$type+1]
V(net2)$label <- ""
V(net2)$label[V(net2)$type==F] <- nodes2$media[V(net2)$type==F
]
V(net2)$label.cex=.4
V(net2)$label.font=2

plot(net2, vertex.label.color="white", vertex.size=(2-V(net2)$
type)*8)
```



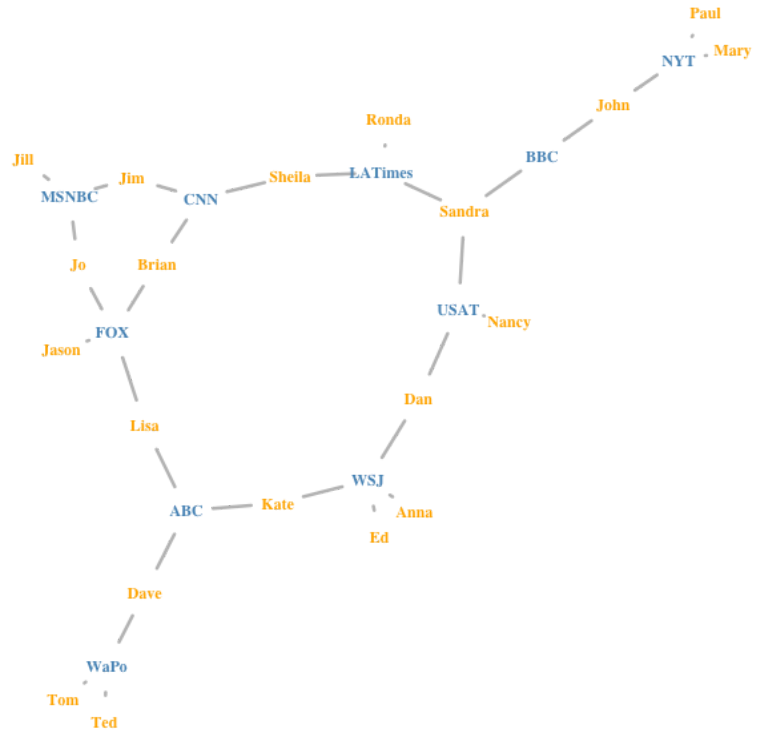
In [63]:

```
plot(net2, vertex.label=NA, vertex.size=7, layout=layout.bipar
tite)
```



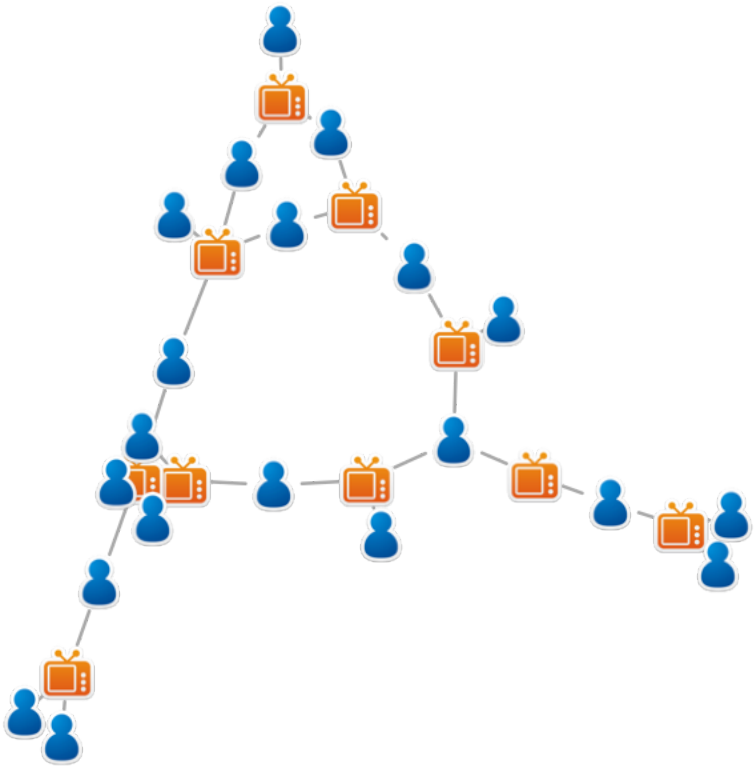
In [64]:

```
plot(net2, vertex.shape="none", vertex.label=nodes2$media,
     vertex.label.color=V(net2)$color, vertex.label.font=2,
     vertex.label.cex=.6, edge.color="gray70",  edge.width=2)
```



In [65]:

```
library(png)

img.1 <- readPNG("./Images/news.png")
img.2 <- readPNG("./Images/user.png")

V(net2)$raster <- list(img.1, img.2)[V(net2)$type+1]

plot(net2, vertex.shape="raster", vertex.label=NA,
     vertex.size=16, vertex.size2=16, edge.width=2)
```



In [66]:

```
detach(package:png)
detach(package:igraph)
```

In [67]:

```
library(network)

net3 <- network(links,  vertex.attr=nodes, matrix.type="edgeli
st",
                loops=F, multiple=F, ignore.eval = F)
```

In [68]:

```
net3[,]
net3 %n% "net.name" <- "Media Network" #  network attribute
net3 %v% "media"    # Node attribute
net3 %e% "type"     # Node attribute
```

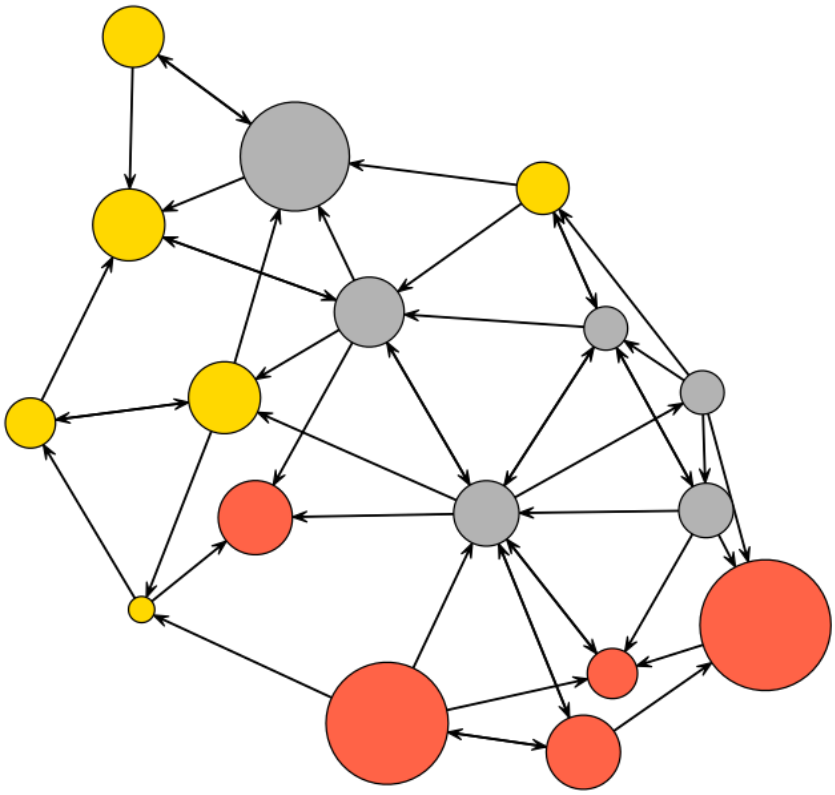
	s01	s02	s03	s04	s05	s06	s07	s08	s09	s10	s11	s12	s
s01	0	1	1	1	0	0	0	0	0	0	0	0	(
s02	1	0	1	0	0	0	0	0	1	1	0	0	(
s03	1	0	0	1	1	0	0	1	0	1	1	1	(
s04	0	0	1	0	0	1	0	0	0	0	1	1	(
s05	1	1	0	0	0	0	0	0	1	0	0	0	(
s06	0	0	0	0	0	1	0	0	0	0	0	0	(
s07	0	0	1	0	0	0	0	1	0	1	0	0	(
s08	0	0	1	0	0	0	1	0	1	0	0	0	(
s09	0	0	0	0	0	0	0	0	0	1	0	0	(
s10	0	0	1	0	0	0	0	0	0	0	0	0	(
s11	0	0	0	0	0	0	0	0	0	0	0	0	(
s12	0	0	0	0	0	1	0	0	0	0	0	0	'
s13	0	0	0	0	0	0	0	0	0	0	0	1	(
s14	0	0	0	0	0	0	0	0	0	0	1	0	'
s15	1	0	0	1	0	1	0	0	0	0	0	0	(
s16	0	0	0	0	0	1	0	0	0	0	0	0	(
s17	0	0	0	1	0	0	0	0	0	0	0	0	(

'NY Times' 'Washington Post' 'Wall Street Journal'
'USA Today' 'LA Times' 'New York Post' 'CNN'
'MSNBC' 'FOX News' 'ABC' 'BBC' 'Yahoo News'
'Google News' 'Reuters.com' 'NYTimes.com'
'WashingtonPost.com' 'AOL.com'

'hyperlink' 'hyperlink' 'hyperlink' 'mention'
'hyperlink' 'hyperlink' 'hyperlink' 'hyperlink'
'hyperlink' 'hyperlink' 'hyperlink' 'hyperlink'
'mention' 'hyperlink' 'hyperlink' 'hyperlink'
'mention' 'mention' 'hyperlink' 'mention' 'mention'
'hyperlink' 'hyperlink' 'mention' 'hyperlink'
'hyperlink' 'mention' 'mention' 'mention' 'hyperlink'
