

What are social networks?

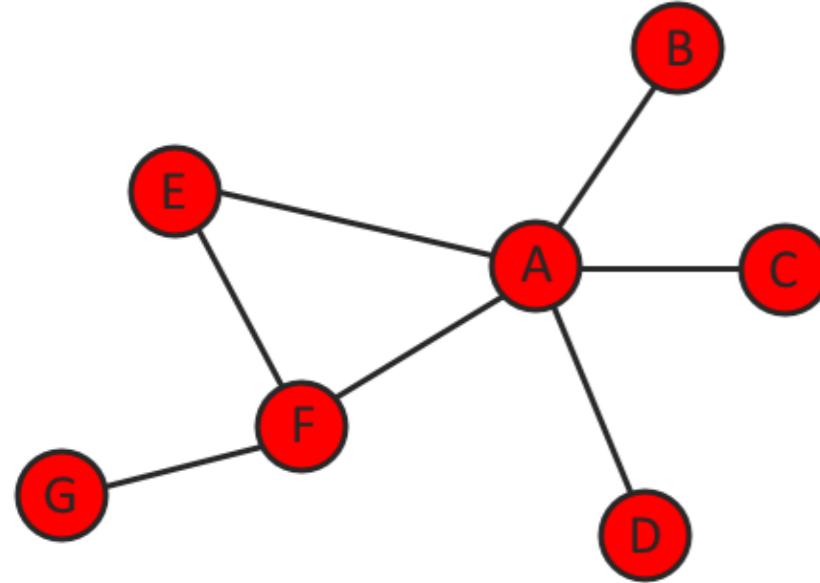
NETWORK ANALYSIS IN R



James Curley

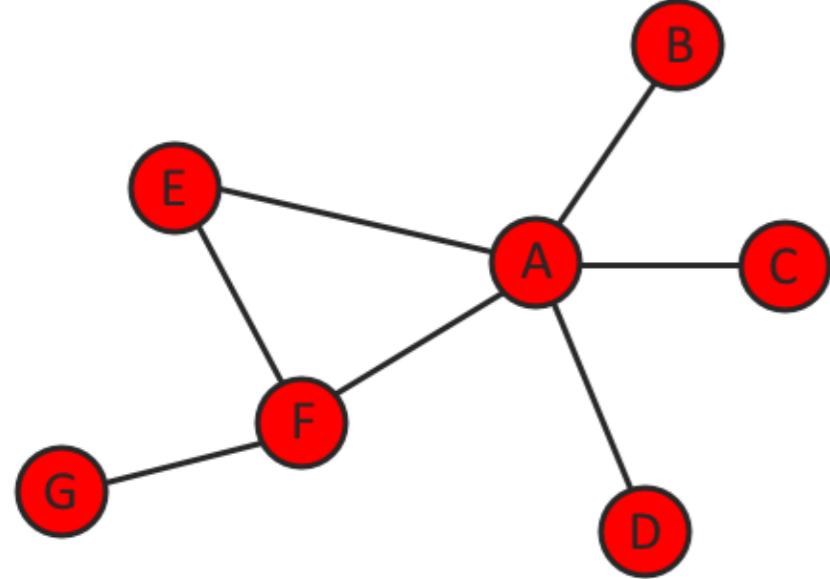
Associate Professor, University of Texas
at Austin

Network data: adjacency matrix



	A	B	C	D	E	F	G
A	0	1	1	1	1	1	0
B	1	0	0	0	0	0	0
C	1	0	0	0	0	0	0
D	1	0	0	0	0	0	0
E	1	0	0	0	0	1	0
F	1	0	0	0	1	0	1
G	0	0	0	0	0	1	0

Network data: edgelist



A	B
A	C
A	D
A	E
A	F
E	F
F	G

The igraph R package

A	B
A	C
A	D
A	E
A	F
E	F
F	G

```
library(igraph)
```

2 columns

```
g <- graph.edgelist(as.matrix(df),  
                     directed = FALSE)
```

```
g
```

```
IGRAPH UN-- 7 7 --  
+ attr: name (v/c)  
+ edges (vertex names):  
[1] A--B A--C A--D A--E A--F E--F F--G
```

7 vertices and 7 edges

Adding Vertex Attributes

```
g <- set_vertex_attr(  
  g,  
  "age",  
  value = c(  
    20, 25, 21, 23, 24, 23, 22  
  ))
```

Adding Edge Attributes

```
g <- set_edge_attr(  
  g,  
  "frequency",  
  value = c(  
    2, 1, 1, 1, 3, 2, 4  
  ))
```

Adding attributes II

edges.df

from	to	frequency
A	B	2
A	C	1
A	D	1
A	E	1
A	F	3
E	F	2
F	G	4

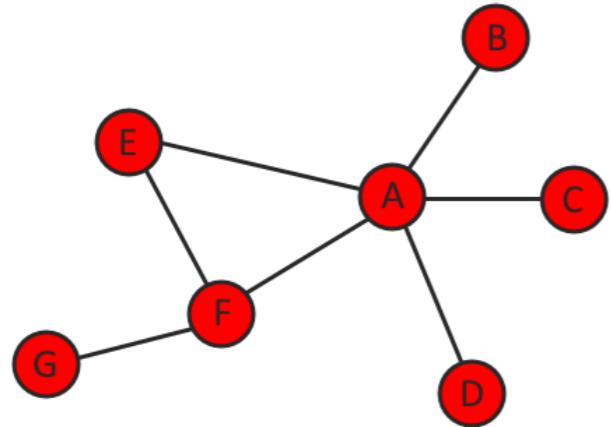
vertices.df

name	age
A	20
B	25
C	21
D	23
E	24
F	23
G	22

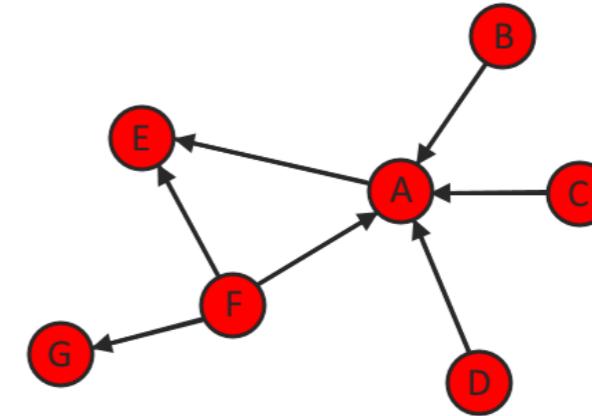
```
graph_from_data_frame(d = edges.df, vertices = vertices.df,  
                      directed = FALSE)
```

Directionality

Undirected



Directed



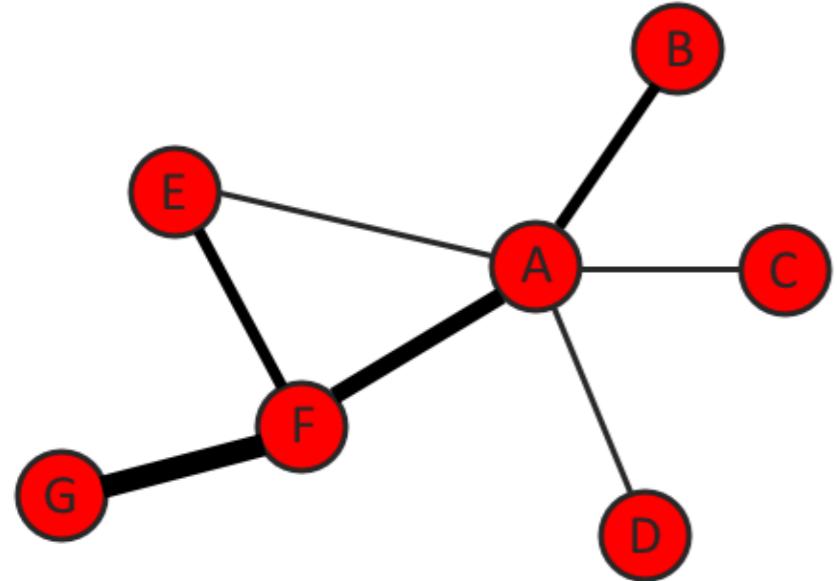
Undirected:

```
IGRAPH UN-- 7 7 --
+ attr: name (v/c)
+ edges (vertex names):
[1] A--B A--C A--D A--E A--F E--F F--G
```

Directed:

```
IGRAPH DN-- 7 7 --
+ attr: name (v/c)
+ edges (vertex names):
[1] A->E B->A C->A D->A F->A F->E F->G
```

Subsetting networks



```
E(g)[[inc('E')]]
```

```
+ 2/7 edges (vertex names):  
tail head tid hid frequency  
4   E   A   5   1       1  
6   F   E   6   5       2
```

```
E(g)[[frequency>=3]]
```

```
+ 2/7 edges (vertex names):  
tail head tid hid frequency  
5   F   A   6   1       3  
7   G   F   7   6       4
```

`[[` filters the object and returns a `data.frame` but we can apply the same filter with a `[` and return `igraph` object.

`edges.df`

from	to	frequency
A	B	2
A	C	1
A	D	1
A	E	1
A	F	3
E	F	2
F	G	4

```
# Create a new igraph object by deleting  
edges that are less than 2 hours long  
g2 <- delete.edges(g1, E(g1)[hours < 2])
```

Getting information from graph object

```
v(g)
```

```
+ 7/7 vertices, named:  
[1] A B C D E F G
```

```
E(g)
```

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

```
vertex_attr(g)
```

```
$name  
[1] "A" "B" "C" "D" "E" "F" "G"  
  
$age  
[1] 20 25 21 23 24 23 22
```

```
gorder(g)
```

```
[1] 7
```

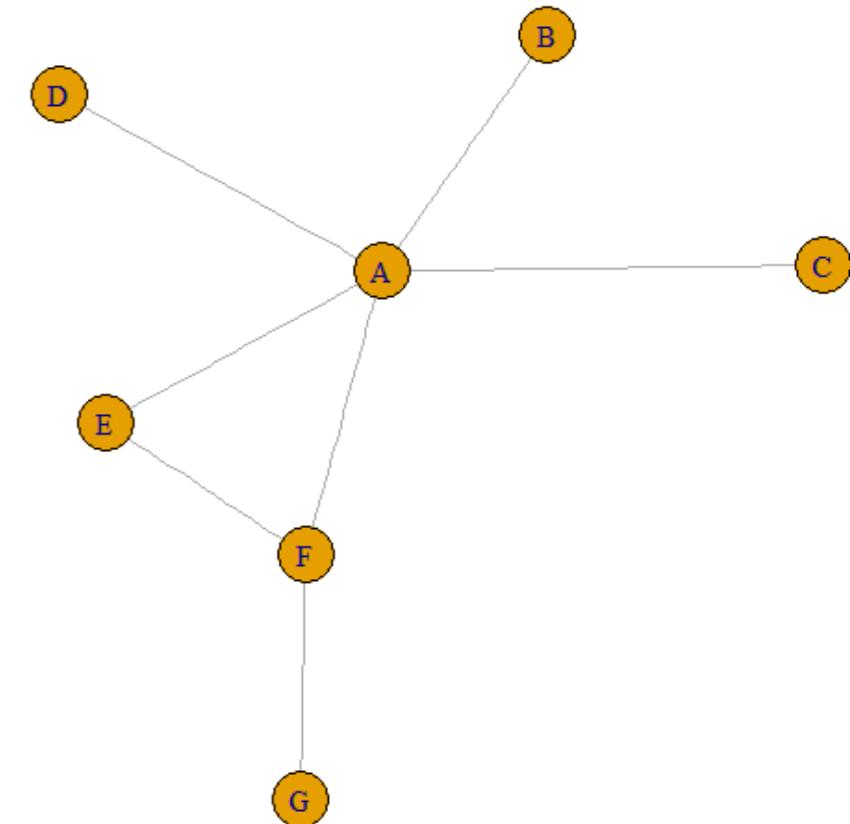
```
gsize(g)
```

```
[1] 7
```

```
edge_attr(g)
```

```
$frequency  
[1] 2 1 1 1 3 2 4
```

```
plot(g)
```



```
is.directed(g)
```

```
[1] TRUE
```

```
is.weighted(g)
```

```
[1] FALSE
```

Is there an edge between A & E?

```
g['A', 'E']
```

```
[1] 1
```

Find the starting vertex of all edges:

```
head_of(g, E(g))
```

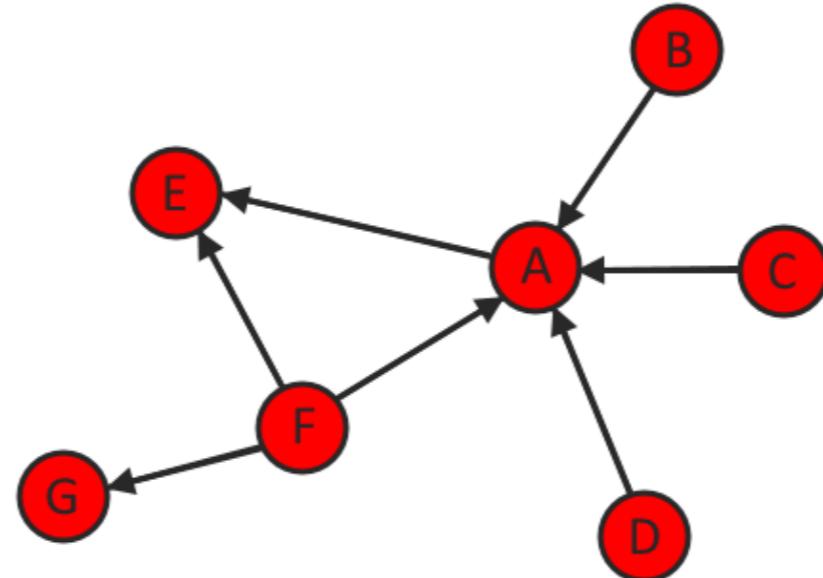
```
+ 7/7 vertices, named:  
[1] A B C D F F F
```

Show all edges to or from A:

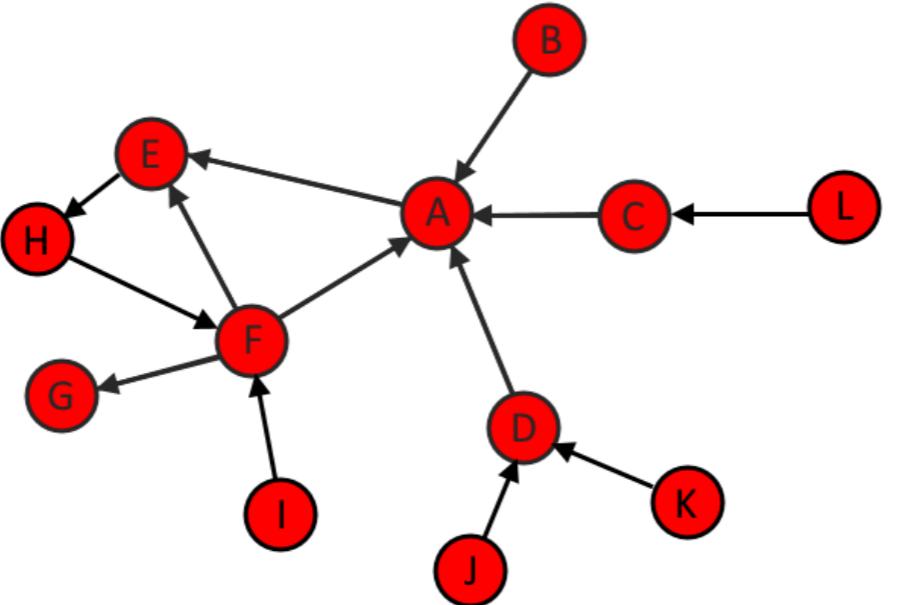
```
incident(g, 'A', mode=c("all"))
```

```
+ 5/7 edges (vertex names):
```

```
[1] A->E B->A C->A D->A F->A
```



Identifying neighbors



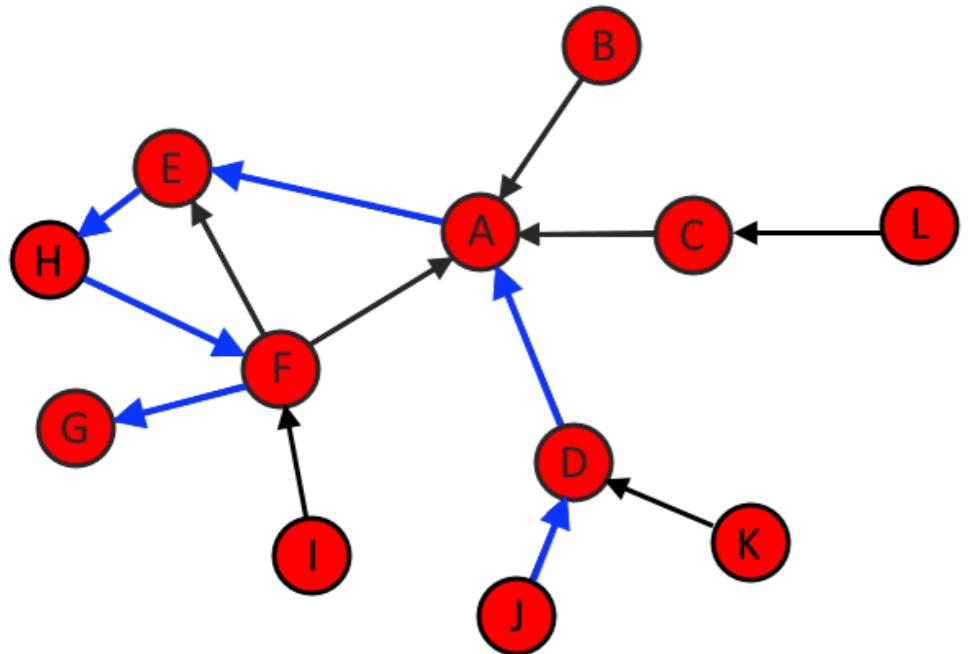
```
neighbors(g, "F", mode = c("all"))
```

```
+ 5/12 vertices, named:  
[1] A E G H I
```

```
x <- neighbors(  
  g, "F", mode = c("all")  
)  
  
y <- neighbors(  
  g, "D", mode = c("all")  
)  
  
intersection(x,y)
```

```
A
```

Paths



```
farthest_vertices(g)
```

```
$vertices  
+ 2/12 vertices, named:
```

```
[1] J G
```

```
$distance
```

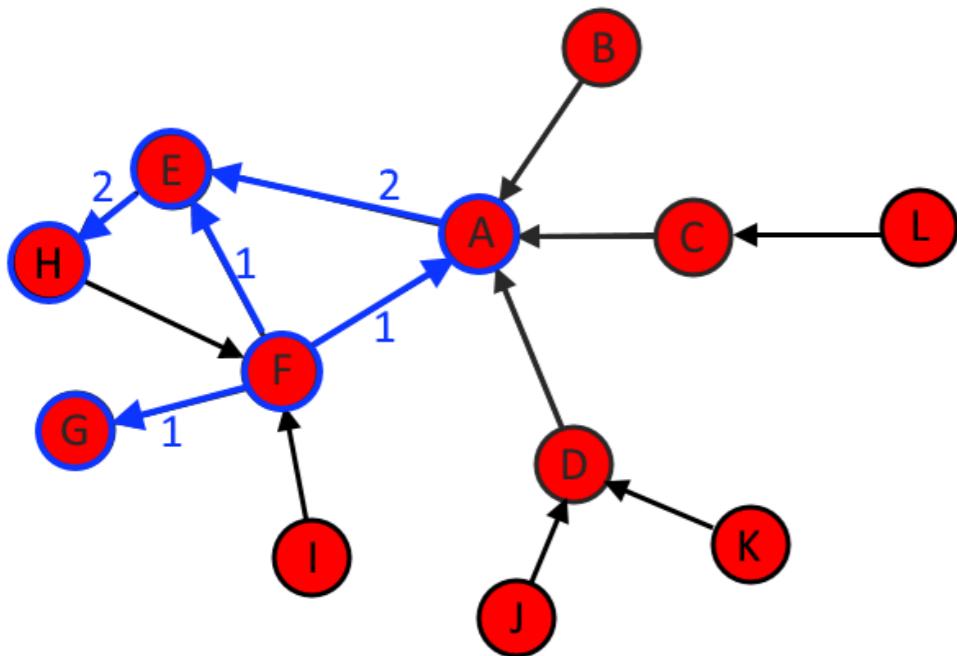
```
[1] 6
```

```
get_diameter(g)
```

```
+ 7/12 vertices, named:
```

```
[1] J D A E H F G
```

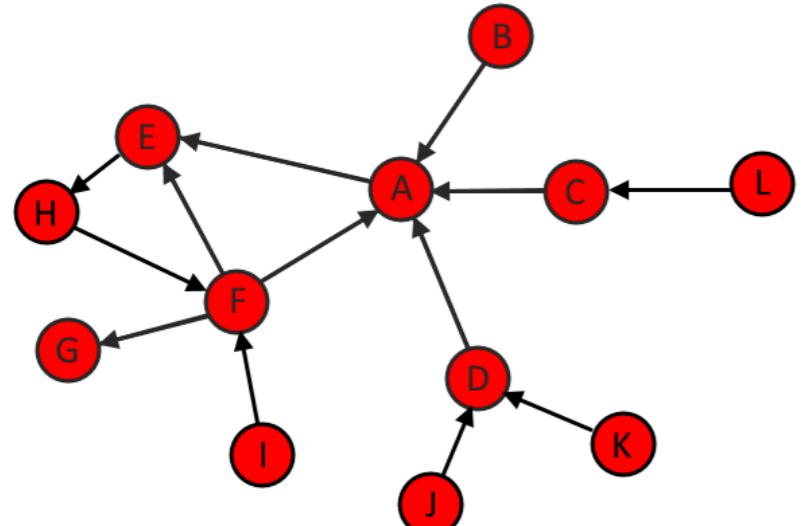
Identifying vertices reachable in N steps



```
ego(g, 2, 'F', mode=c('out'))
```

```
+ 5/12 vertices, named:  
[1] F A E G H
```

