DSI Summer Workshops Series

June 14, 2018

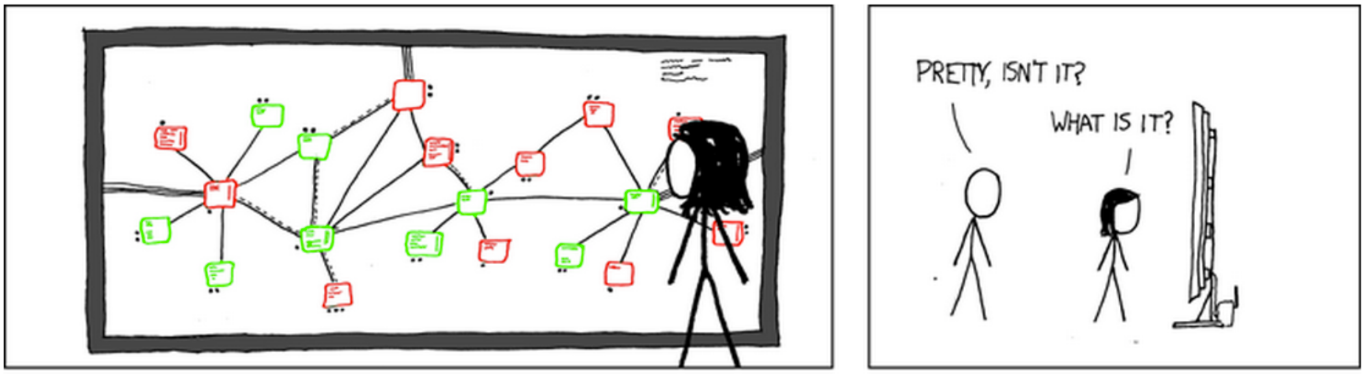
Peggy Lindner
Center for Advanced Computing & Data Science (CACDS)
Data Science Institute (DSI)
University of Houston
plindner@uh.edu

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

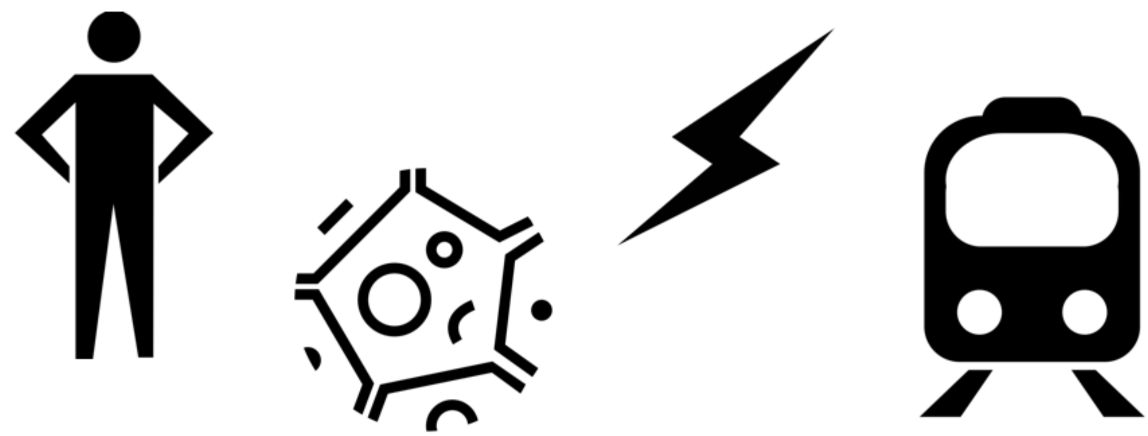
a group or system of interconnected people or things

