Install the package `igraph` and explore the capabilities

```
install.packages("igraph")
library(igraph) # Load the igraph package
```

Create networks

The code below generates an undirected graph with three edges. The numbers are interpreted as vertex IDs, so the edges are 1–>2, 2–>3, 3–>1.

```
g1 <- graph( edges=c(1,2, 2,3, 3, 1), n=3, directed=F )
plot(g1) # A simple plot of the network - we'll talk more about plots later
class(g1)
## [1] "igraph"
g1
## IGRAPH U--- 3 3 --
## + edges:
## [1] 1--2 2--3 1--3

# Now with 10 vertices, and directed by default:
g2 <- graph( edges=c(1,2, 2,3, 3, 1), n=10 )
plot(g2)
g2
## IGRAPH D--- 10 3 --
## + edges:
## [1] 1->2 2->3 3->1

g3 <- graph( c("John", "Jim", "Jim", "Jill", "Jill", "John")) # named
vertices
# When the edge list has vertex names, the number of nodes is not needed
plot(g3)
g3
## IGRAPH DN-- 3 3 --
## + attr: name (v/c)
## + edges (vertex names):
## [1] John->Jim  Jim ->Jill Jill->John
g4 <- graph( c("John", "Jim", "Jim", "Jack", "Jim", "Jack", "John", "John"),
             isolates=c("Jesse", "Janis", "Jennifer", "Justin") )

# In named graphs we can specify isolates by providing a list of their names.

plot(g4, edge.arrow.size=.5, vertex.color="gold", vertex.size=15,
     vertex.frame.color="gray", vertex.label.color="black",
     vertex.label.cex=0.8, vertex.label.dist=2, edge.curved=0.2)
```

Edge, vertex, and network attributes

Access vertices and edges:

```
E(g4) # The edges of the object
## + 4/4 edges (vertex names):
## [1] John->Jim  Jim ->Jack Jim ->Jack John->John
V(g4) # The vertices of the object
## + 7/7 vertices, named:
## [1] John     Jim      Jack     Jesse    Janis    Jennifer Justin
```

You can also examine the network matrix directly:

```
g4[]
## 7 x 7 sparse Matrix of class "dgCMatrix"
##          John Jim Jack Jesse Janis Jennifer Justin
## John        1   1    .     .     .        .      .
## Jim         .   .    2     .     .        .      .
## Jack        .   .    .     .     .        .      .
## Jesse       .   .    .     .     .        .      .
## Janis       .   .    .     .     .        .      .
## Jennifer    .   .    .     .     .        .      .
## Justin      .   .    .     .     .        .      .

g4[1,]
##     John      Jim     Jack    Jesse    Janis Jennifer   Justin
##        1        1        0        0        0        0        0
```

Add attributes to the network, vertices, or edges:

```
V(g4)$name # automatically generated when we created the network.
## [1] "John"     "Jim"      "Jack"     "Jesse"    "Janis"    "Jennifer"
## [7] "Justin"
V(g4)$gender <- c("male", "male", "male", "male", "female", "female", "male")
E(g4)$type <- "email" # Edge attribute, assign "email" to all edges
E(g4)$weight <- 10    # Edge weight, setting all existing edges to 10
```

Examine attributes:

```
edge_attr(g4)

## $type

## [1] "email" "email" "email" "email"

##

## $weight

## [1] 10 10 10 10
vertex_attr(g4)
## $name

## [1] "John"     "Jim"      "Jack"     "Jesse"    "Janis"    "Jennifer"
## [7] "Justin"

##

## $gender

## [1] "male"   "male"   "male"   "male"   "female" "female" "male"
graph_attr(g4)
## named list()
```

Another way to set attributes (you can similarly use `set_edge_attr()`, `set_vertex_attr()`, etc.):

```
g4 <- set_graph_attr(g4, "name", "Email Network")
g4 <- set_graph_attr(g4, "something", "A thing")
graph_attr_names(g4)
## [1] "name"      "something"
graph_attr(g4, "name")
## [1] "Email Network"
graph_attr(g4)
## $name
## [1] "Email Network"
##
## $something
## [1] "A thing"
g4 <- delete_graph_attr(g4, "something")
graph_attr(g4)
## $name
## [1] "Email Network"
plot(g4, edge.arrow.size=.5, vertex.label.color="black",
vertex.label.dist=1.5, vertex.color=c( "pink",
"skyblue")[1+(V(g4)$gender=="male")] )
```

The graph `g4` has two edges going from Jim to Jack, and a loop from John to himself. We can simplify our graph to remove loops & multiple edges between the same nodes. Use `edge.attr.comb` to indicate how edge attributes are to be combined - possible options include `sum`, `mean`, `prod` (product), `min`, `max`, `first`/`last` (selects the first/last edge's attribute). Option "ignore" says the attribute should be disregarded and dropped.

```
g4s <- simplify( g4, remove.multiple = T, remove.loops = F,

                 edge.attr.comb=c(weight="sum", type="ignore") )

plot(g4s, vertex.label.dist=1.5)
g4s
## IGRAPH DNW- 7 3 -- Email Network
## + attr: name (g/c), name (v/c), gender (v/c), weight (e/n)
## + edges (vertex names):
## [1] John->John John->Jim  Jim ->Jack
```