Out-degree and in-degree

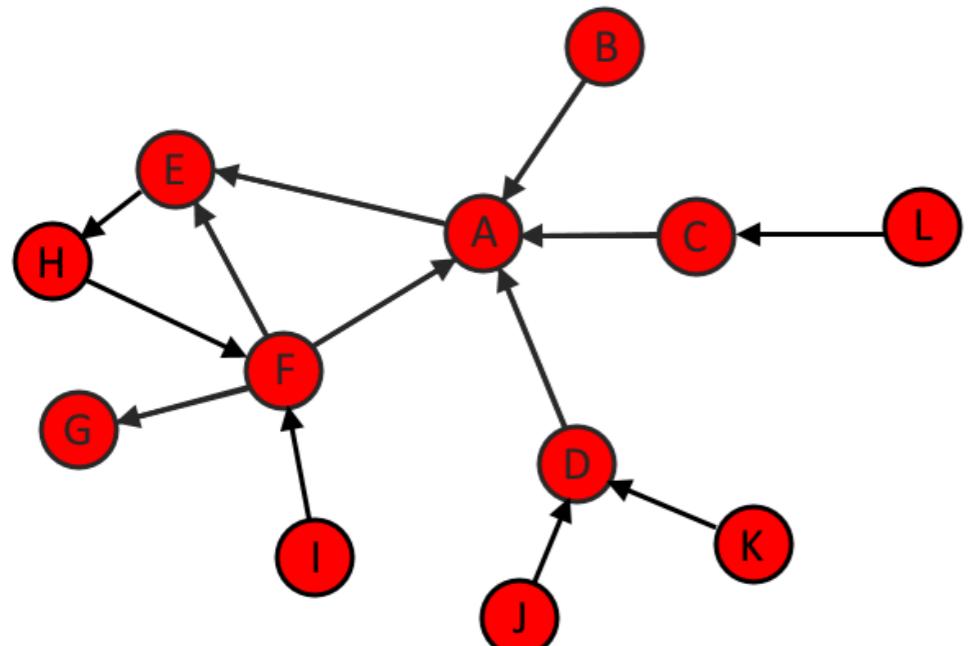


	out-degree	in-degree
A	1	4
B	1	0
C	1	1
D	1	2
E	1	2
F	3	2
G	0	1
H	1	1
I	0	1
J	1	0
K	1	0

```
degree(g, mode = c("out"))
```

```
A B C D E F G H I J K L  
1 1 1 1 1 3 0 1 1 1 1 1
```

Betweenness



I to H:

I → F → E → H
I → F → A → E → H

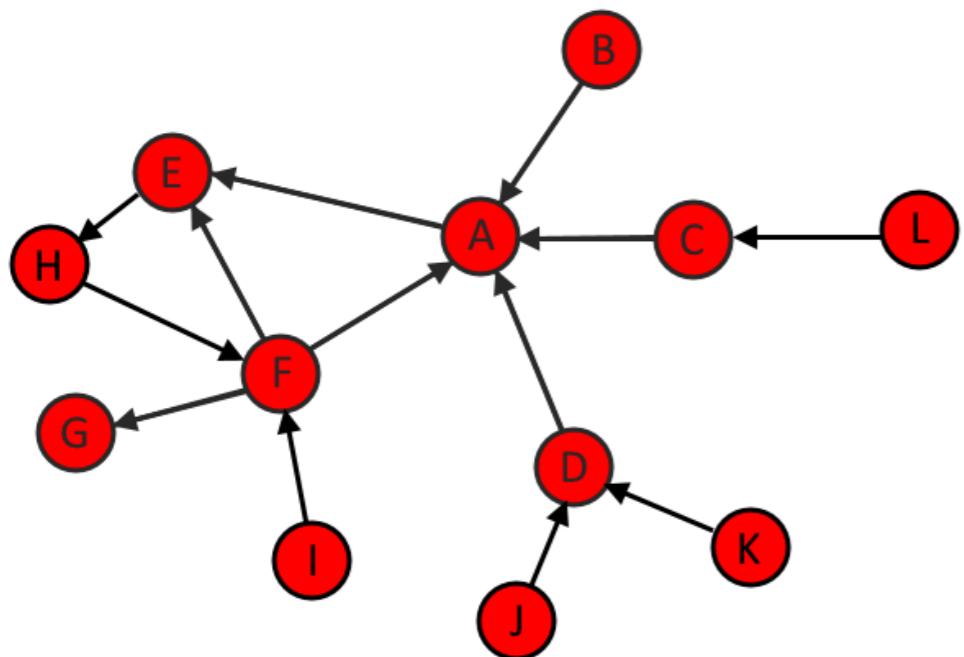
K to E:

K → D → A → E

B to G:

B → A → E → H → F → G

Betweenness



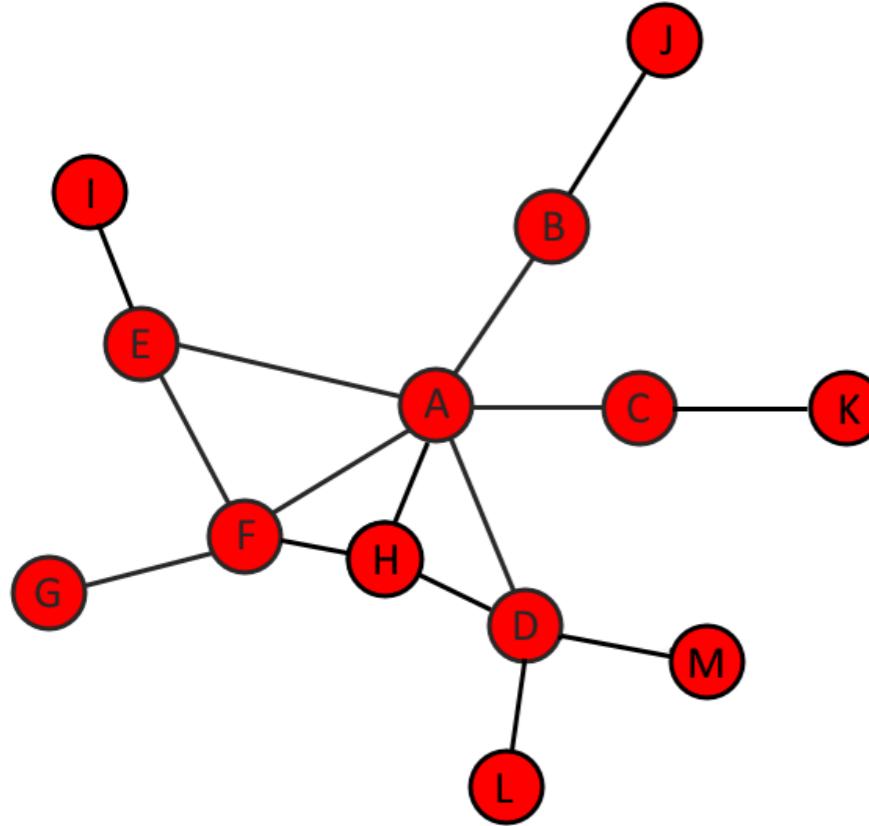
```
betweenness(g, directed = TRUE)
```

A	B	C	D	E	F	G	H	I	J	K	L
24	0	5	10	23	16	0	17	0	0	0	0

```
betweenness(g, directed = TRUE,  
normalized = TRUE)
```

A	B	C	D	E	F
0.22	0.00	0.05	0.09	0.21	0.15
G	H	I	J	K	L
0.00	0.15	0.00	0.00	0.00	0.00

Eigenvector centrality



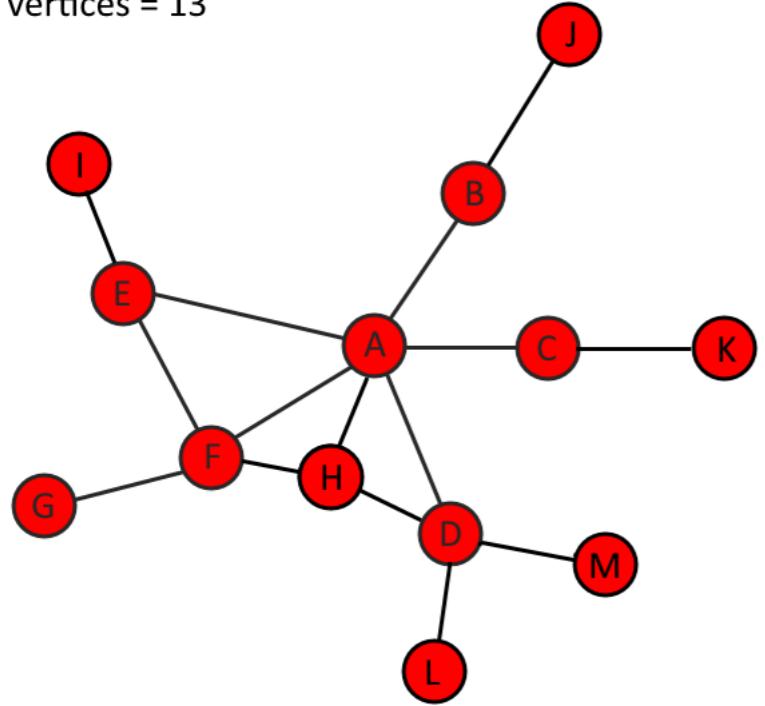
```
eigen_centrality(g)$vector
```

A	B	C	D	E	F	G
1.00	0.33	0.33	0.63	0.58	0.76	0.23
H	I	J	K	L	M	
0.71	0.17	0.10	0.10	0.19	0.19	

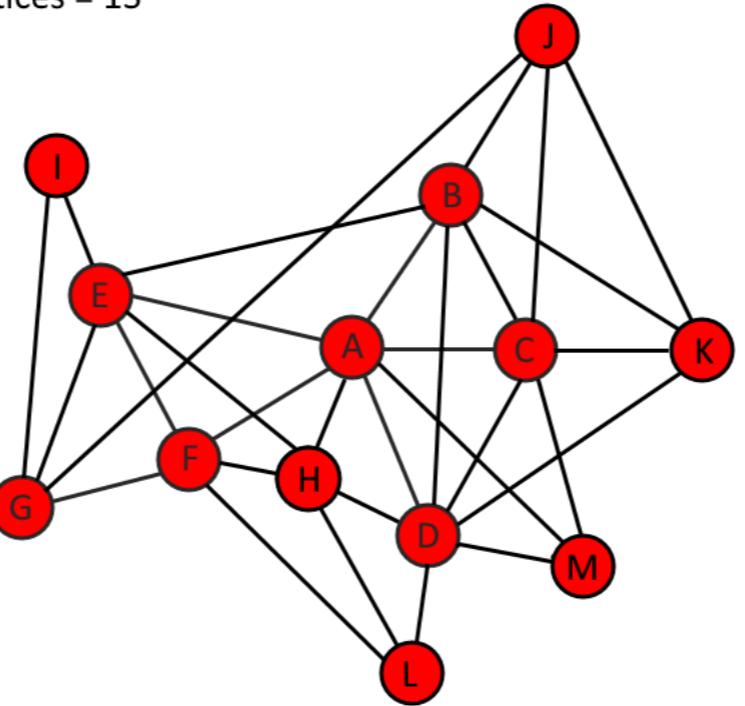
Eigenvector centrality is a measure of how well connected a vertex is. Vertices with the highest eigenvector centrality are those that are connected to many others but especially to other vertices who themselves are highly connected to others.

Density

density = 0.19
edges = 15
vertices = 13



density = 0.38
edges = 30
vertices = 13

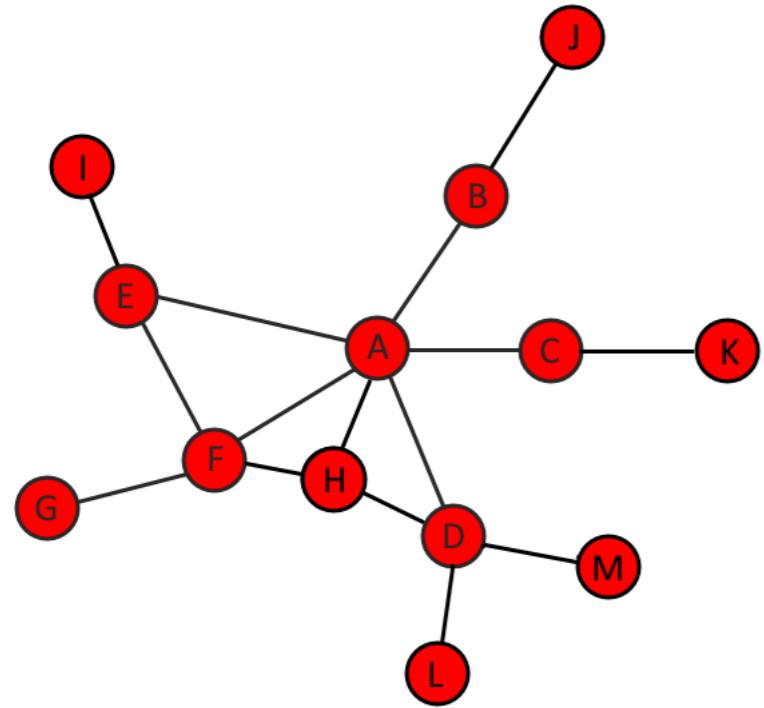


Is equivalent to the proportion of edges that actually do exist in a network out of all those that potentially could exist between every pair of vertices.

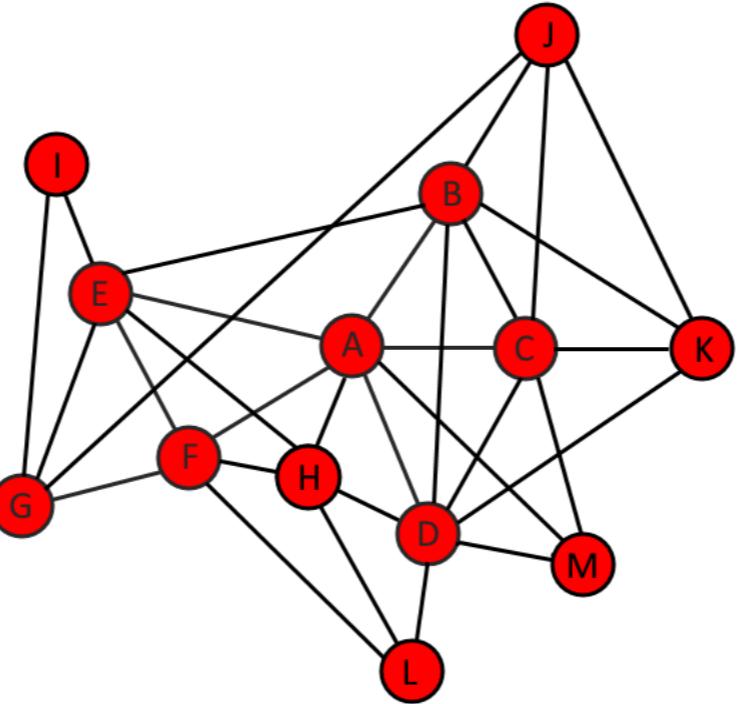
`edge_density(g)`

Average path length

average path length = 2.47



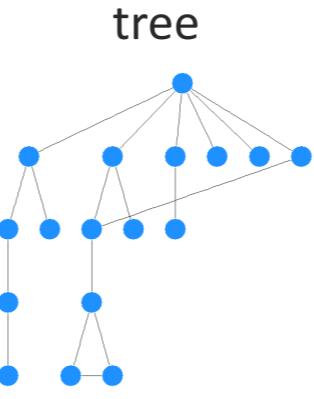
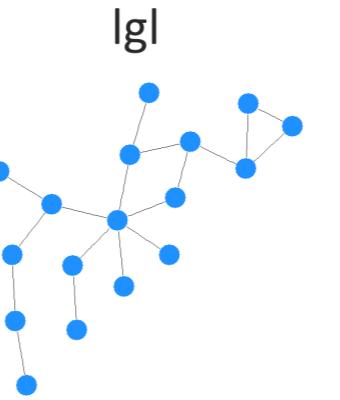
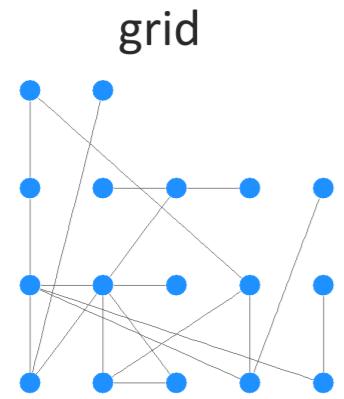
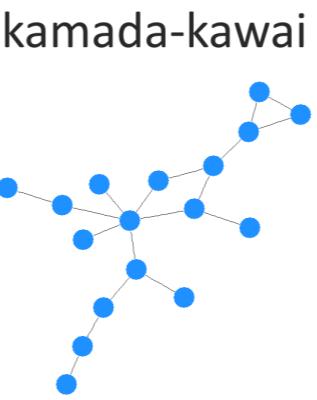
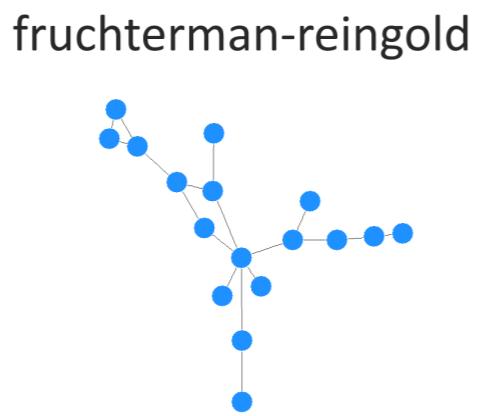
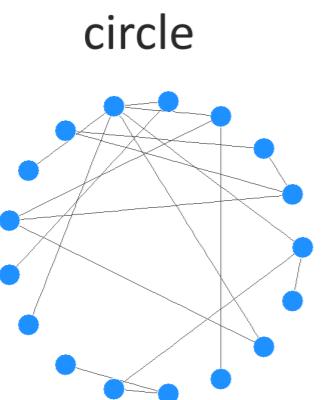
average path length = 1.81



the mean of the lengths of the shortest paths between all pairs of vertices in the network.

```
mean_distance(g, directed = FALSE)
```

igraph layouts

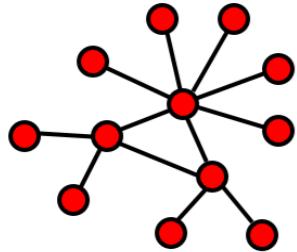


```
plot(g, layout = layout.fruchterman.reingold(g))
```

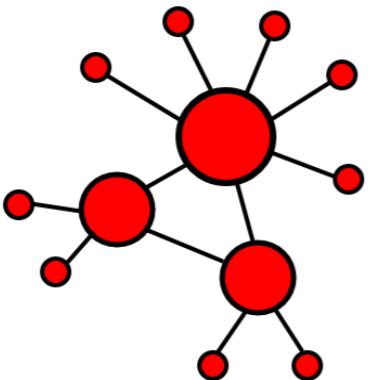
layout_as_bipartite
layout_as_star
layout_as_tree
layout_components
layout_in_circle
layout_nicely
layout_on_grid
layout_on_sphere
layout_randomly
layout_with_dh
layout_with_drl
layout_with_fr
layout_with_gem
layout_with_graphopt
layout_with_kk
layout_with_lgl
layout_with_mds
layout_with_sugiyama

Styling vertices and edges

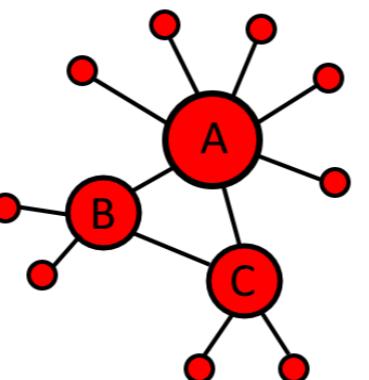
default



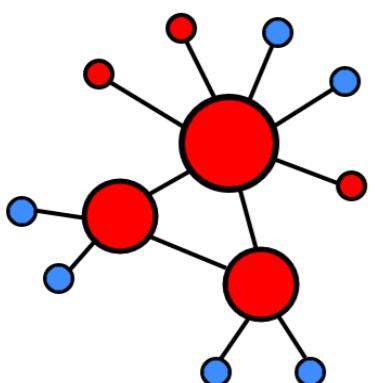
size



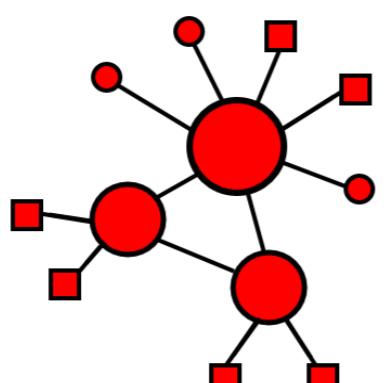
labels



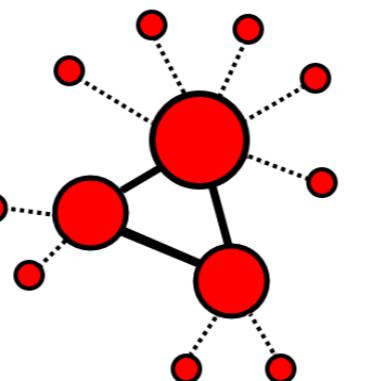
color



shape



edges



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

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 NA

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

Source: <https://kateto.net/netscix2016.html>

Let's practice!