a•nal•y•sis

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

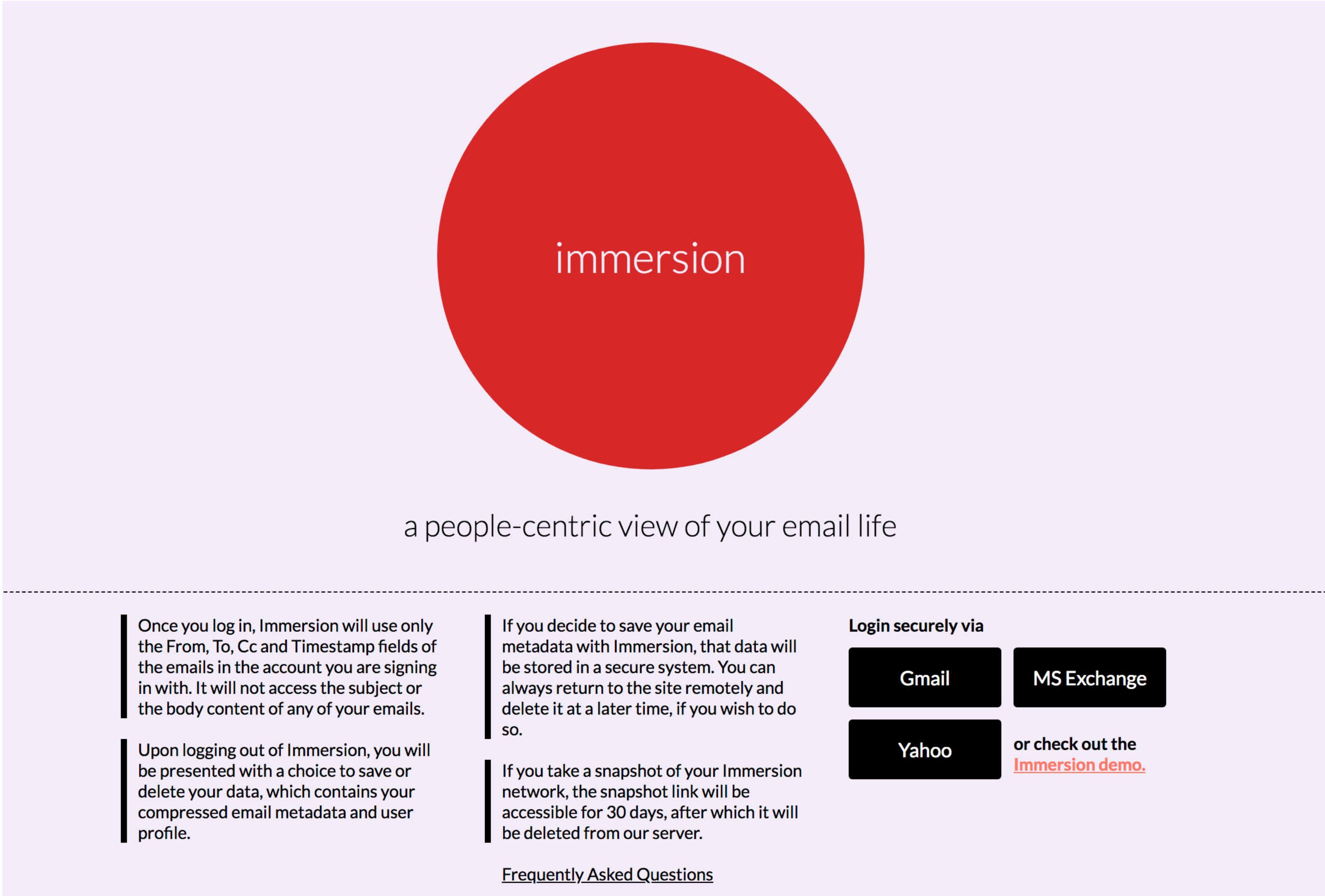
net•work a•nal•y•sis

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.