The description of an igraph object starts with up to 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 (7 5) 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

Specific graphs

Empty graph

```
eg <- make_empty_graph(40)

plot(eg, vertex.size=10, vertex.label=NA)
```

Full graph

```
fg <- make_full_graph(40)

plot(fg, vertex.size=10, vertex.label=NA)
```

Simple star graph

```
st <- make_star(40)

plot(st, vertex.size=10, vertex.label=NA)
```

Tree graph

```
tr <- make_tree(40, children = 3, mode = "undirected")

plot(tr, vertex.size=10, vertex.label=NA)
```

Ring graph

```
rn <- make_ring(40)

plot(rn, vertex.size=10, vertex.label=NA)
```

Reading network data from files

In the following sections of the 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 analyses and visualizations we will generate apply to medium and large-scale networks.

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” (see Moodle).

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

Examine the data:

```
head(nodes)
head(links)
nrow(nodes); length(unique(nodes$id))
nrow(links); nrow(unique(links[,c("from", "to")]))
```

Notice that there are more links than unique from-to combinations. That means we have cases in the data where there are multiple links between the same two nodes. We will collapse all links of the same type between the same two nodes by summing their weights, using `aggregate()` by “from”, “to”, & “type”. We don’t use `simplify()` here so as not to collapse different link types.

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

DATASET 2: matrix

Two-mode or bipartite graphs have two different types of actors and links that go across, but not within each type. Our second media example is a network of that kind, examining links between news sources and their consumers.

```
nodes2 <- read.csv("Dataset2-Media-User-Example-NODES.csv", header=T,
as.is=T)

links2 <- read.csv("Dataset2-Media-User-Example-EDGES.csv", header=T,
row.names=1)
```

Examine the data:

```
head(nodes2)

head(links2)
```

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

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

Turning networks into igraph objects

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.

Dataset 1

```
library(igraph)

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

class(net)
## [1] "igraph"
net
## IGRAPH 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 (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
```

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

```
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"
```

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

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

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)
```

You might notice that we 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").

If you need them, you can extract an edge list or a matrix from igraph networks.

```
as_edgelist(net, names=T)
as_adjacency_matrix(net, attr="weight")
```

Or data frames describing nodes and edges:

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

Dataset 2

As we have seen above, this time the edges of the network are in a matrix format. We can read those into a graph object using `graph_from_incidence_matrix()`. In igraph, bipartite networks have a node attribute called `type` that is FALSE (or 0) for vertices in one mode and TRUE (or 1) for those in the other mode.

```
head(nodes2)
##    id   media media.type media.name audience.size
## 1 s01     NYT          1  Newspaper            20
## 2 s02    WaPo          1  Newspaper            25
## 3 s03     WSJ          1  Newspaper            30
## 4 s04    USAT          1  Newspaper            32
## 5 s05 LATimes          1  Newspaper            20
## 6 s06     CNN          2         TV            56
head(links2)
##     U01 U02 U03 U04 U05 U06 U07 U08 U09 U10 U11 U12 U13 U14 U15 U16 U17
## s01   1   1   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0
## s02   0   0   0   1   1   0   0   0   0   0   0   0   0   0   0   0   0
## s03   0   0   0   0   0   1   1   1   1   0   0   0   0   0   0   0   0
## s04   0   0   0   0   0   0   0   0   1   1   1   0   0   0   0   0   0
## s05   0   0   0   0   0   0   0   0   0   0   1   1   1   0   0   0   0
## s06   0   0   0   0   0   0   0   0   0   0   0   0   1   1   0   0   1

##     U18 U19 U20
## s01   0   0   0
## s02   0   0   1
## s03   0   0   0
## s04   0   0   0
## s05   0   0   0
## s06   0   0   0
net2 <- graph_from_incidence_matrix(links2)
table(V(net2)$type)
##

## FALSE  TRUE
##    10    20
```

To transform a one-mode network matrix into an igraph object, use instead `graph_from_adjacency_matrix()`.

We can also easily generate bipartite projections for the two-mode network: (co-memberships are easy to calculate by multiplying the network matrix by its transposed matrix, or using igraph's `bipartite.projection()` function).

```
net2.bp <- bipartite.projection(net2)
```

We can calculate the projections manually as well:

```
   as_incidence_matrix(net2)  %*% t(as_incidence_matrix(net2))

 t(as_incidence_matrix(net2)) %*%   as_incidence_matrix(net2)
plot(net2.bp$proj1, vertex.label.color="black", vertex.label.dist=1,

     vertex.size=7, vertex.label=nodes2$media[!is.na(nodes2$media.type)])

plot(net2.bp$proj2, vertex.label.color="black", vertex.label.dist=1,

     vertex.size=7, vertex.label=nodes2$media[ is.na(nodes2$media.type)])
```

Homework assignment

1. Obtain the degrees of all nodes from the adjacency matrix of an undirected graph
   - verify against the `degree` function
2. Obtain the neighbors of a given node provided an adjacency matrix of an undirected graph
   - verify against the `neighbors` function
3. Write a function to determine the second neighborhood a given node v in an undirected graph and test it against the `neighbors` function.
4. Write a function to determine the most similar pair of nodes, i.e. nodes which share the largest number of neighbors relative to the number of nodes in the union of neighbors in a given undirected graph.