NETWORK ANALYSIS IN R

Network Randomizations

NETWORK ANALYSIS IN R

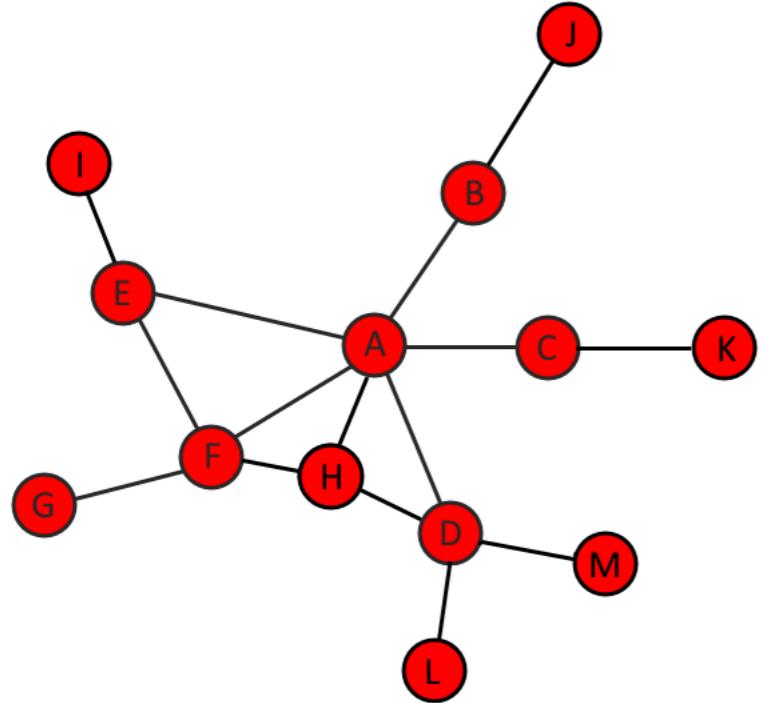


James Curley

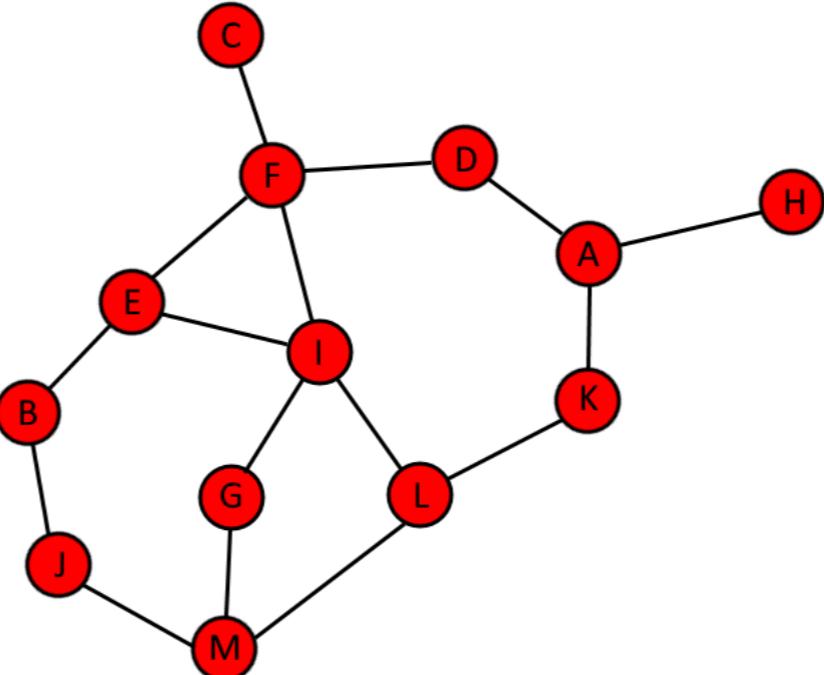
Associate Professor, University of Texas
at Austin

Random graphs

density = 0.19
vertices = 13



density = 0.21
vertices = 13



```
erdos.renyi.game(n = gorder(g), p.or.m = edge_density(g), type = "gnp")
```

Random graphs & randomization tests

1. Generate 1000 random graphs based on the original network
 - e.g. with the same number of vertices and approximate density.
2. Calculate the average path length of the original network.
3. Calculate the average path length of the 1000 random networks.
4. Determine how many random networks have an average path length greater or less than the original network's average path length.

Generate 1000 random graphs:

```
gl <- vector('list',1000)

for(i in 1:1000){
  gl[[i]] <- erdos.renyi.game(
    n = gorder(g),
    p.or.m = edge_density(g),
    type = "gnp"
  )
}
```

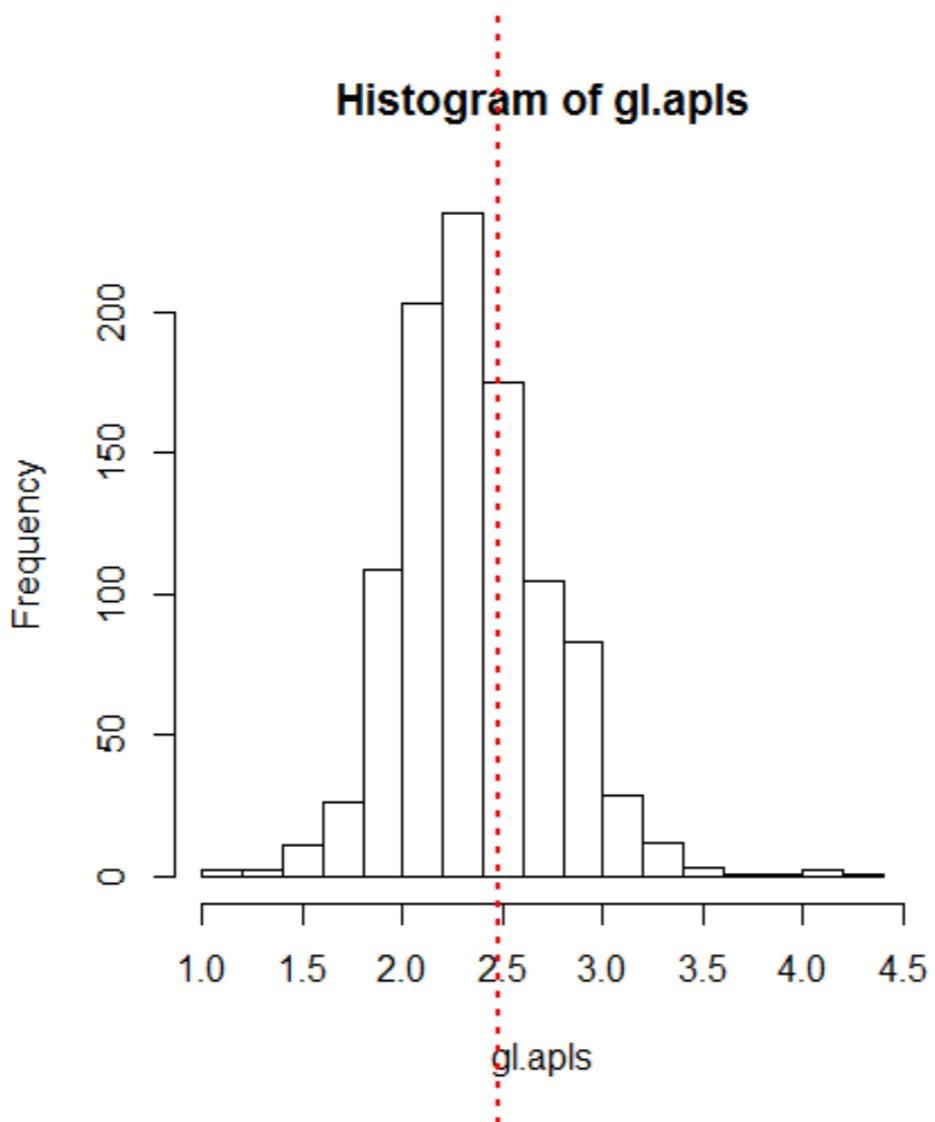
Calculate average path length of 1000 random graphs:

```
gl.apls <- unlist(
  lapply(gl, mean_distance, directed = FALSE)
)
```

Comparing to the original network

```
hist(gl.apls, breaks = 20)

abline(
  v = mean_distance(
    g, directed=FALSE
  ),
  col = "red",
  lty = 3,
  lwd = 2
)
```



Let's practice!

NETWORK ANALYSIS IN R

Network substructures

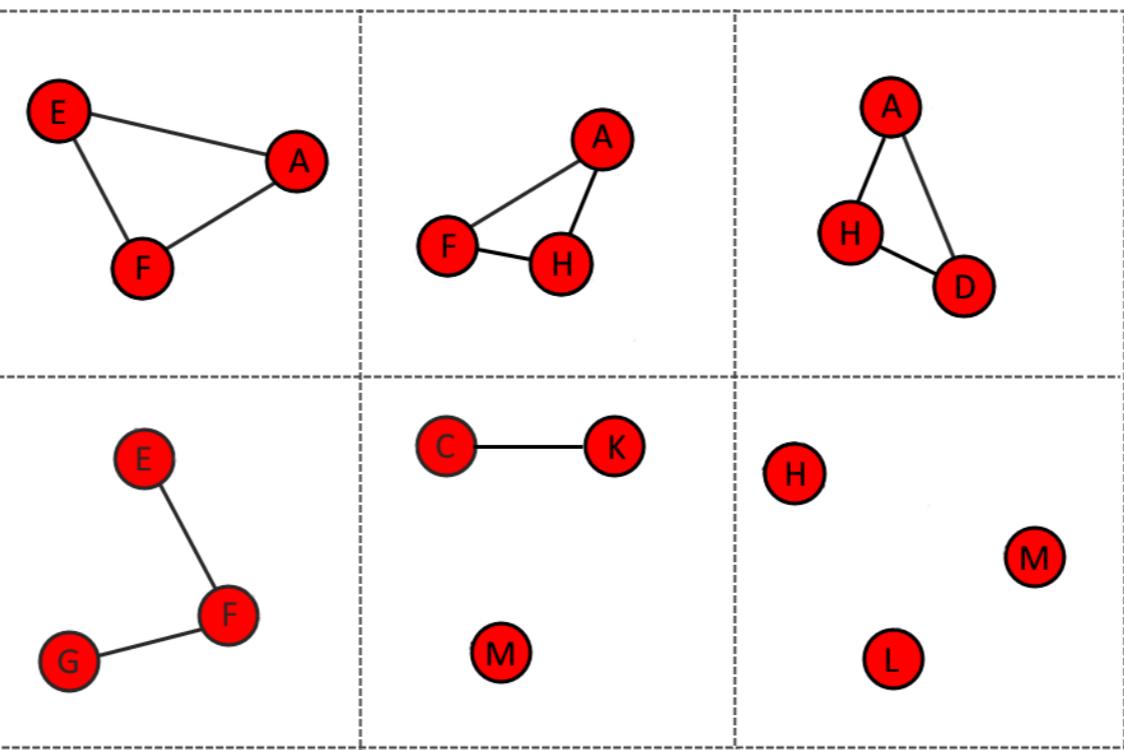
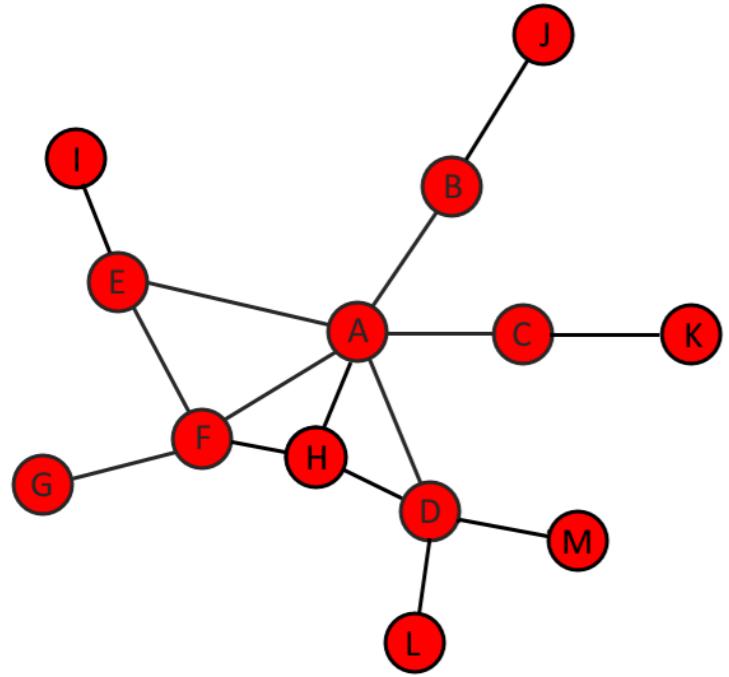
NETWORK ANALYSIS IN R



James Curley

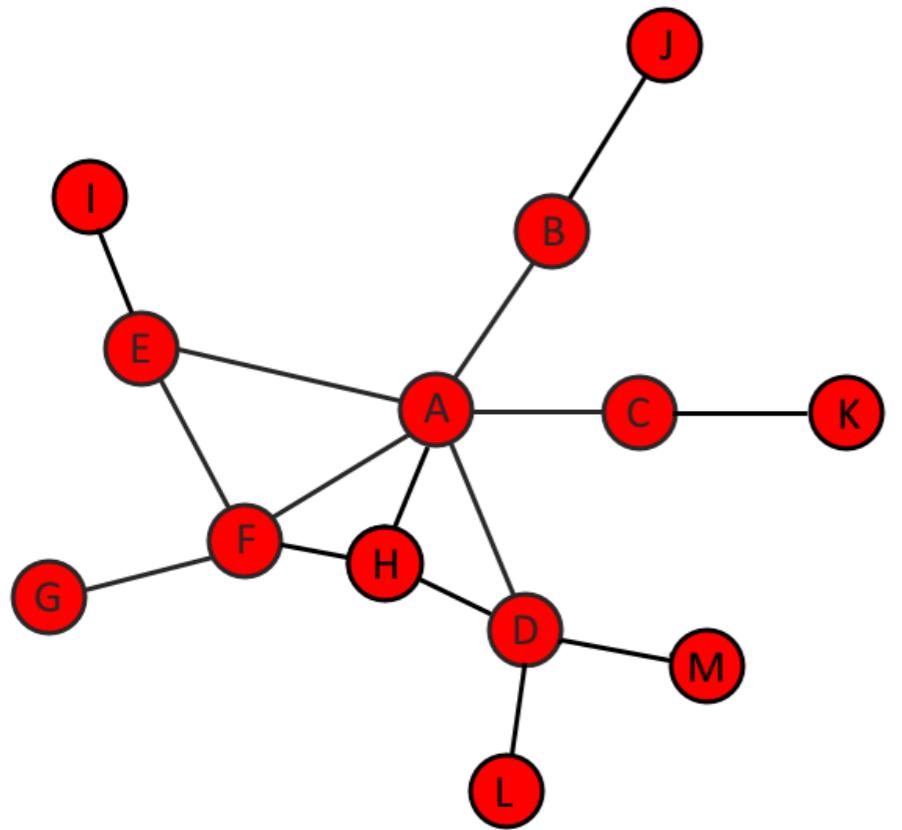
Associate Professor, University of Texas
at Austin

Transitivity



`triangles(g)`

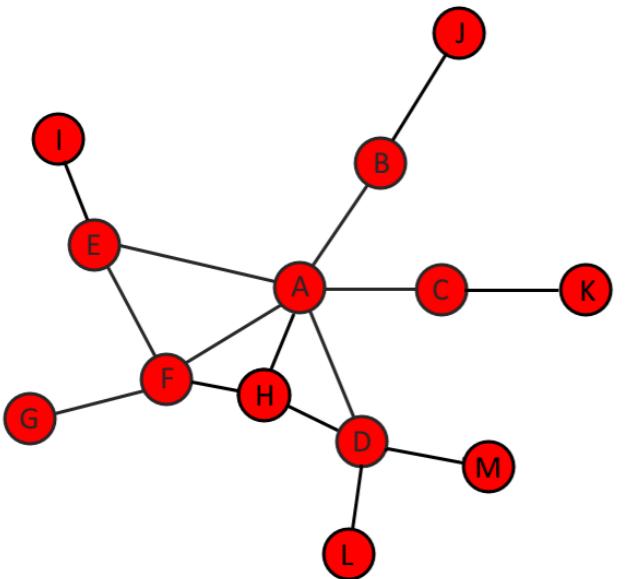
Global transitivity



`transitivity(g)`

[1] 0.26

Local transitivity



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

0.2

```
count_triangles(g, vids = 'A')
```

3

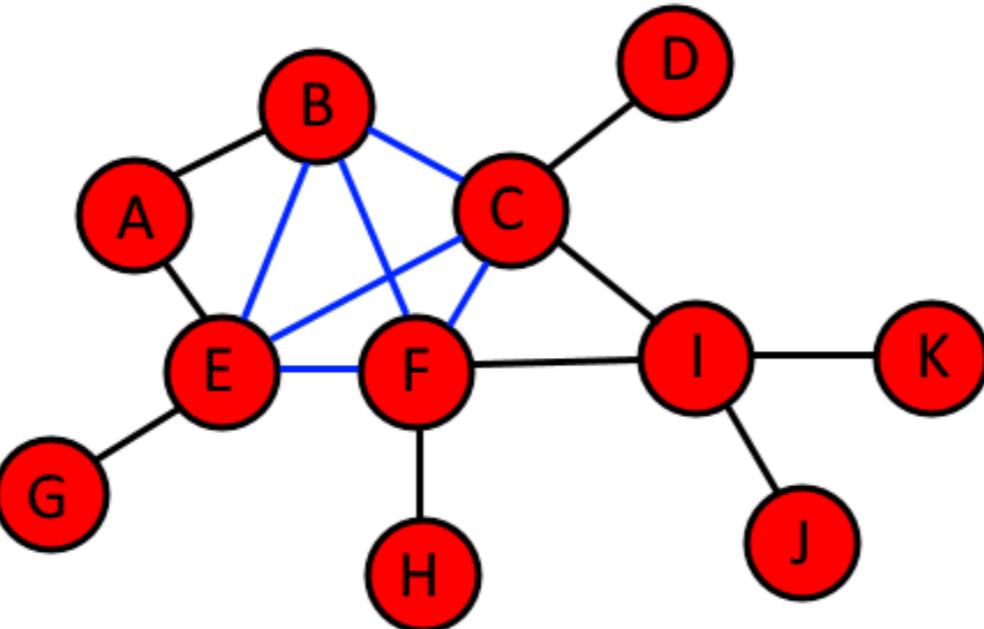
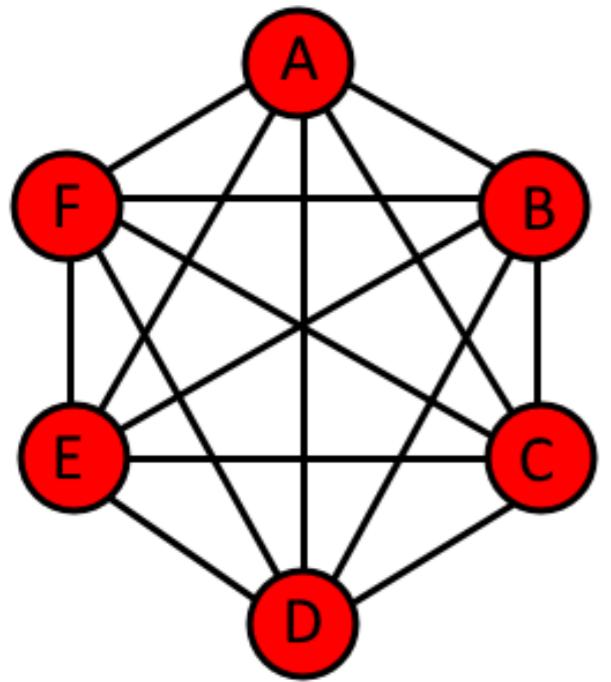
```
transitivity(g,  
            vids = 'F',  
            type = "local")
```

0.33

```
count_triangles(g, vids = 'F')
```

2

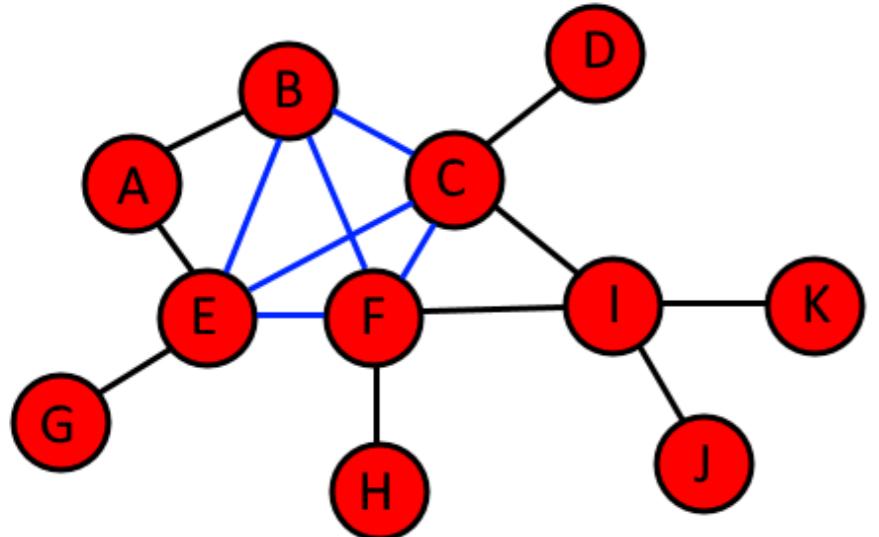
Cliques



Identifying cliques

```
largest_cliques(g)
```

```
+ 4/11 vertices, named:  
[1] C F B E
```



```
max_cliques(g)
```

```
...  
[[6]]  
+ 3/11 vertices, named:  
[1] A B E  
  
[[7]]  
+ 3/11 vertices, named:  
[1] I C F  
  
[[8]]  
+ 4/11 vertices, named:  
[1] E B F C
```

Let's practice!