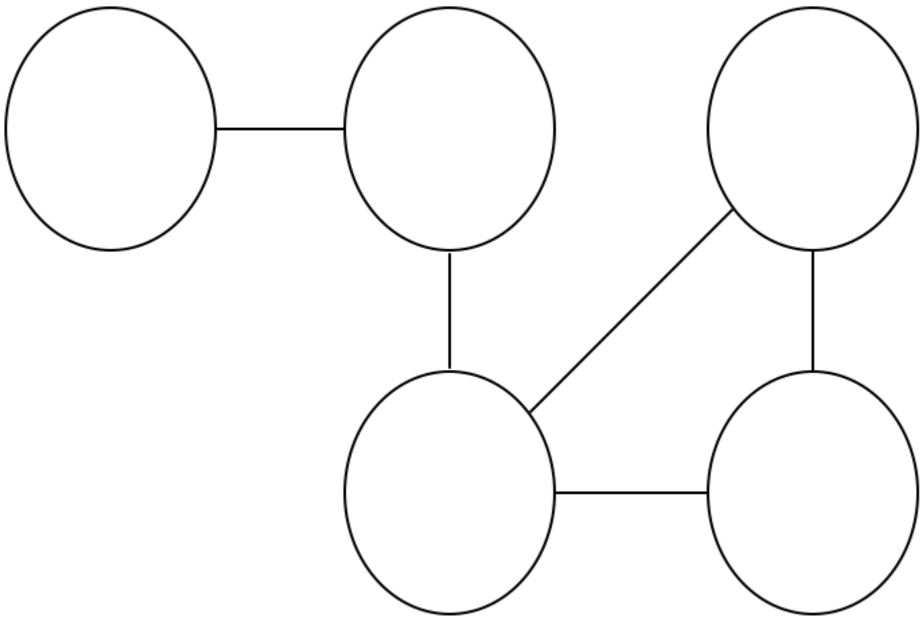
questions

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?

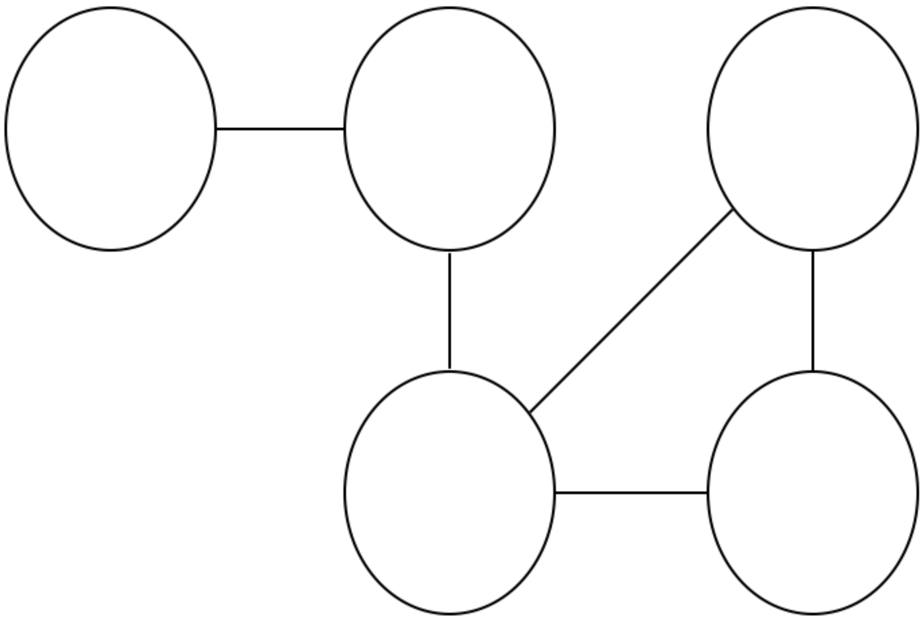


https://immersion.media.mit.edu/demo (https://immersion.media.mit.edu/demo)