'mention' 'hyperlink' 'mention' 'mention' 'mention'
'hyperlink' 'mention' 'hyperlink' 'mention'
'hyperlink' 'mention' 'mention' 'mention' 'hyperlink'
'hyperlink' 'hyperlink' 'hyperlink' 'mention'
'hyperlink'

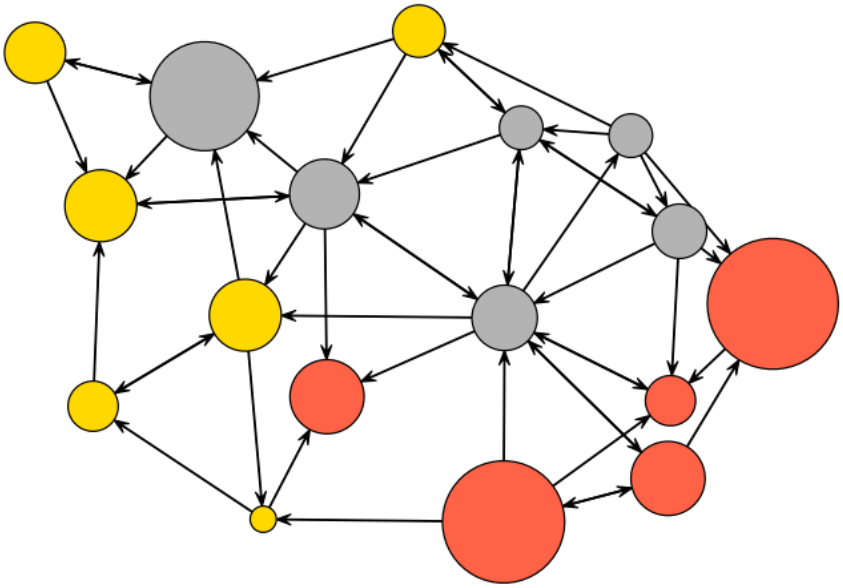
In [69]:

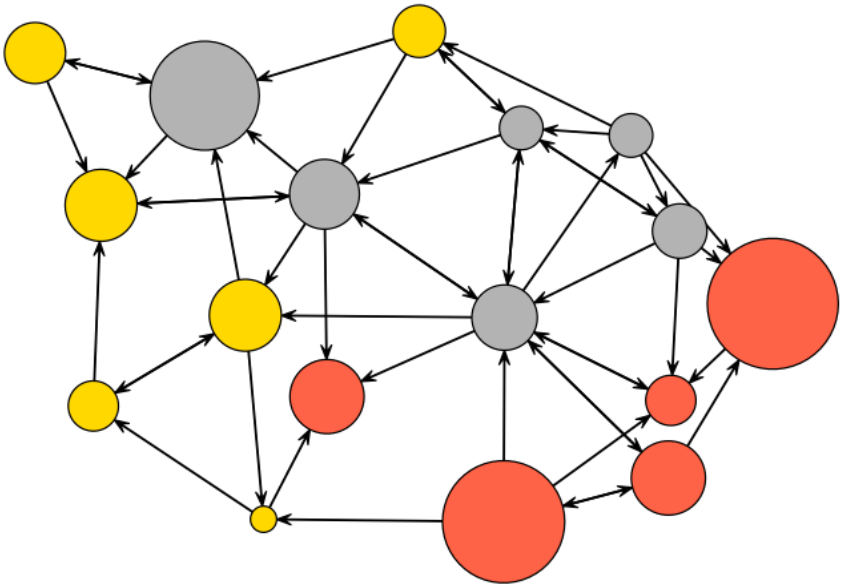
```
net3 %v% "col" <- c("gray70", "tomato", "gold")[net3 %v% "media.type"]
plot(net3, vertex.cex=(net3 %v% "audience.size")/7, vertex.col="col")
```



In [70]:

```
l <- plot(net3, vertex.cex=(net3 %v% "audience.size")/7, verte
x.col="col")
plot(net3, vertex.cex=(net3 %v% "audience.size")/7, vertex.col
="col", coord=l)
```





In [71]:

```
install.packages("networkD3")
```

```
The downloaded binary packages are in
        /var/folders/jw/knt_b30n31xgtwmrfn00sctm000
0gn/T//RtmpHjraEZ/downloaded_packages
```

In [72]:

```
library(networkD3)

el <- data.frame(from=as.numeric(factor(links$from))-1,
                 to=as.numeric(factor(links$to))-1 )
```

In [73]:

```
nl <- cbind(idn=factor(nodes$media, levels=nodes$media), nodes
)
```

In [74]:

```
forceNetwork(Links = el, Nodes = nl, Source="from", Target="t
o",
               NodeID = "idn", Group = "type.label",linkWidth
= 1,
               linkColour = "#afafaf", fontSize=12, zoom=T, le
gend=T,
               Nodesize=6, opacity = 0.8, charge=-300,
               width = 600, height = 400)
```

Tutorial based on input from:

https://rpubs.com/kateto/netviz (https://rpubs.com/kateto/netviz)