NETWORK ANALYSIS IN R

Close relationships: assortativity & reciprocity

NETWORK ANALYSIS IN R

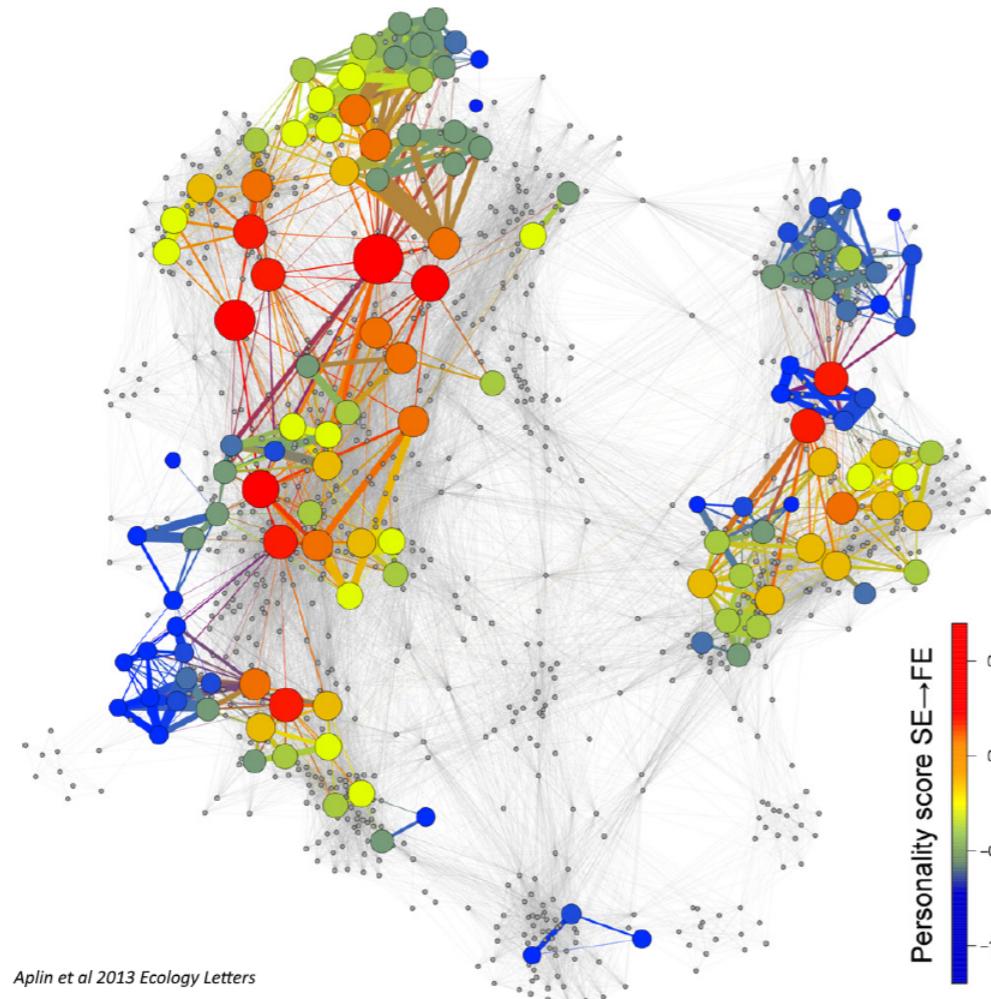
James Curley

Associate Professor, University of Texas
at Austin

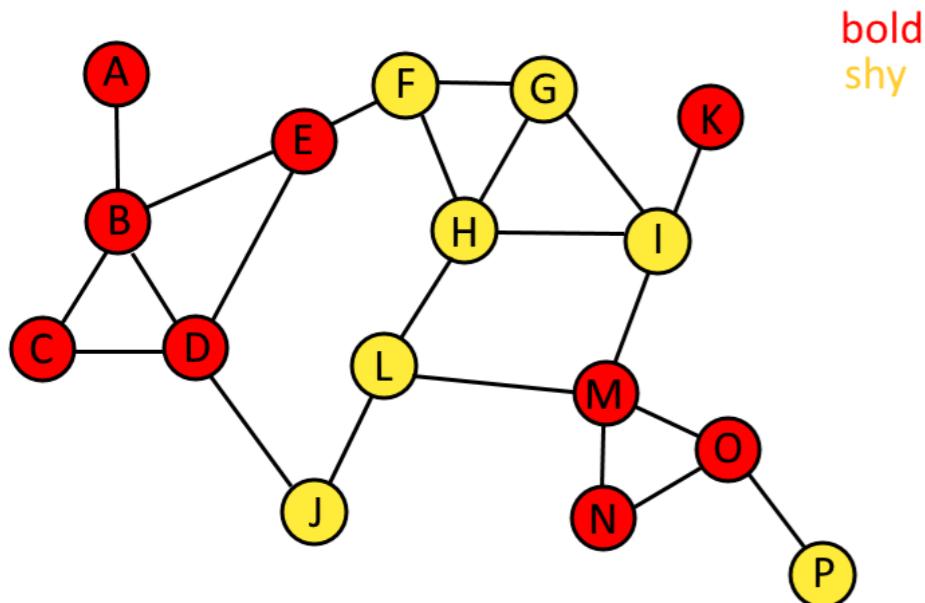


Assortativity

The preferential attachment of vertices to other vertices that are similar in numerical or categorical attributes.



Assortativity



```
assortativity(g, values)
```

```
0.45
```

```
assortativity.degree(
```

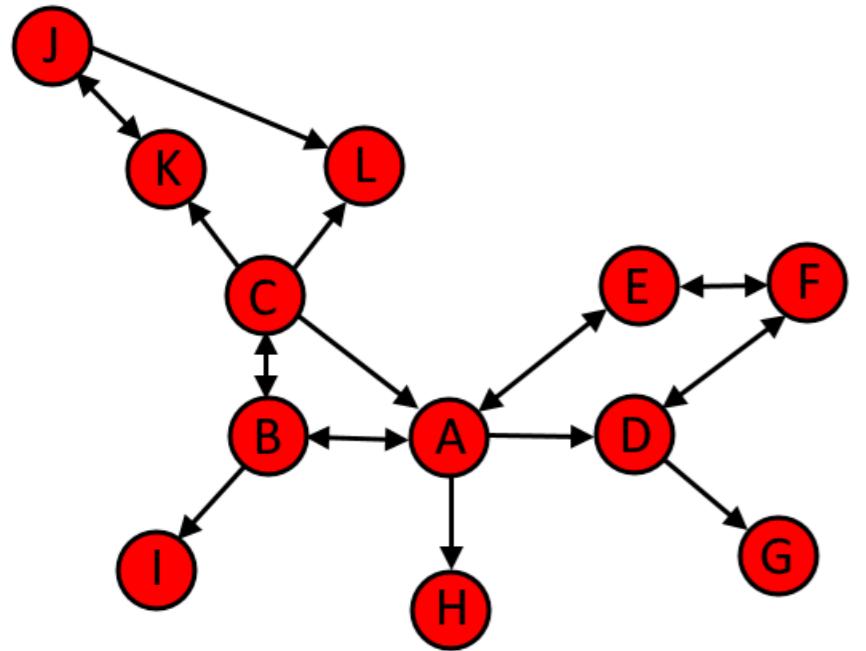
```
g,
```

```
directed = FALSE
```

```
)
```

```
-0.31
```

Reciprocity



`reciprocity(g)`

0.6

Let's practice!

NETWORK ANALYSIS IN R

Community detection

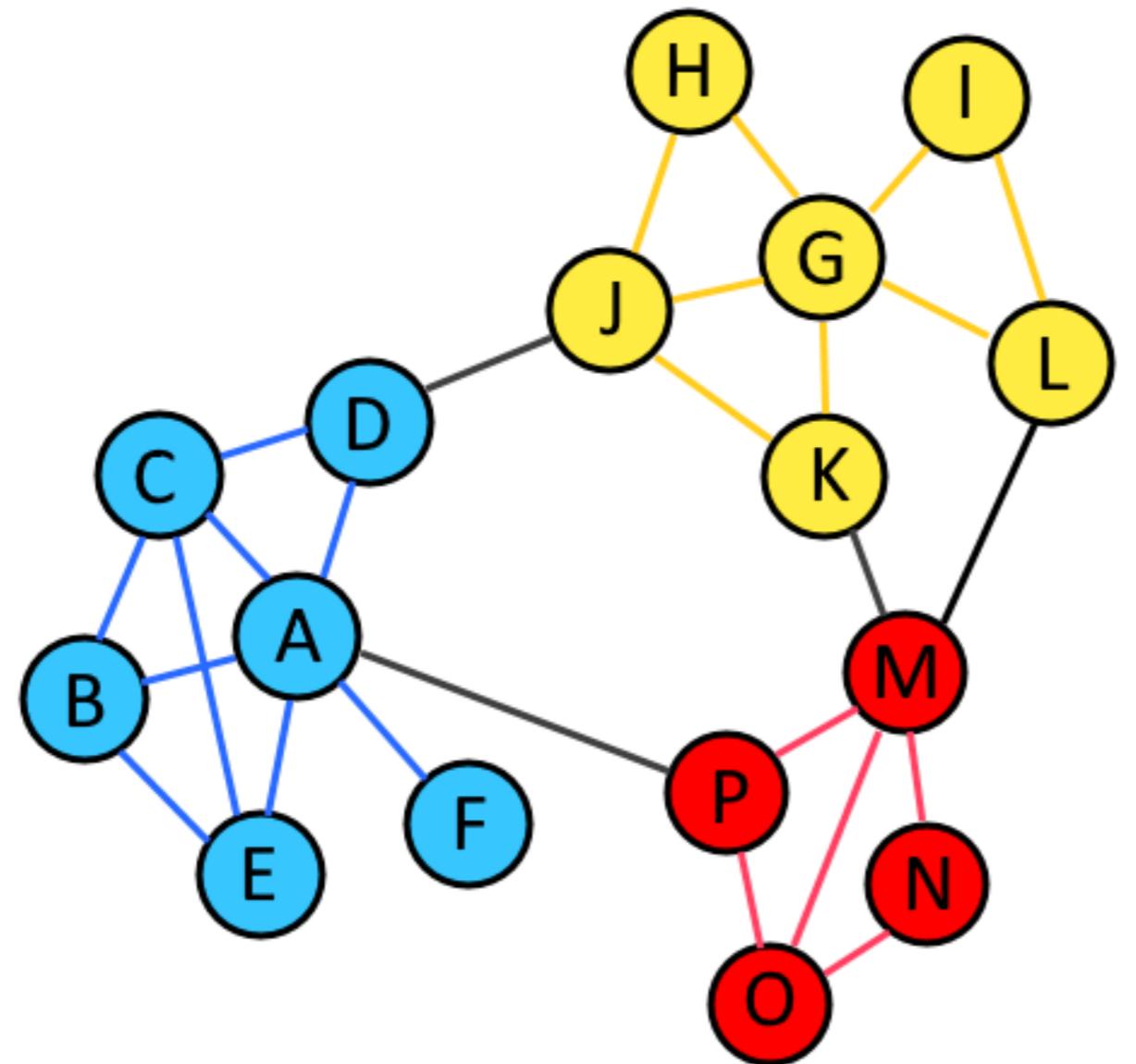
NETWORK ANALYSIS IN R



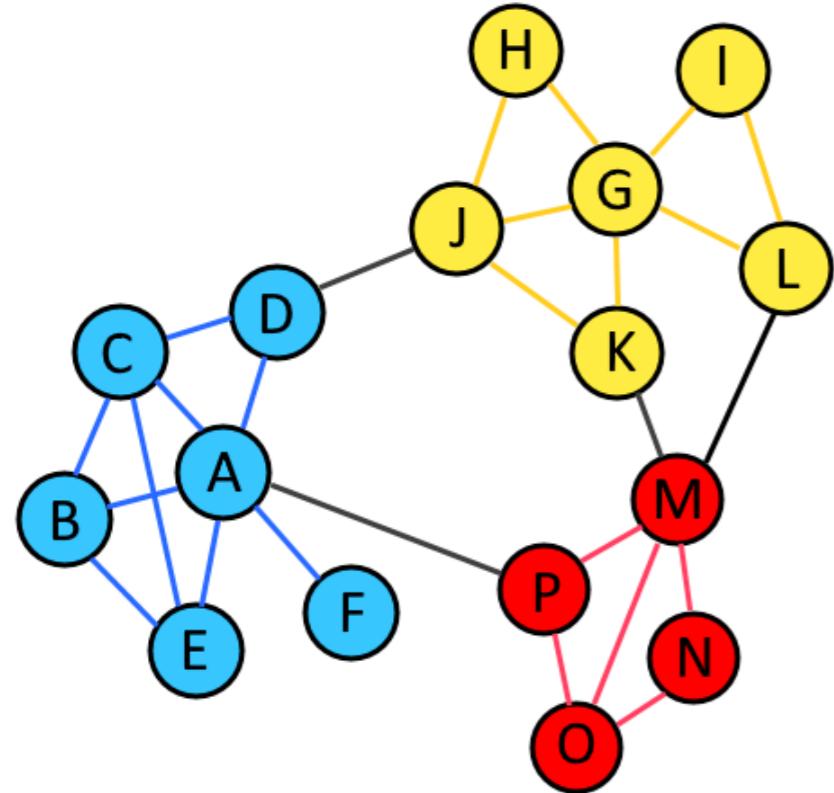
James Curley

Associate Professor, University of Texas
at Austin

Community detection in networks



Fast-greedy detection



```
fastgreedy.community(g)
```

```
IGRAPH clustering fast greedy,  
groups: 3, mod: 0.5
```

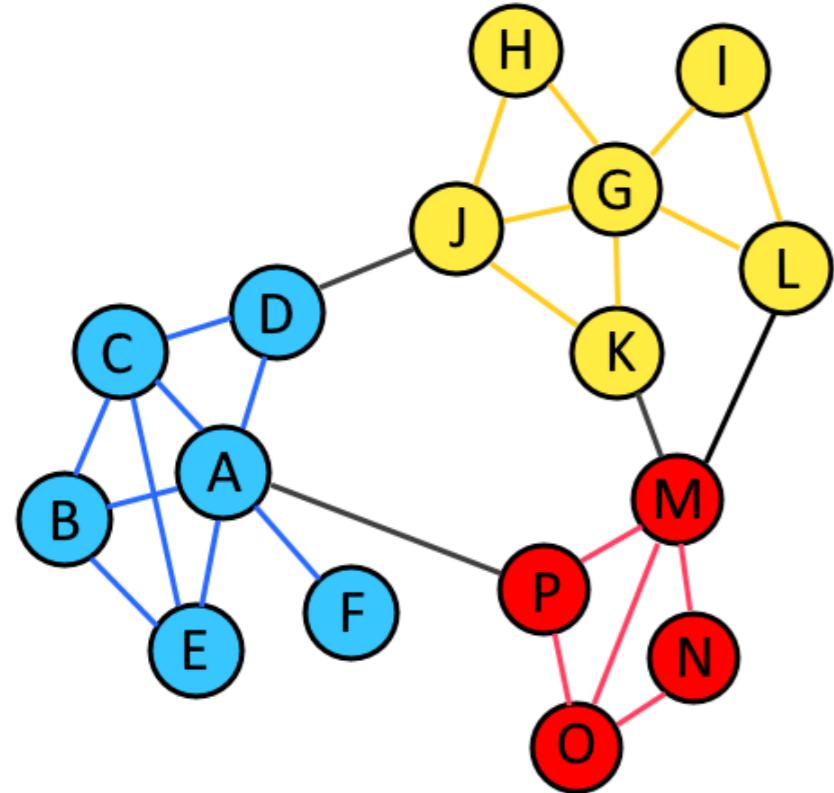
```
+ groups:
```

```
$`1`  
[1] "A" "B" "C" "D" "E" "F"
```

```
$`2`  
[1] "J" "G" "H" "I" "K" "L"
```

```
$`3`  
[1] "M" "N" "O" "P"
```

Edge-betweenness detection



```
edge.betweenness.community(g)
```

```
IGRAPH clustering edge betweenness,  
groups: 3, mod: 0.5
```

```
+ groups:
```

```
$`1`  
[1] "A" "B" "C" "D" "E" "F"
```

```
$`2`  
[1] "J" "G" "H" "I" "K" "L"
```

```
$`3`  
[1] "M" "N" "O" "P"
```

```
x <- fastgreedy.community(g)  
length(x)
```

```
[1] 3
```

```
sizes(x)
```

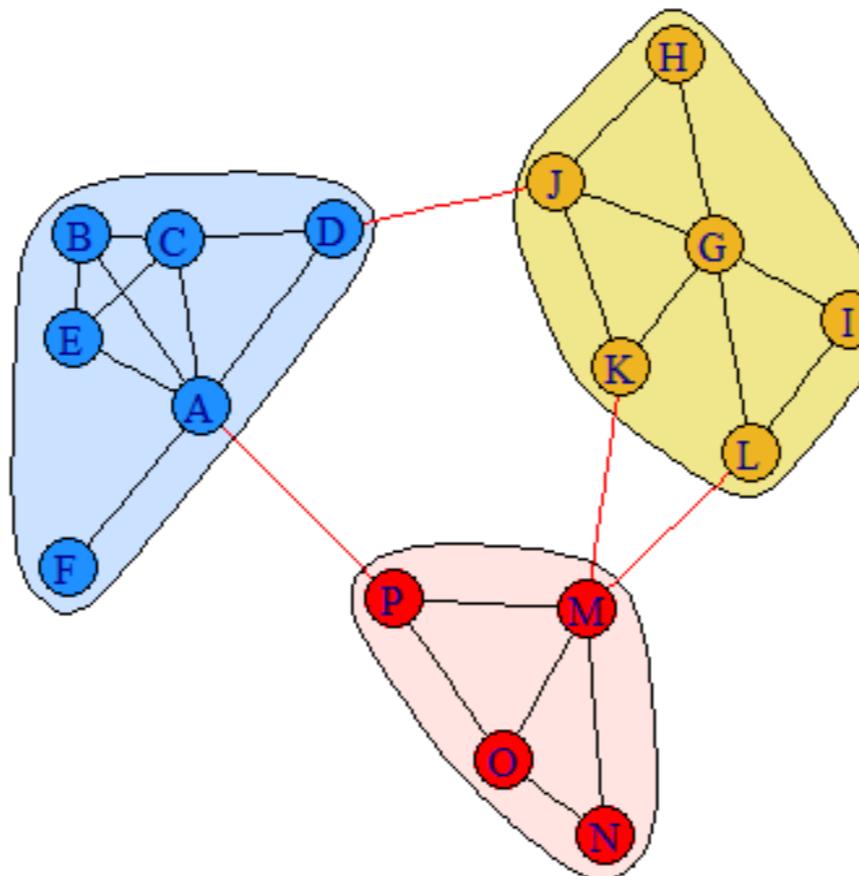
```
Community sizes
```

```
1 2 3  
6 6 4
```

```
membership(x)
```

```
A B C D E F J G H I K L M N O P  
1 1 1 1 1 1 2 2 2 2 2 3 3 3 3 3
```

```
plot(x, g)
```



Let's practice!