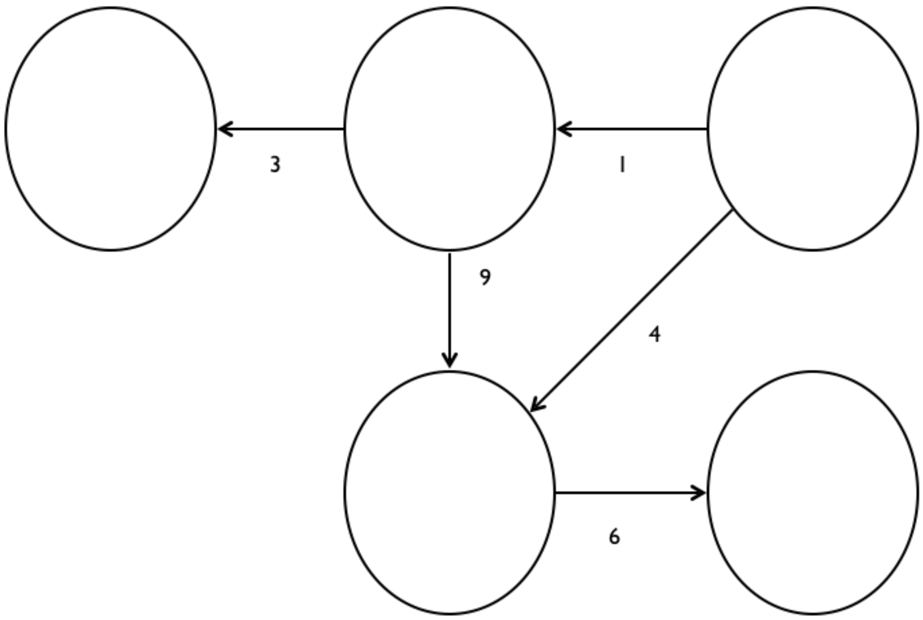
Graphs



nodes edges undirected

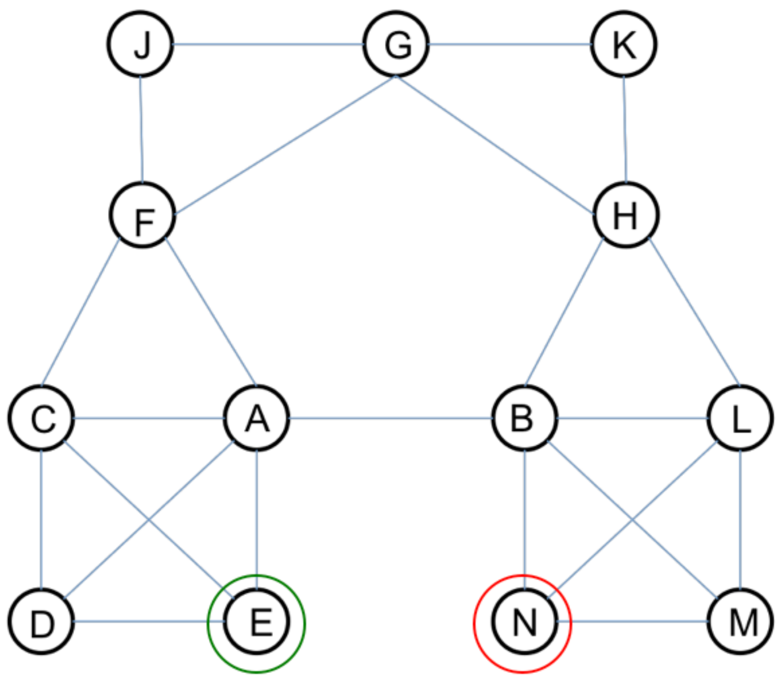


directed weighted



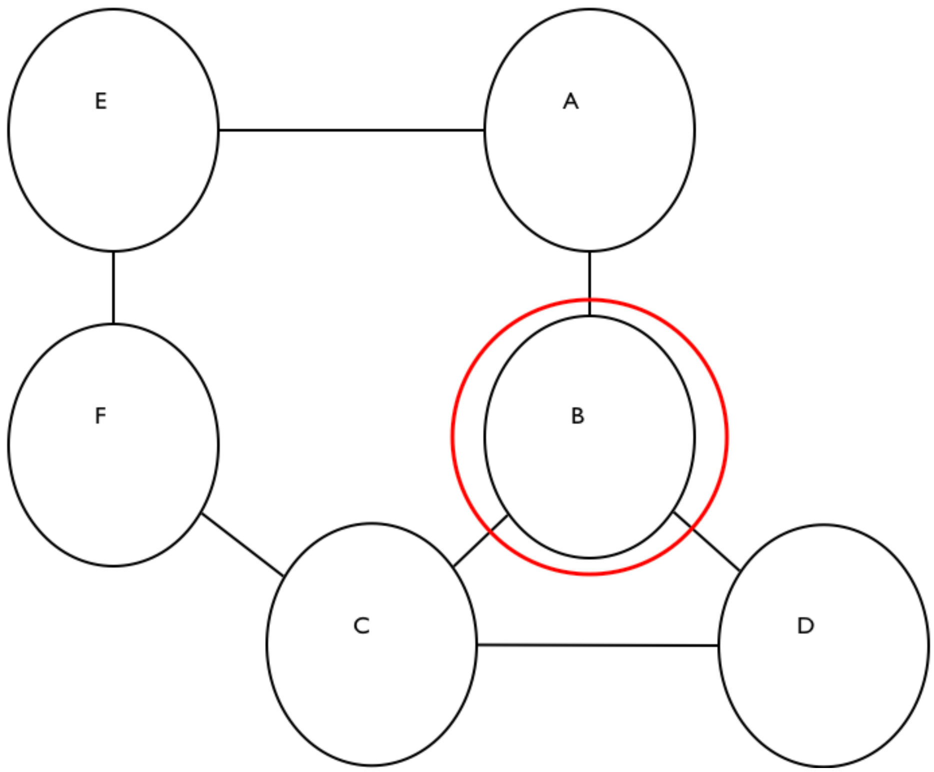
path

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



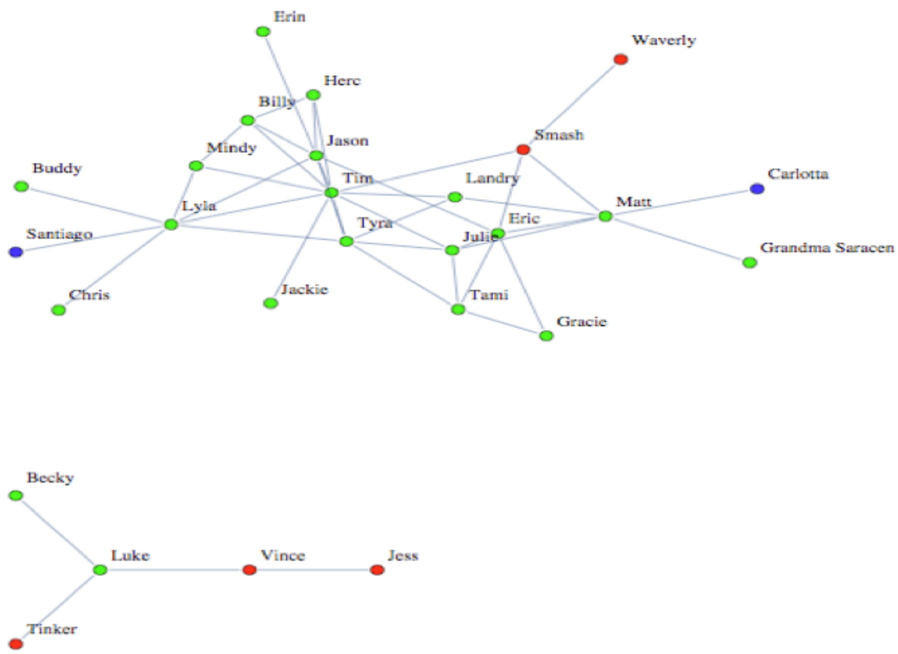
pivotal node

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



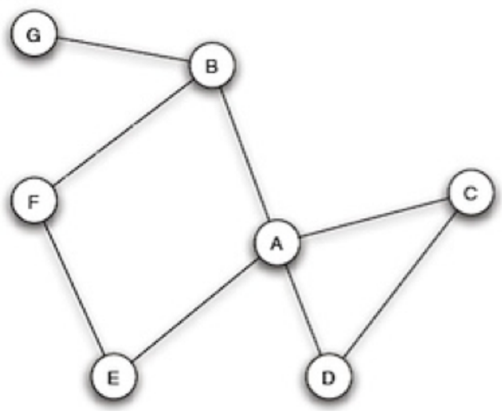
component

a subset of connected nodes

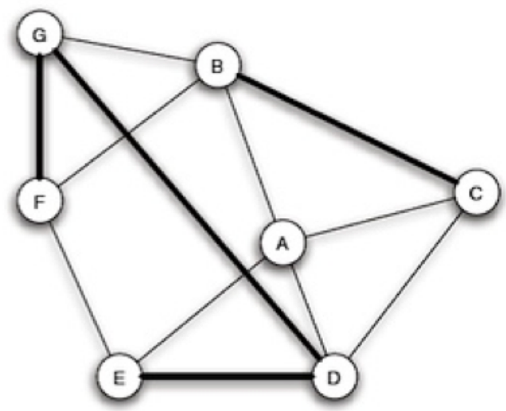


clustering coefficient

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



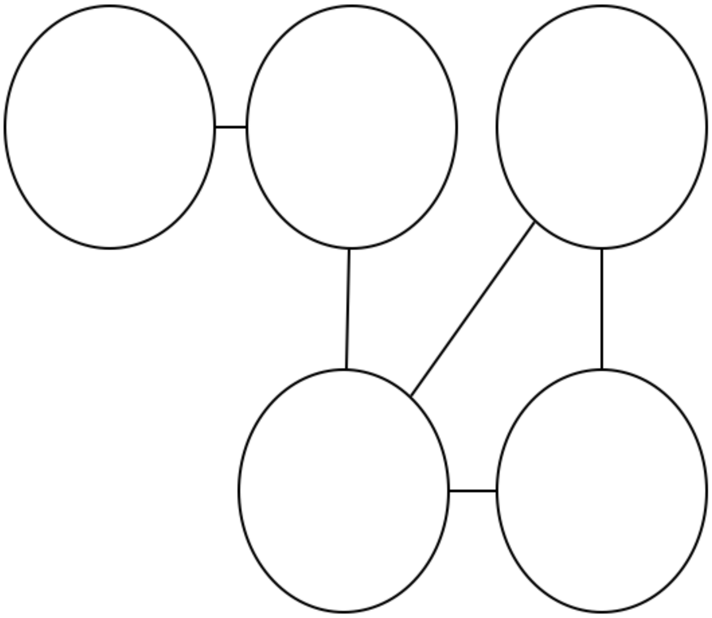
(a) *Before new edges form.*



(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 [ ]:

```
library(igraph)
library(network)
library(sna)
library(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 [ ]:

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

In [ ]:

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

In [ ]:

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

In [ ]:

```
# 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 [ ]:

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

In [ ]:

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


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

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 [ ]:

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

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 [ ]:

```
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]
```

First plot ...

In [ ]:

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

In [ ]:

```
#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 [ ]:

```
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 [ ]:

```
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 [ ]:

```
#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 [ ]:

```
#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 [ ]:

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

In [ ]:

```
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 [ ]:

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

In [ ]:

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

In [ ]:

```
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 [ ]:

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

In [ ]:

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

In [ ]:

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

In [ ]:

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

In [ ]:

```
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 [ ]:

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

In [ ]:

```
# 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 [ ]:

```
# 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 [ ]:

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

In [ ]:

```
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 [ ]:

```
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 [ ]:

```
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 [ ]:

```
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 [ ]:

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

In [ ]:

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

In [ ]:

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

In [ ]:

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

In [ ]:

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

In [ ]:

```
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)
```

In [ ]:

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

In [ ]:

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

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

In [ ]:

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

In [ ]:

```
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 [ ]:

```
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()
```

In [ ]:

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

In [ ]:

```
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)
```

In [ ]:

```
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 [ ]:

```
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 [ ]:

```
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")
```

In [ ]:

```
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")
```

In [ ]:

```
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 [ ]:

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

In [ ]:

```
# 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 [ ]:

```
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 [ ]:

```
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 [ ]:

```
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 [ ]:

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

In [ ]:

```
head(nodes2)
head(links2)

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

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

In [ ]:

```
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 [ ]:

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

In [ ]:

```
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 [ ]:

```
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 [ ]:

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

In [ ]:

```
library(network)

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

In [ ]:

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

In [ ]:

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

In [ ]:

```
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 [ ]:

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

In [ ]:

```
library(networkD3)

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

In [ ]:

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

In [ ]:

```
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)