NETWORK ANALYSIS IN R

Interactive network visualizations

NETWORK ANALYSIS IN R



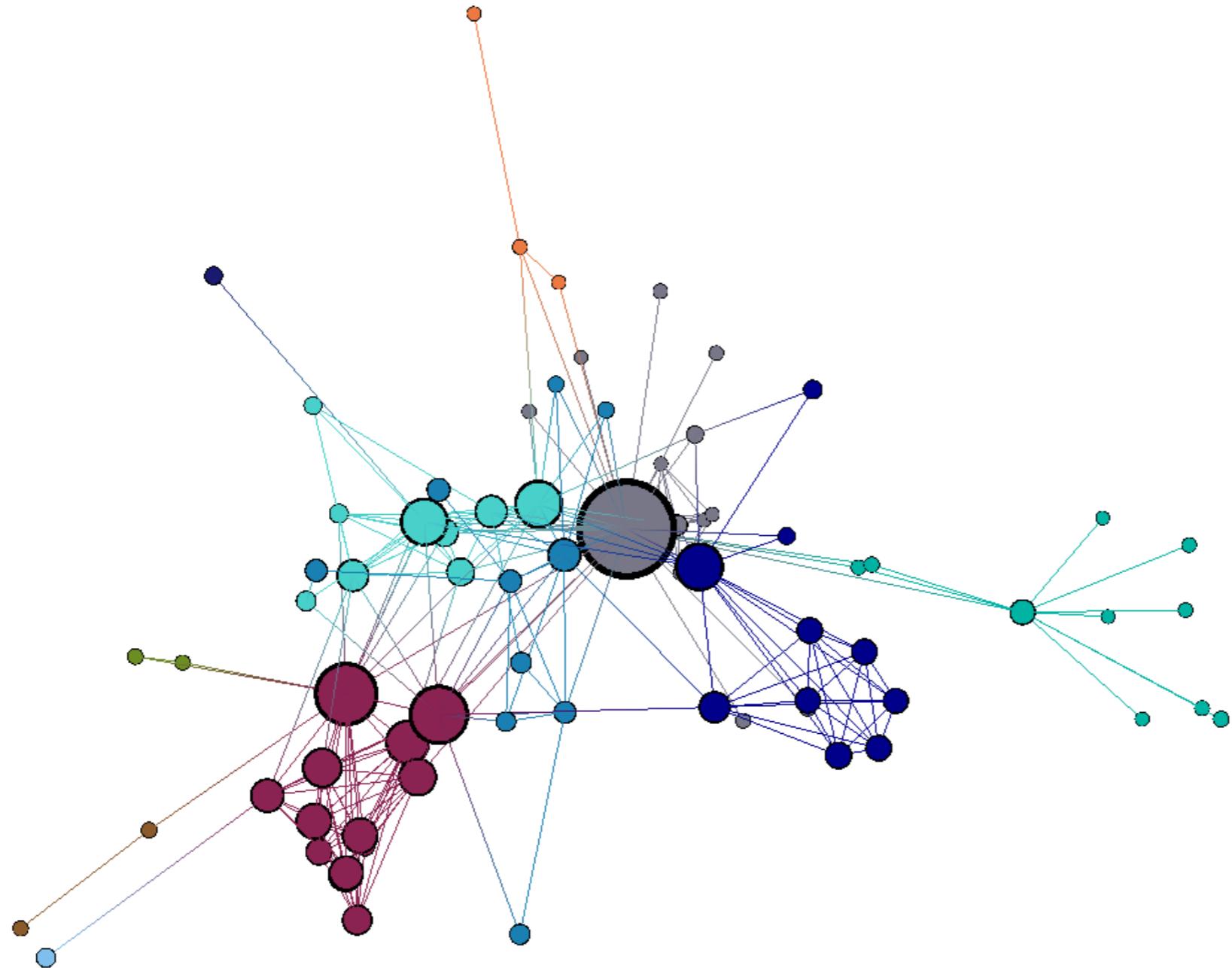
James Curley

Associate Professor, University of Texas
at Austin

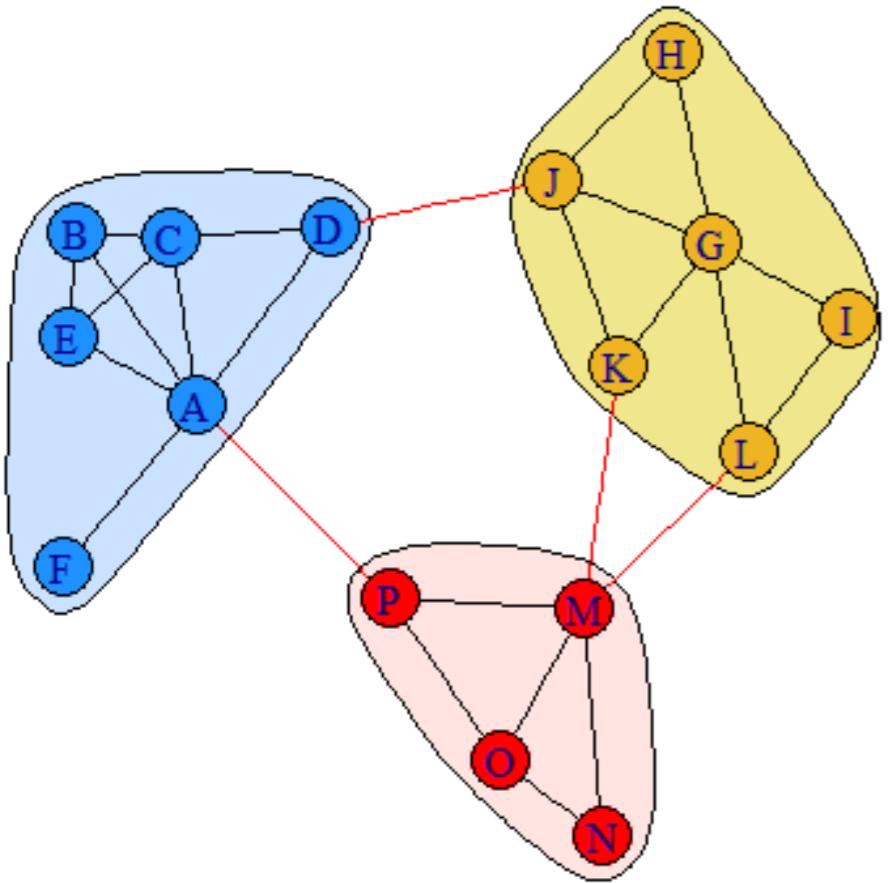
R network visualization packages

- igraph
- statnet
- ggnet
- ggnetwork
- ggraph
- visNetwork
- networkD3
- sigma
- rgexf (igraph to Gephi)
- **threejs**

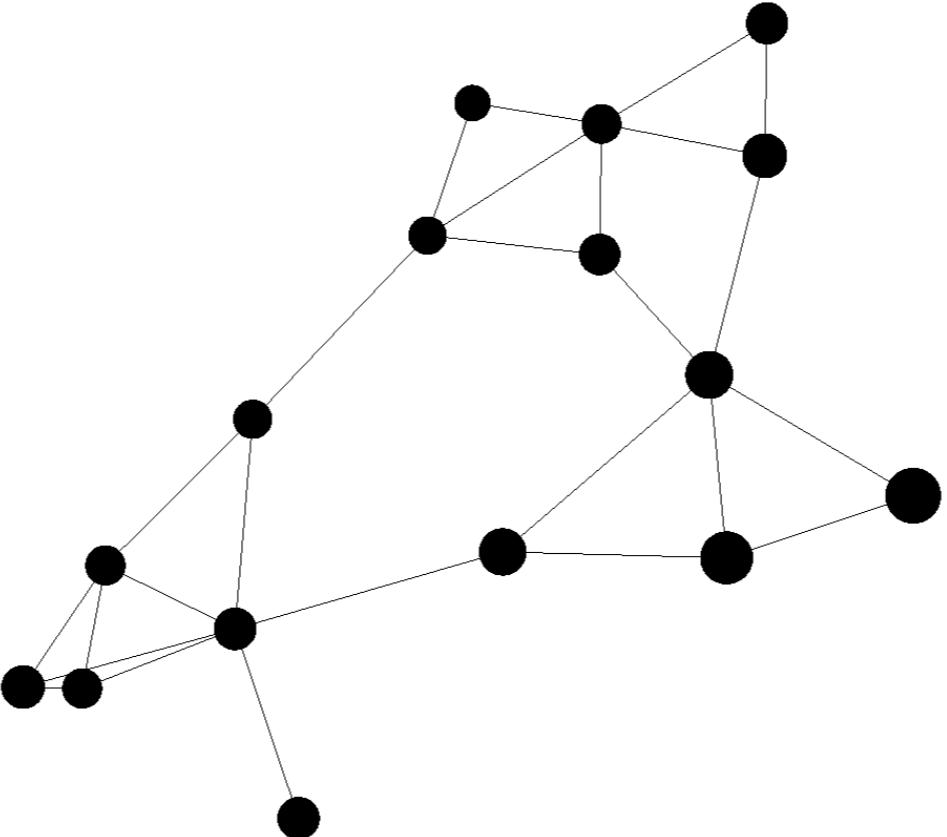
threejs



Creating a threejs visualization

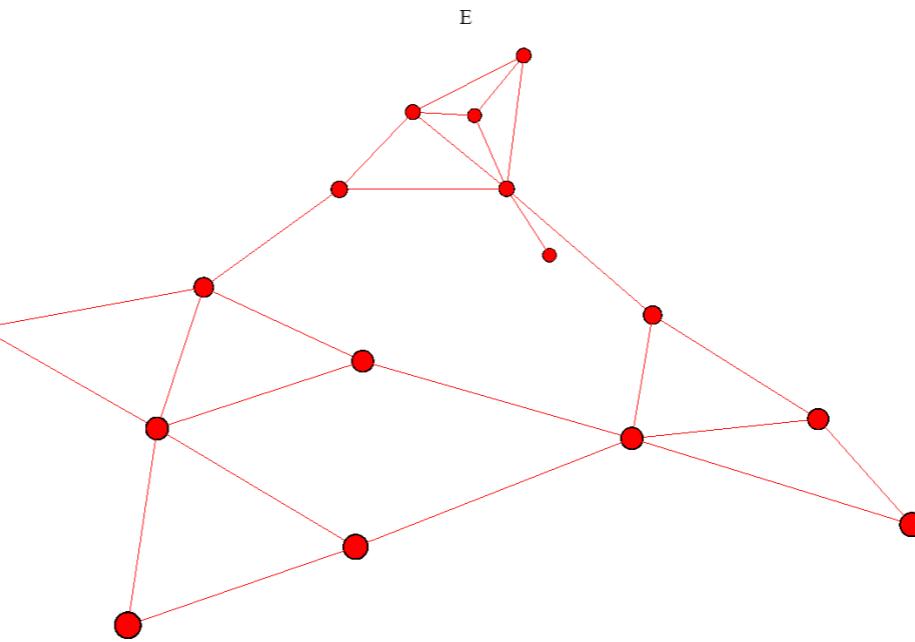


```
library(threejs)  
graphjs(g)
```



Adding attributes

```
g <- set_vertex_attr(  
  g,  
  "label",  
  value = V(g)$name  
)  
  
g <- set_vertex_attr(  
  g,  
  "color",  
  value = "mistyrose"  
)  
  
graphjs(g, vertex.size = 1)
```

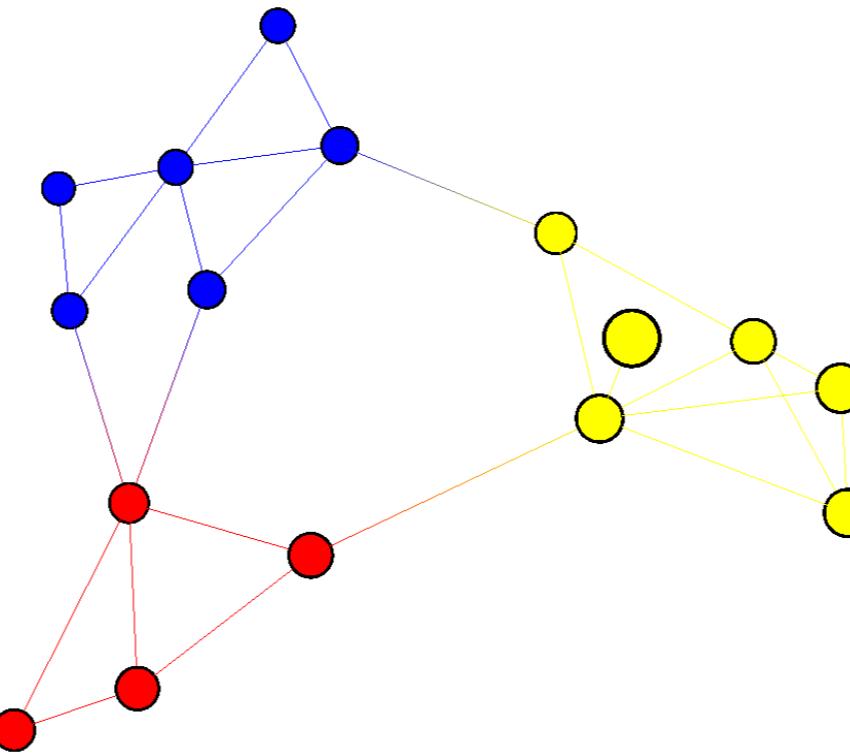


Coloring communities

```
x = edge.betweenness.community(g)
i <- membership(x)

g <- set_vertex_attr(
  g,
  "color",
  value = c(
    "yellow", "blue", "red"
  )[i]
)

graphjs(g)
```



Let's practice!

NETWORK ANALYSIS IN R

Motivation: social networks and predictive analytics

PREDICTIVE ANALYTICS USING NETWORKED DATA IN R

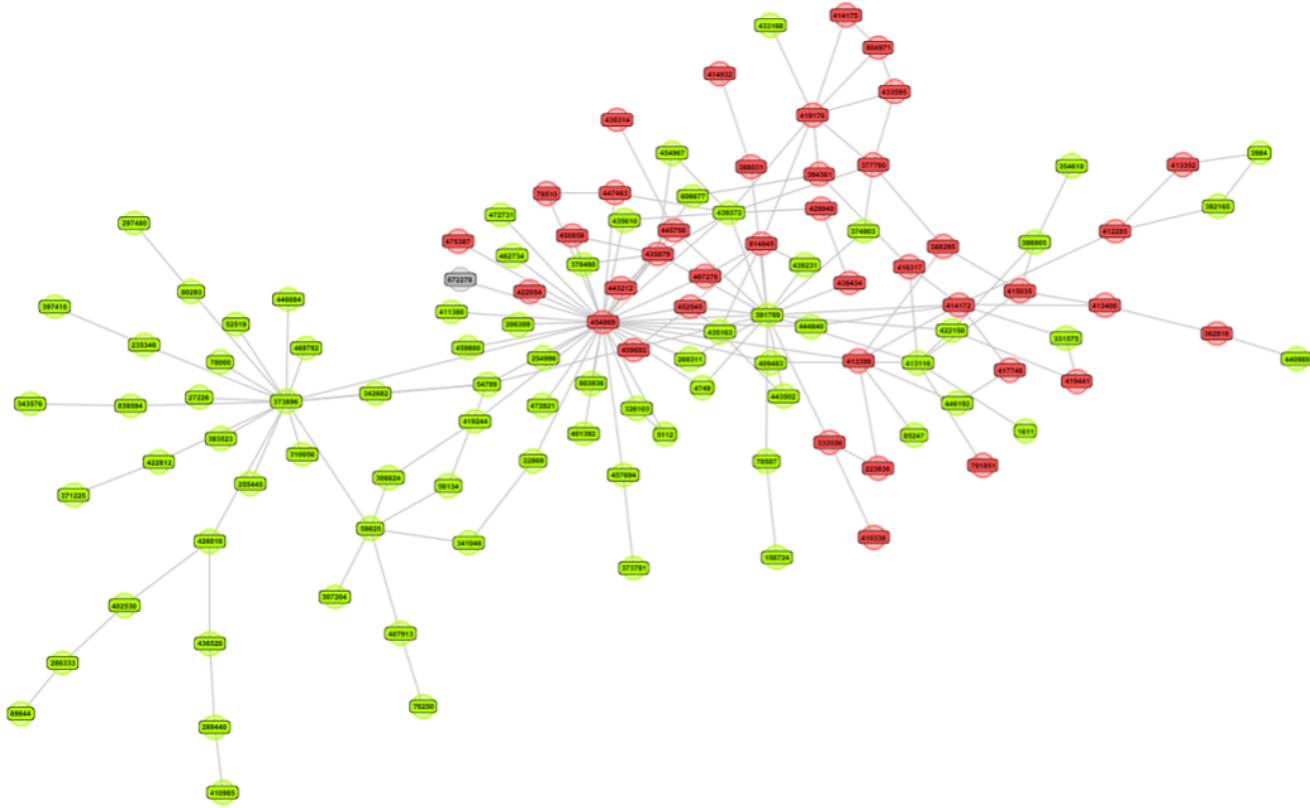
Bart Baesens, Ph.D.

Professor of Data Science, KU Leuven
and University of Southampton



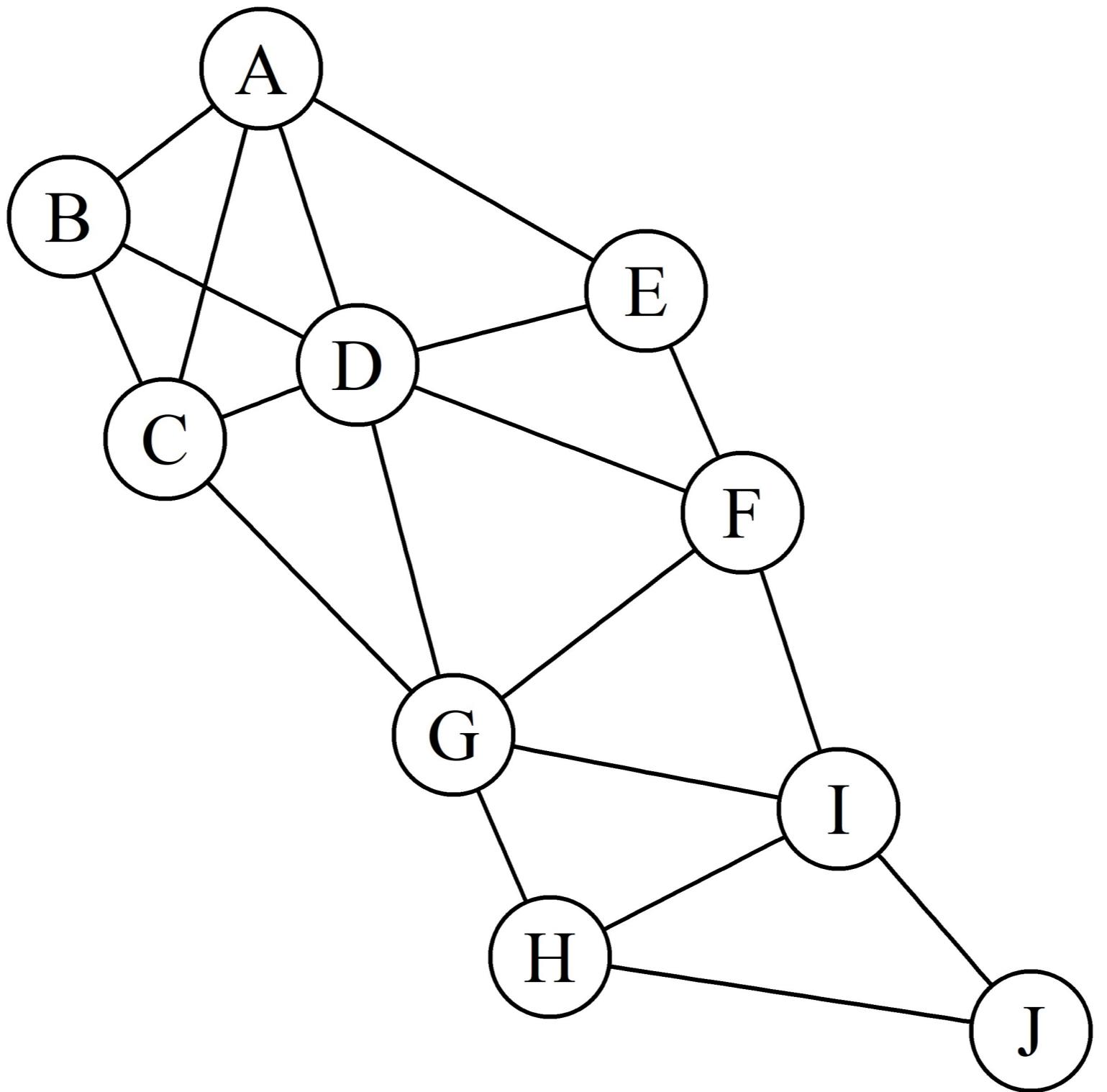
Applications

- Age
- Gender
- Fraud
- Churn
 - Customer defection
 - Companies predict who is most likely to churn using
 1. Machine learning techniques
 2. Social networks



Overview

- Labeled social networks
 - Construct and label networks
 - Network learning
- Homophily
 - Measure relational dependency
 - Heterophilicity and dyadicity
- Network featurization
 - Compute node features
- Predictive modeling with networks
 - Turn a network into a flat dataset
 - Predict churn among customers



Collaboration Network

```
library(igraph);

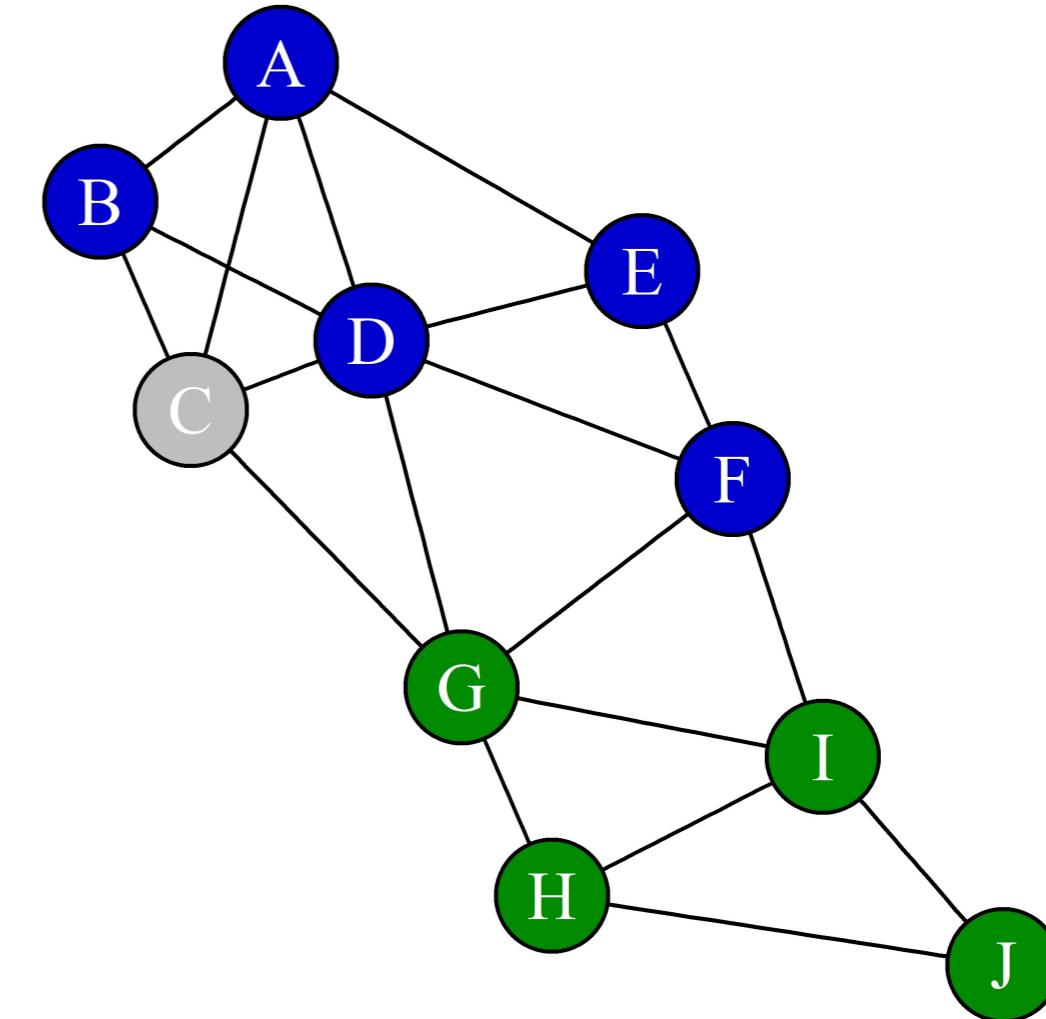
DataScienceNetwork <- data.frame(
  from = c('A', 'A', 'A', 'A', 'B', 'B', 'C', 'C', 'D', 'D', 'D', 'E',
          'F', 'F', 'G', 'G', 'H', 'H', 'I'),
  to = c('B', 'C', 'D', 'E', 'C', 'D', 'D', 'G', 'E', 'F', 'G', 'F', 'G', 'I',
         'I', 'H', 'I', 'J', 'J'))
g <- graph_from_data_frame(DataScienceNetwork, directed = FALSE)
```

```
pos <- cbind(c(2, 1, 1.5, 2.5, 4, 4.5, 3, 3.5, 5, 6),
              c(10.5, 9.5, 8, 8.5, 9, 7.5, 6, 4.5, 5.5, 4))
plot.igraph(g, edge.label = NA, edge.color = 'black', layout = pos,
            vertex.label = V(g)$name, vertex.color = 'white',
            vertex.label.color = 'black', vertex.size = 25)
```

Collaboration Network

```
v(g)$technology <-  
  c('R', 'R', '?', 'R', 'R',  
    'R', 'P', 'P', 'P', 'P')  
v(g)$color <- v(g)$technology
```

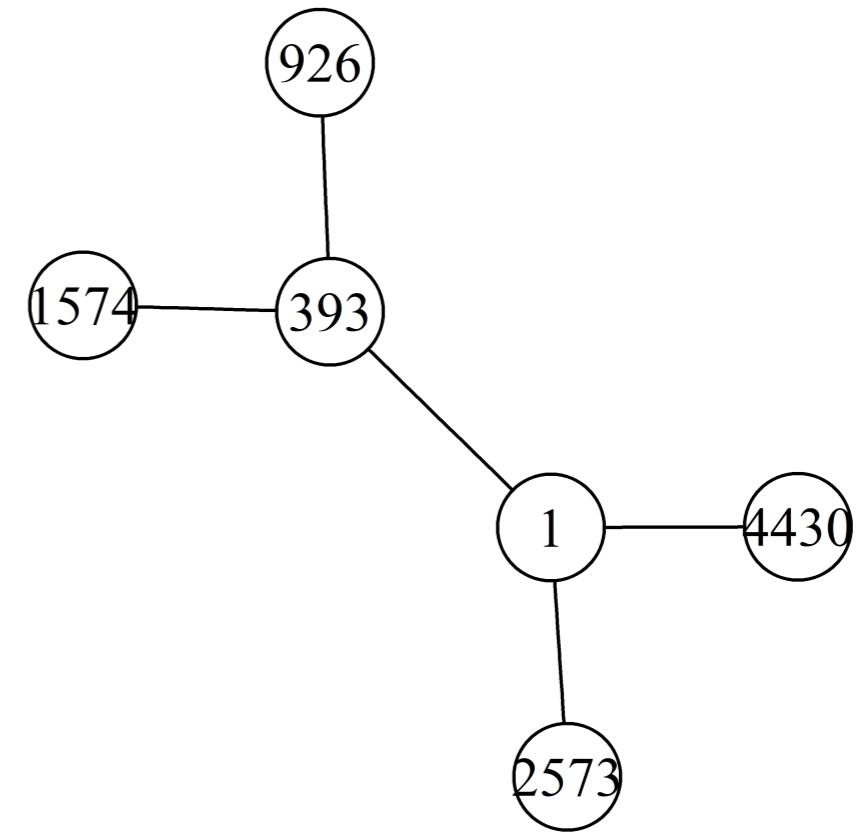
```
v(g)$color <- gsub('R', "blue3", v(g)$color)  
v(g)$color <- gsub('P', "green4", v(g)$color)  
v(g)$color <- gsub('?', "gray", v(g)$color)
```



Churn Network

edgeList

	from	to
1	1	393
2	1	2573
3	1	4430
4	393	926
5	393	1574



Let's practice!

PREDICTIVE ANALYTICS USING NETWORKED DATA IN R

Labeled networks and network learning

PREDICTIVE ANALYTICS USING NETWORKED DATA IN R



María Óskarsdóttir, Ph.D.

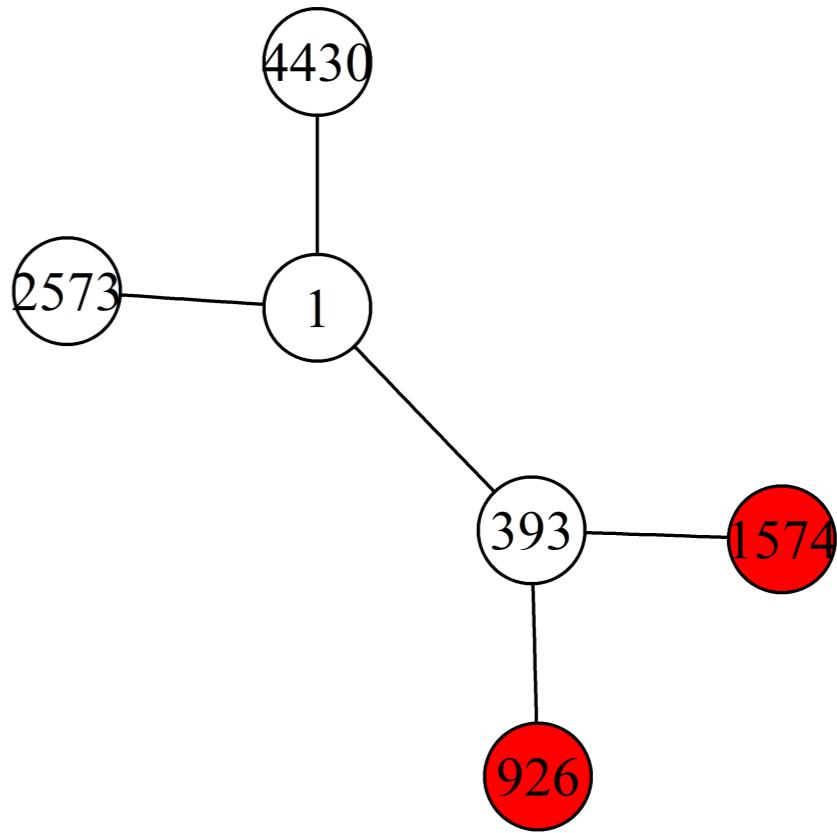
Post-doctoral researcher

customers

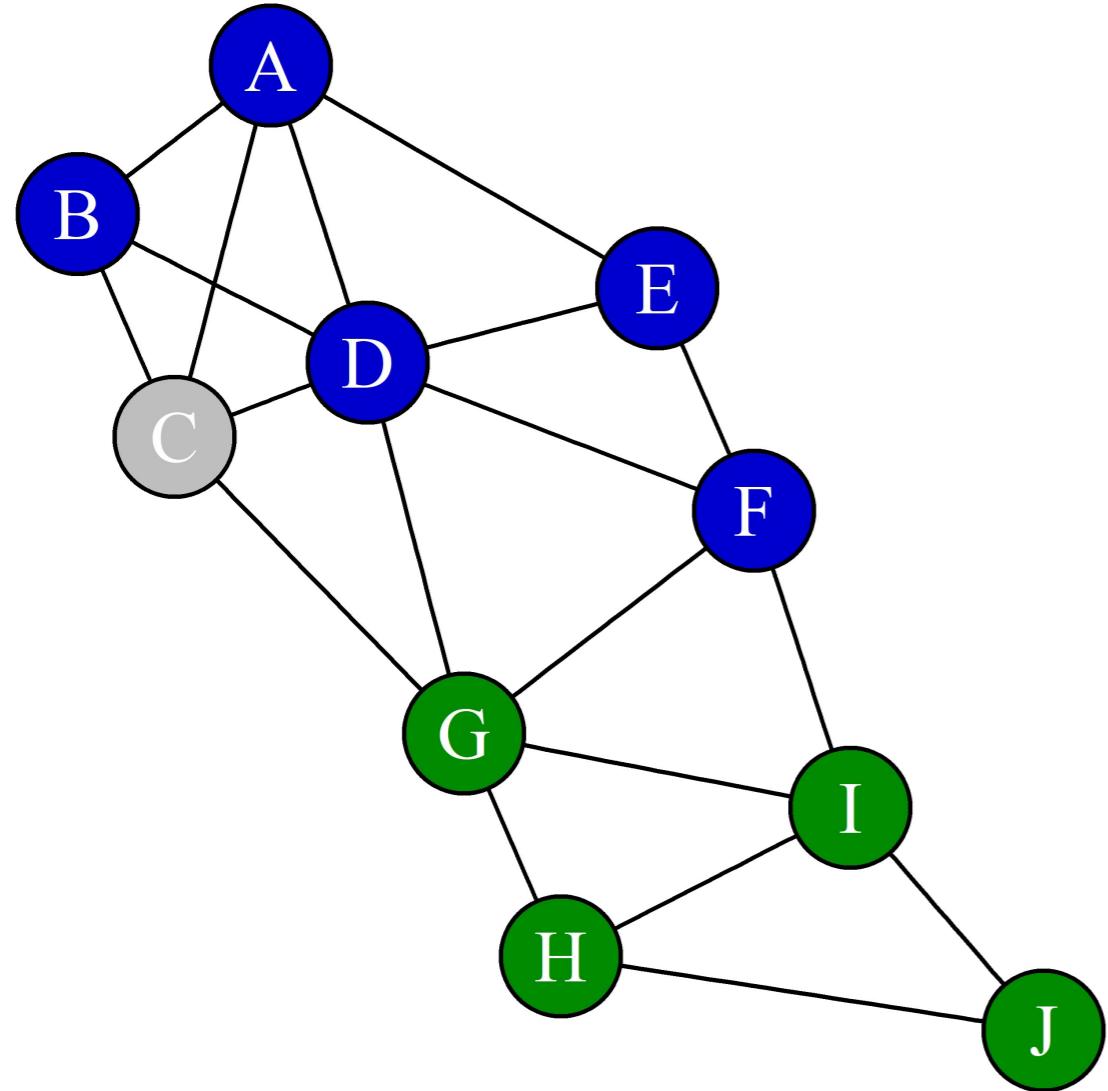
```
  id churn  
1   1    0  
2 393    0  
3 2573   0  
4 4430   0  
5 926    1  
6 1574   1
```

edgeList

```
  from  to  
1   1   393  
2   1   2573  
3   1   4430  
4 393   926  
5 393   1574
```

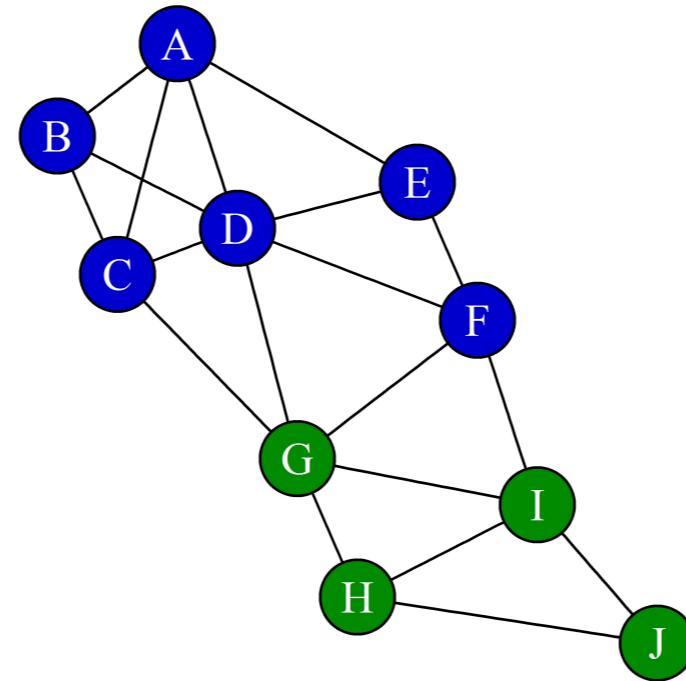


The Relational Neighbor Classifier



- Neighbors of **Cecelia**
 - A,B,D,G
- Neighbors of **Cecelia** that prefer R
 - A, B, D (75%)
- Neighbors of **Cecelia** that prefer Python
 - G (25%)
- **Cecelia** has a higher probability to prefer R

The Relational Neighbor Classifier



```
rNeighbors <- c(4,3,3,5,3,2,3,0,1,0)
pNeighbors <- c(0,0,1,1,0,2,2,3,3,2)
rRelationalNeighbor <- rNeighbors / (rNeighbors + pNeighbors)
rRelationalNeighbor
```

```
1.00 1.00 0.75 0.86 1.00 0.50 0.60 0.00 0.00 0.00
```

Let's practice!

PREDICTIVE ANALYTICS USING NETWORKED DATA IN R

Challenges of network-based inference

PREDICTIVE ANALYTICS USING NETWORKED DATA IN R



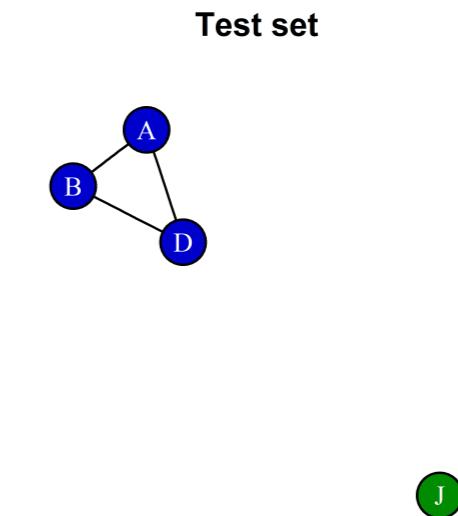
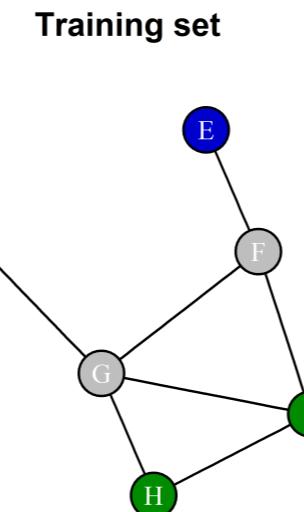
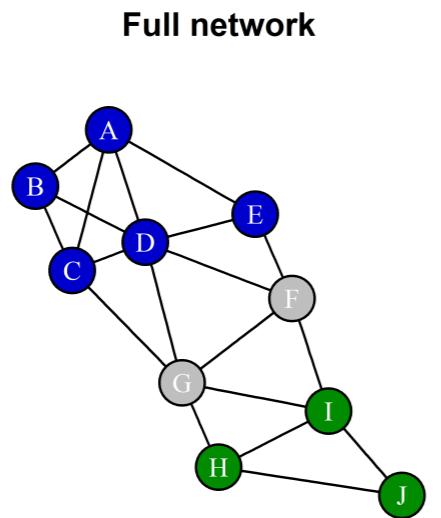
María Óskarsdóttir, Ph.D.

Post-doctoral researcher

First challenge

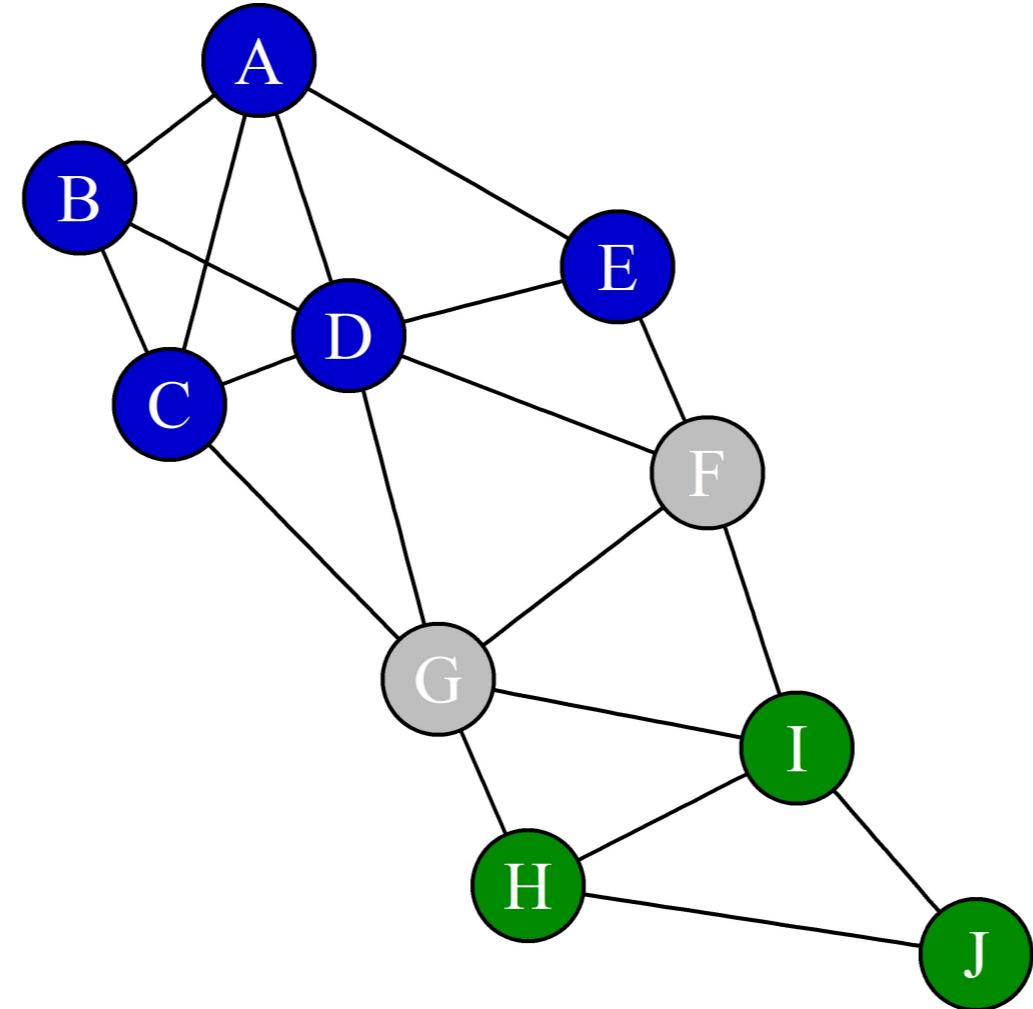
Splitting the data!

```
set.seed(1001)  
sampleVertices <- sample(1:10, 6, replace=FALSE)  
plot(induced_subgraph(g, V(g)[sampleVertices]))  
plot(induced_subgraph(g, V(g)[-sampleVertices]))
```



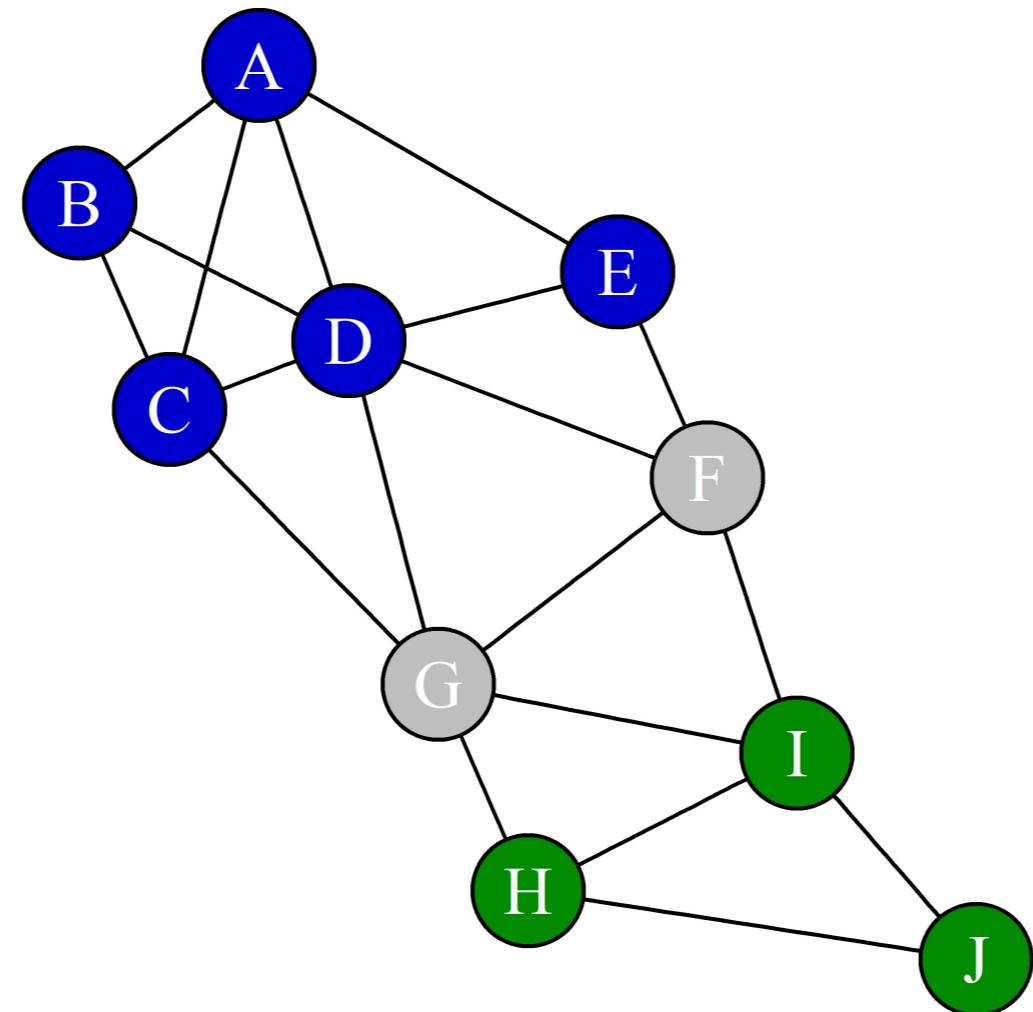
Second challenge

The observations in the dataset are not independent and identically distributed (iid)

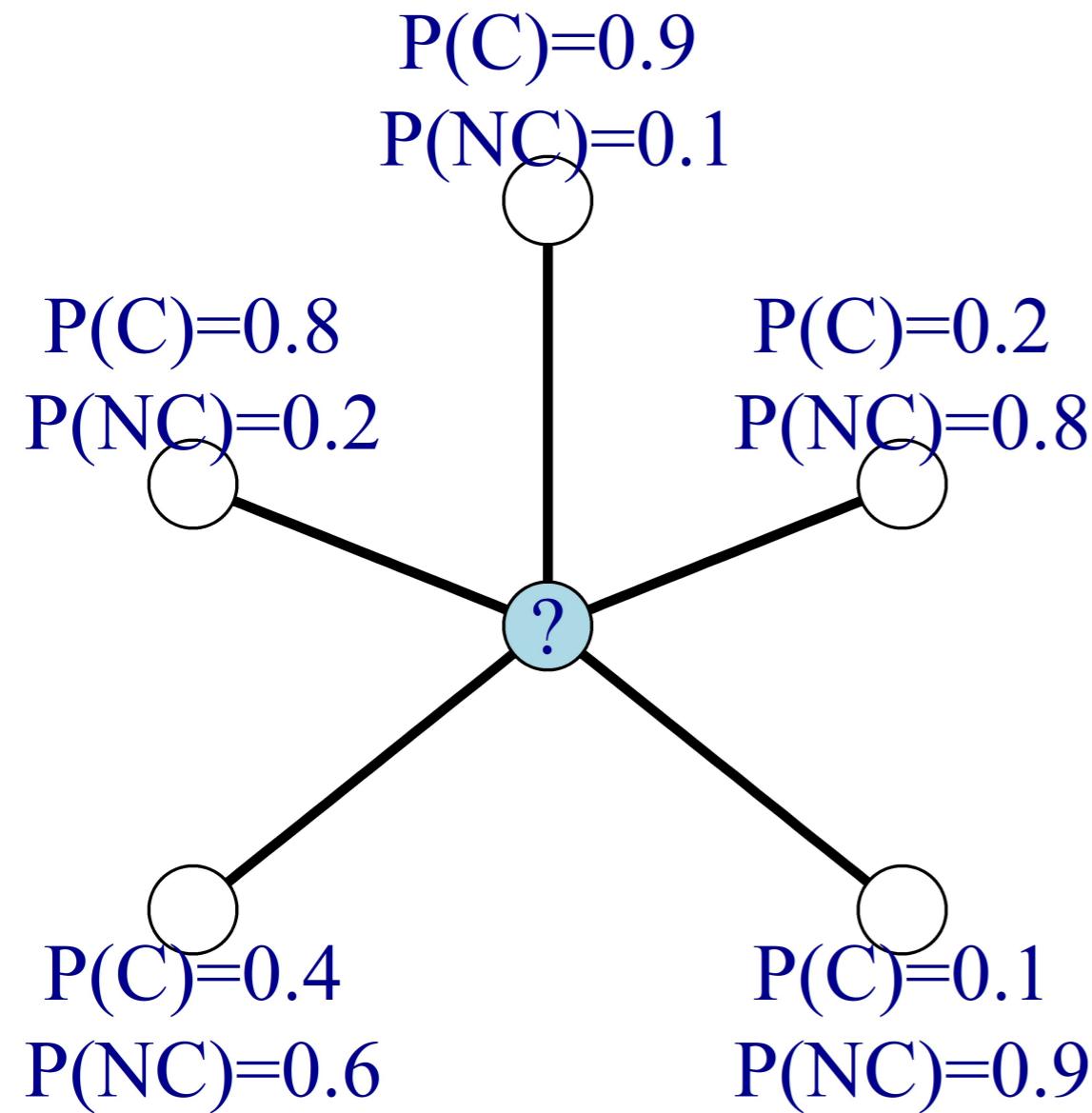


Third challenge

Collective Inference!



Probabilistic relational neighbor classifier



```
# probability churn (C)  
(0.9 + 0.2 + 0.1 + 0.4 + 0.8) / 5
```

0.48

```
# probability non-churn (NC)  
(0.1 + 0.8 + 0.9 + 0.6 + 0.2) / 5
```

0.52

Let's practice!

PREDICTIVE ANALYTICS USING NETWORKED DATA IN R

Homophily

PREDICTIVE ANALYTICS USING NETWORKED DATA IN R



Bart Baesens, Ph.D.

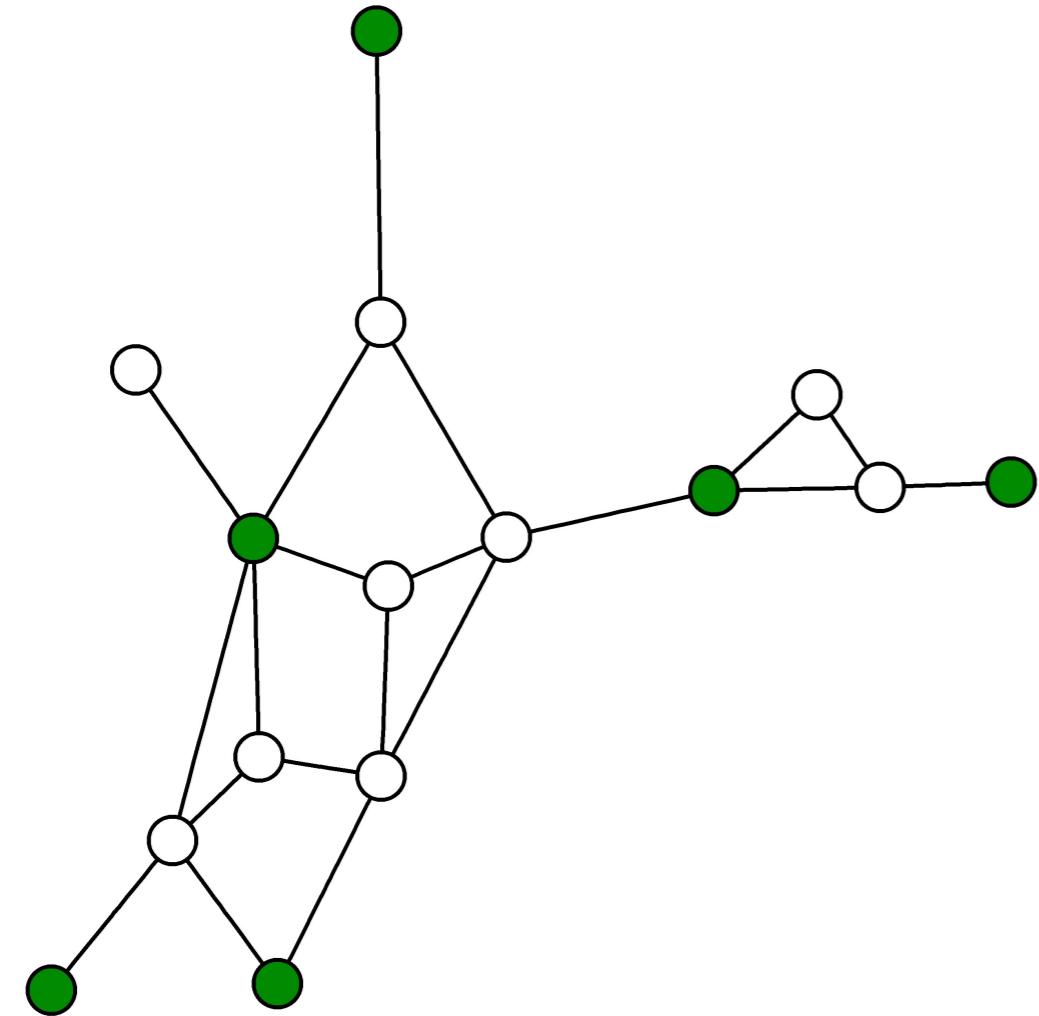
Professor of Data Science, KU Leuven
and University of Southampton

Homophily explained

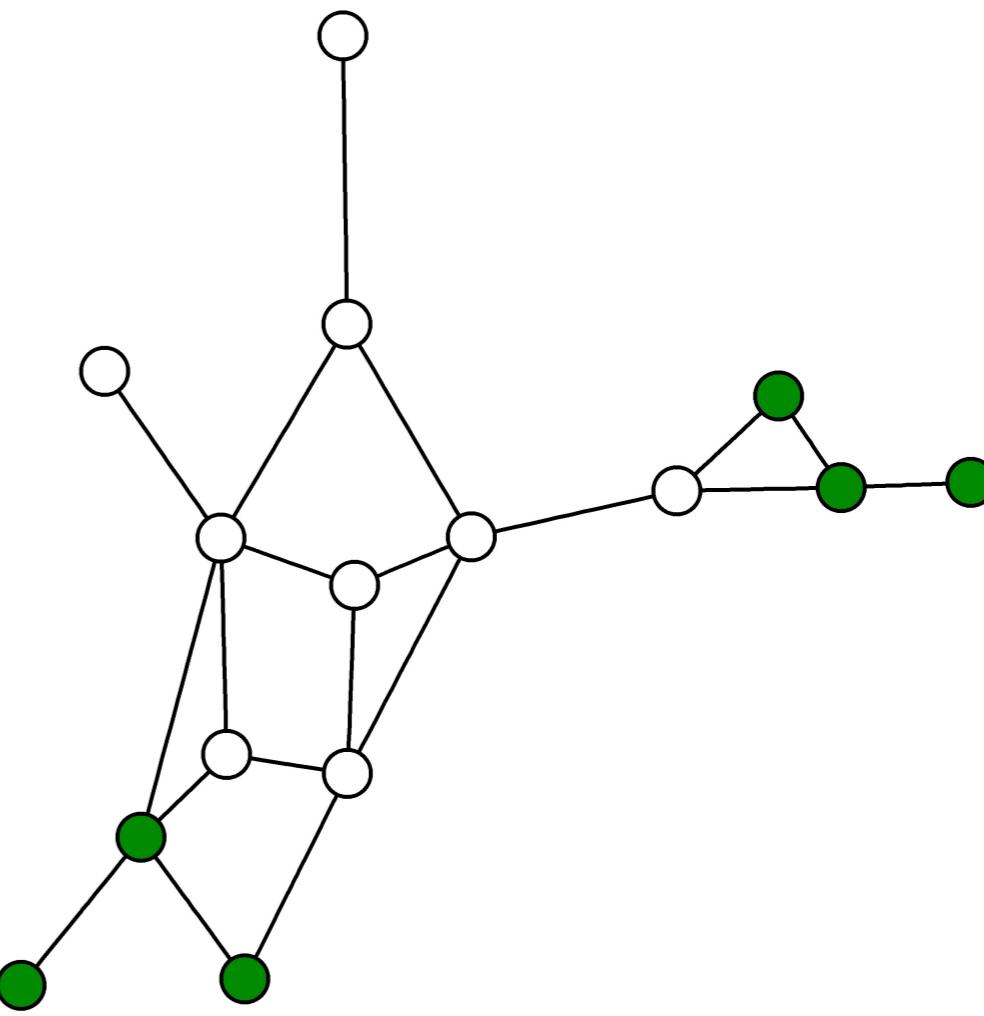
Birds of a feather flock together

- Share common property, hobbies, interest, origin, etc.
- Depends on:
 - Connectedness between nodes with **same** label
 - Connectedness between nodes with **opposite** labels

Homophilic Networks



- Not Homophilic



- Homophilic

```
names <- c('A','B','C','D','E','F','G','H','I','J')
tech <- c(rep('R',6),rep('P',4))
DataScientists <- data.frame(name=names,technology=tech)
DataScienceNetwork <- data.frame(
  from=c('A','A','A','A','B','B','C','C','D','D',
         'D','E','F','F','G','G','H','H','I'),
  to=c('B','C','D','E','C','D','D','G','E','F',
        'G','F','G','I','I','H','I','J','J'),
  label=c(rep('rr',7),'rp','rr','rr','rp','rr','rp',rep('pp',5)))
g <- graph_from_data_frame(DataScienceNetwork,directed = FALSE)
```

Add the technology as a node attribute

```
V(g)$label <- as.character(DataScientists$technology)
V(g)$color <- V(g)$label
V(g)$color <- gsub('R',"blue3",V(g)$color)
V(g)$color <- gsub('P',"green4",V(g)$color)
```

Types of edges

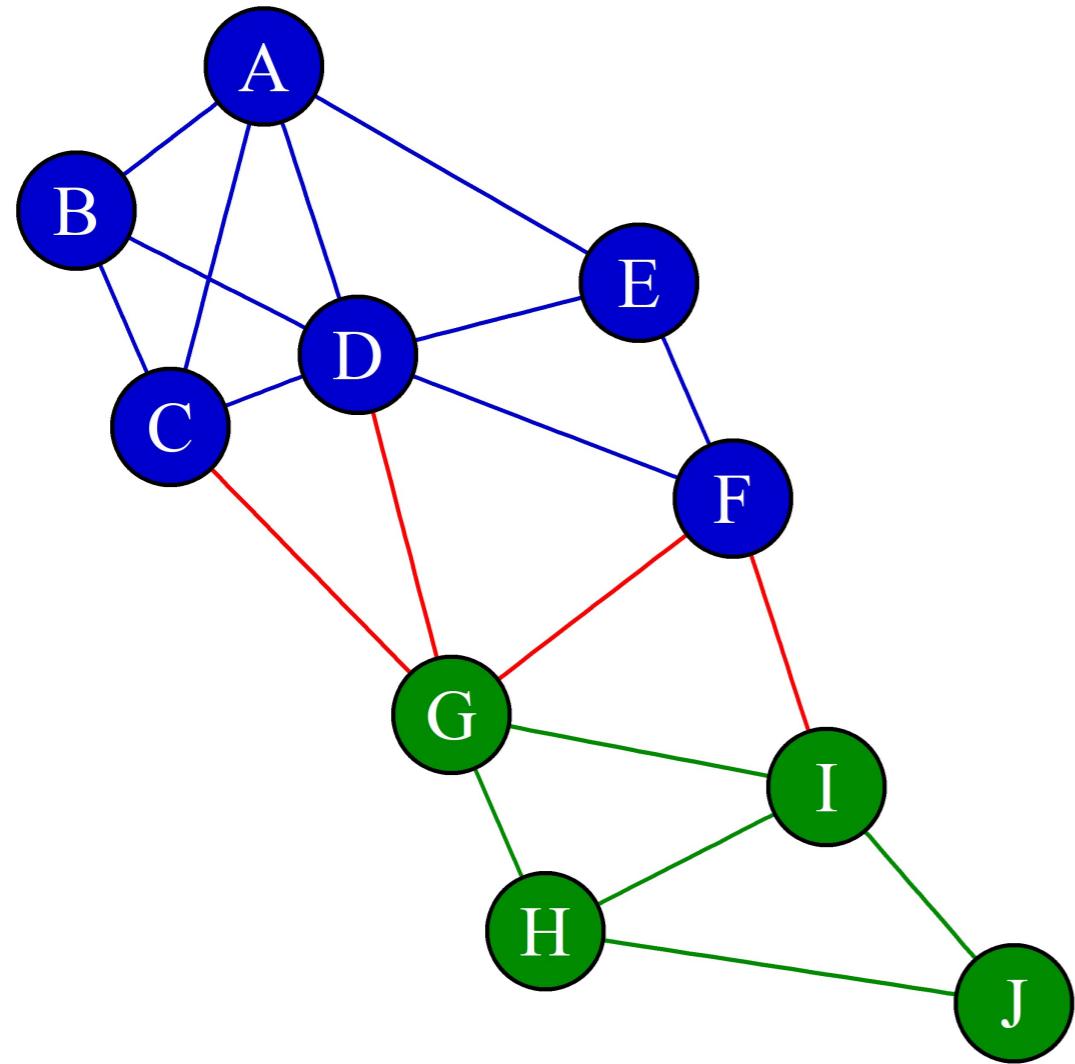
Code to color the edges

```
E(g)$color<-E(g)$label  
E(g)$color=gsub('rp','red',E(g)$color)  
E(g)$color=gsub('rr','blue3',E(g)$color)  
E(g)$color=gsub('pp','green4',E(g)$color)
```

Code to visualize the network

```
pos<-cbind(c(2,1,1.5,2.5,4,4.5,3,3.5,5,6),  
c(10.5,9.5,8,8.5,9,7.5,6,4.5,5.5,4))  
  
plot(g,edge.label=NA,vertex.label.color='white',  
layout=pos, vertex.size = 25)
```

Counting edge types



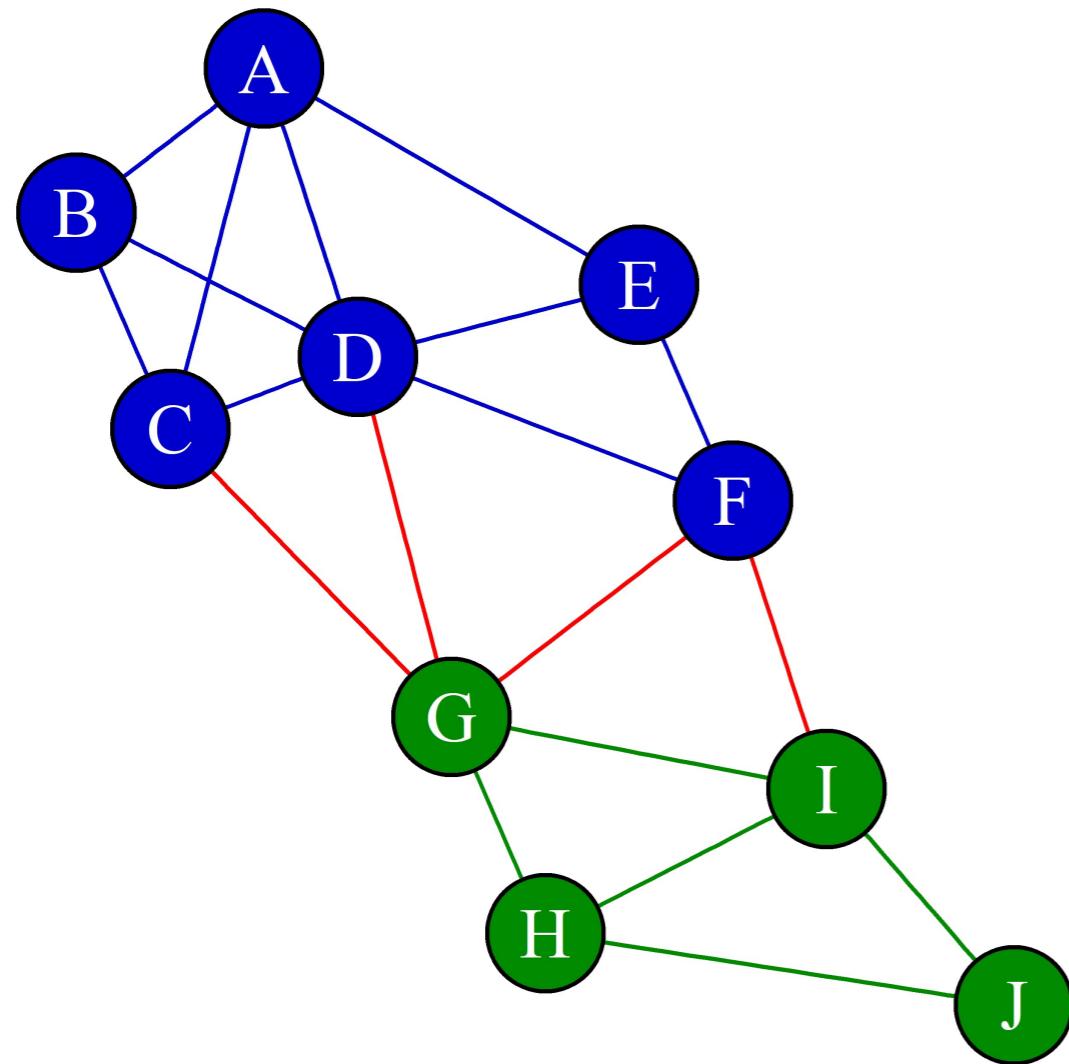
```
# R edges  
edge_rr<-sum(E(g)$label=='rr')
```

```
# Python edges  
edge_pp<-sum(E(g)$label=='pp')
```

```
# cross label edges  
edge_rp<-sum(E(g)$label=='rp')
```

- `edge_rr= 10`
- `edge_pp= 5`
- `edge_rp= 4`

Network connectance



$$p = \frac{2 \cdot \text{edges}}{\text{nodes}(\text{nodes}-1)}$$

```
p <- 2*edges/nodes*(nodes-1)
```

- $p = 0.42$
- Number of edges in a fully connected network: $\binom{\text{nodes}}{2} = \frac{\text{nodes}(\text{nodes}-1)}{2}$

Let's practice!

PREDICTIVE ANALYTICS USING NETWORKED DATA IN R

Dyadicity

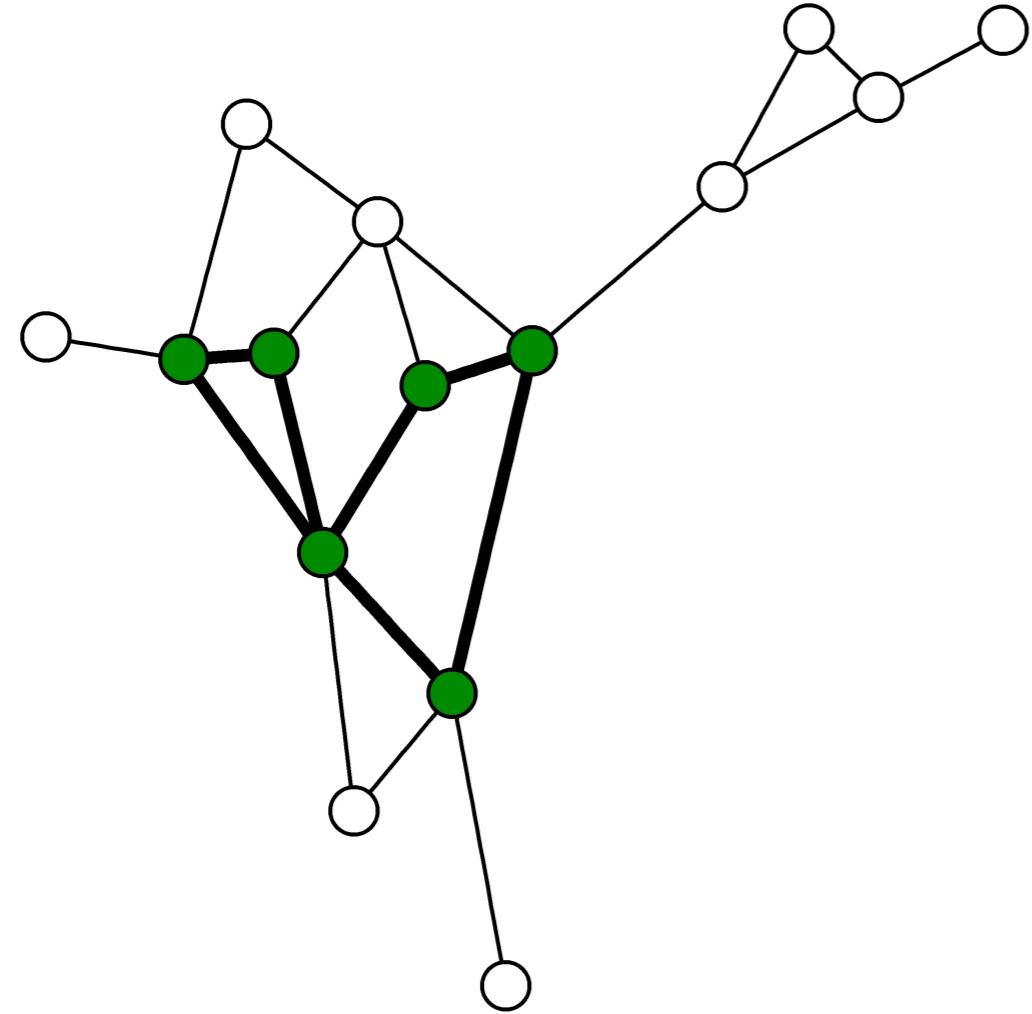
PREDICTIVE ANALYTICS USING NETWORKED DATA IN R



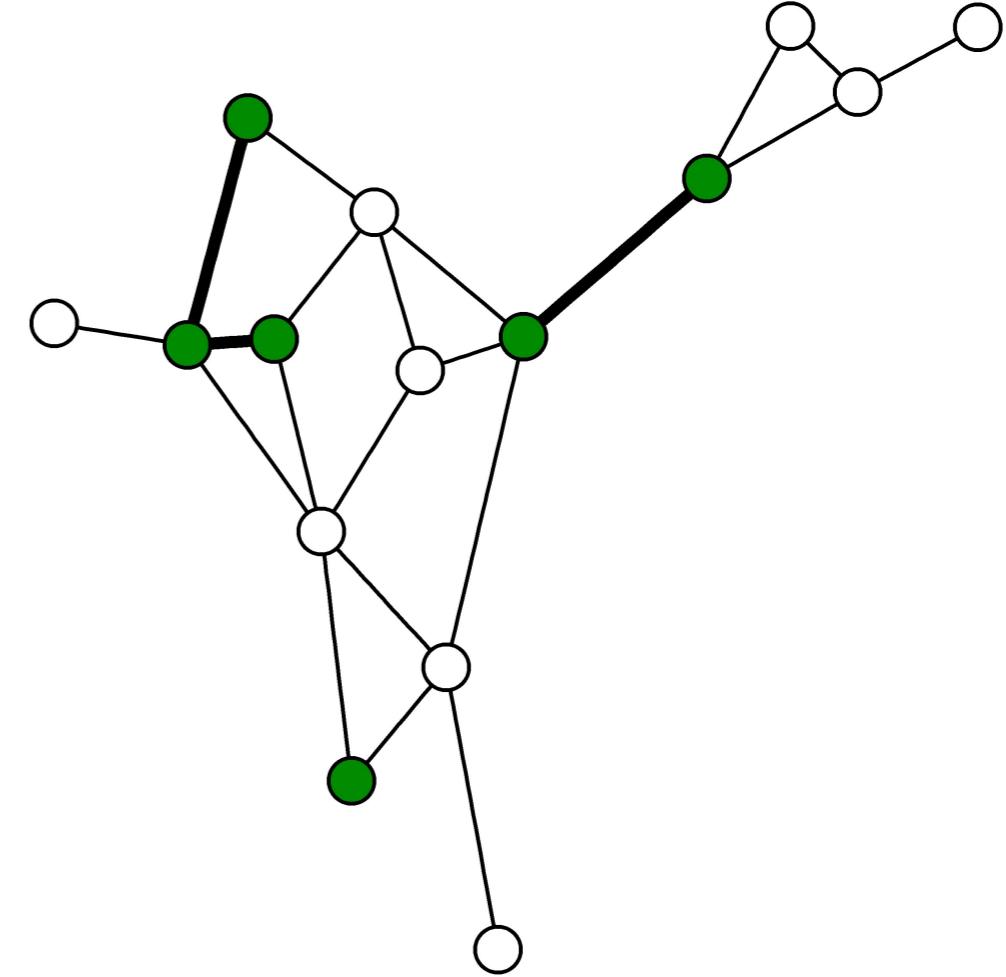
María Óskarsdóttir, Ph.D.

Post-doctoral researcher

Dyadicity



7 edges between green nodes



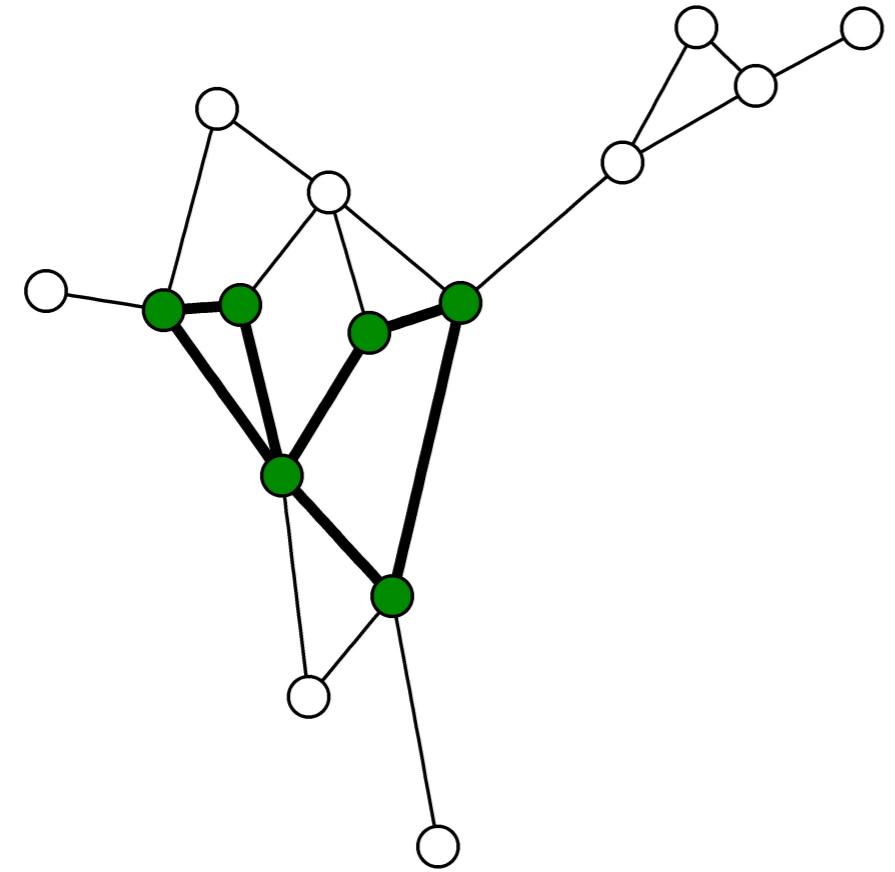
3 edges between green nodes

Dyadicity

Connectedness between nodes with the **same** label compared to what is expected in a random configuration of the network

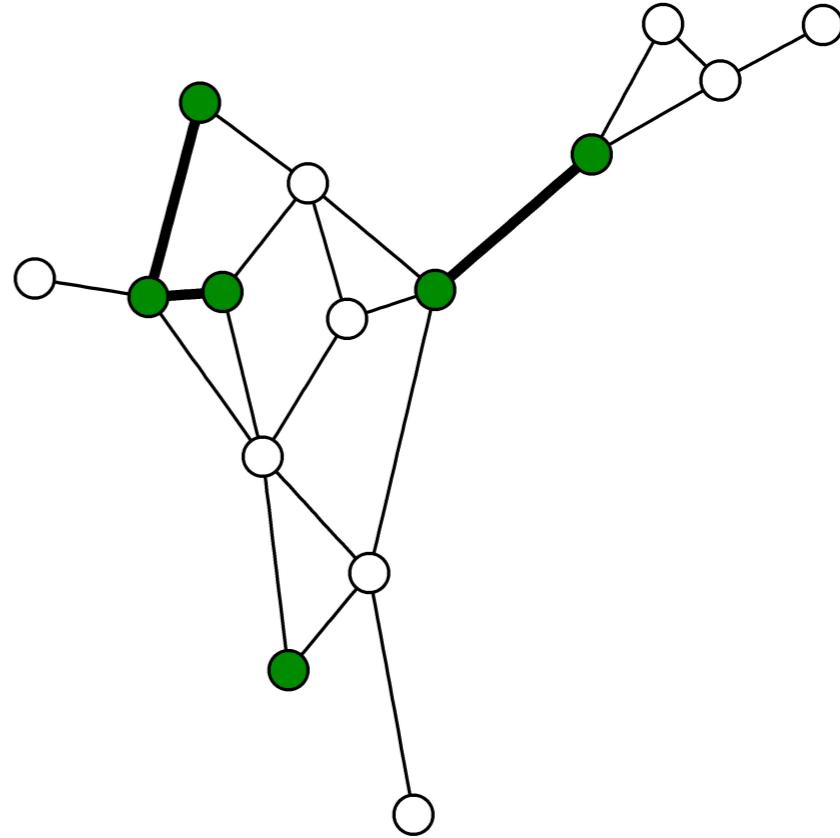
- Expected number of same label edges: $\binom{n_g}{2} \cdot p = \frac{n_g(n_g-1)}{2} \cdot p$
- Example:
 - Network with 9 white nodes, 6 green nodes, 21 edges, and connectance $p = 0.2$
 - Expected number of edges connecting two green nodes is 3 ($= \frac{6 \cdot 5 \cdot p}{2}$)
- Dyadicity equals the actual number of same label edges divided by the expected number of same label edge
 - $D = \frac{\text{number of same label edges}}{\text{expected number of same label edges}}$

Dyadicity



7 edges between green nodes

- $D = 7/3 = 2.33$



3 edges between green nodes

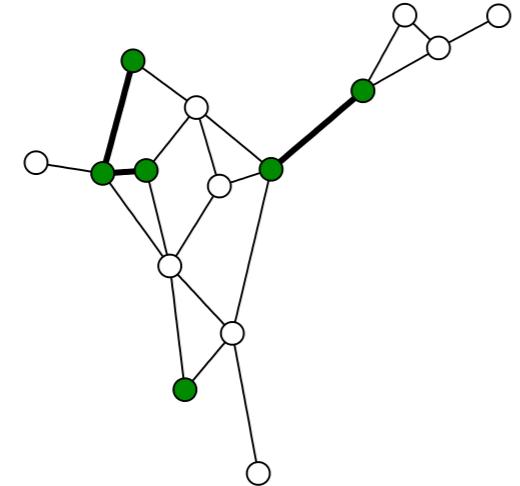
- $D = 3/3 = 1$

Types of Dyadicity

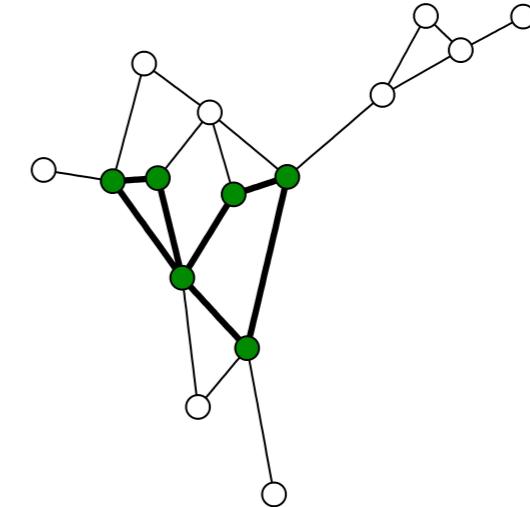
Three scenarios

1. $D > 1 \Rightarrow$ Dyadic
2. $D \simeq 1 \Rightarrow$ Random
3. $D < 1 \Rightarrow$ Anti-Dyadic

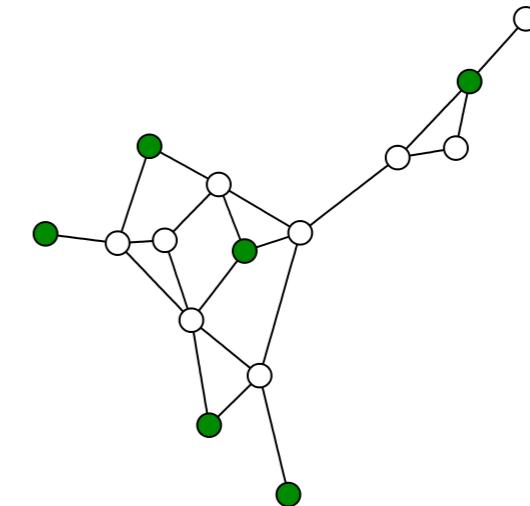
$$D = 1$$



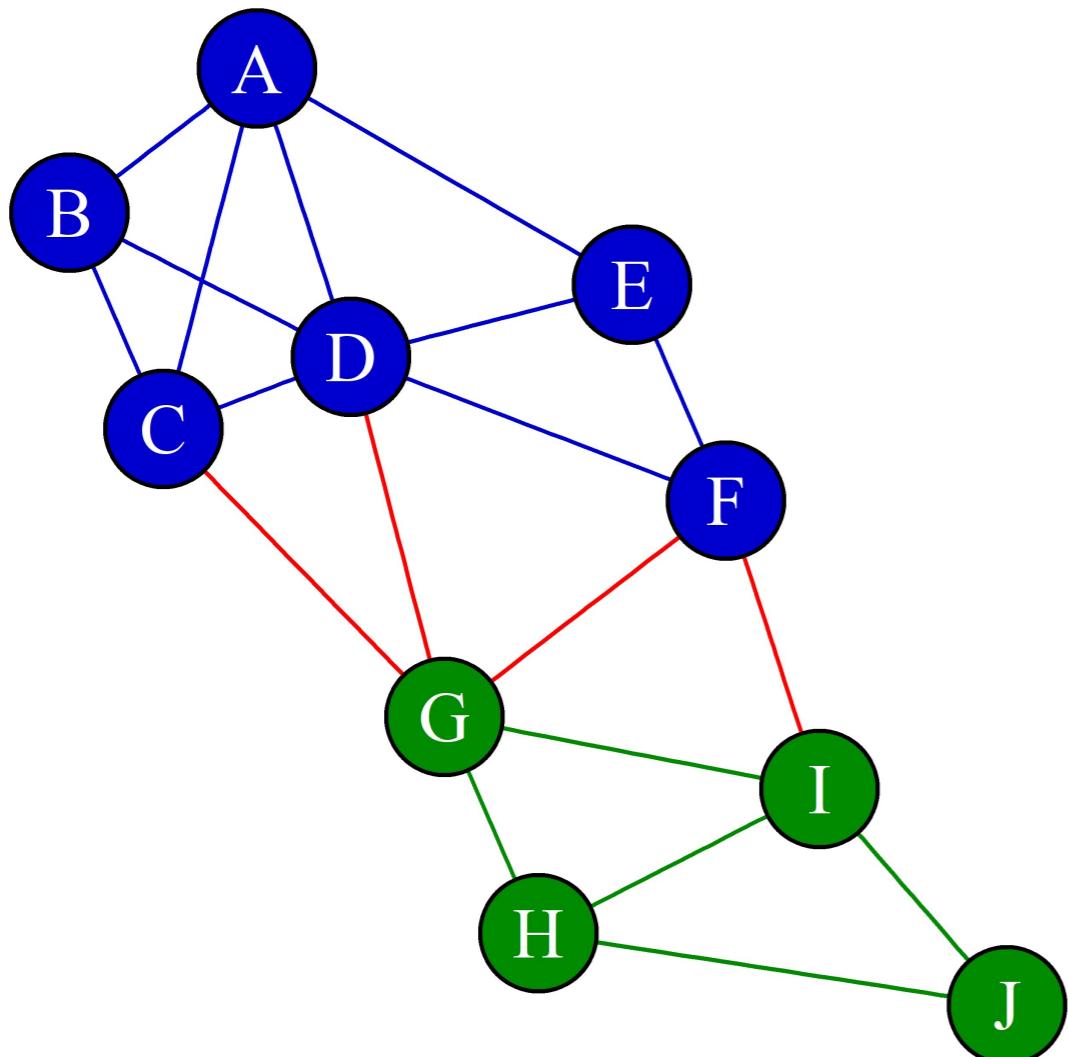
$$D = 2.33$$



$$D = 0$$



Dyadicity in the Network of Data Scientists



```
p <- 2 * 19 / (10 * 9)
expectedREdges <- 6 * 5 / 2 * p
expectedPEdges <- 4 * 3 / 2 * p
dyadicityR <- rEdges / expectedREdges
dyadicityP <- pEdges / expectedPEdges
dyadicityR
```

1.578947

dyadicityP

1.973684

Let's practice!

PREDICTIVE ANALYTICS USING NETWORKED DATA IN R

Heterophilicity

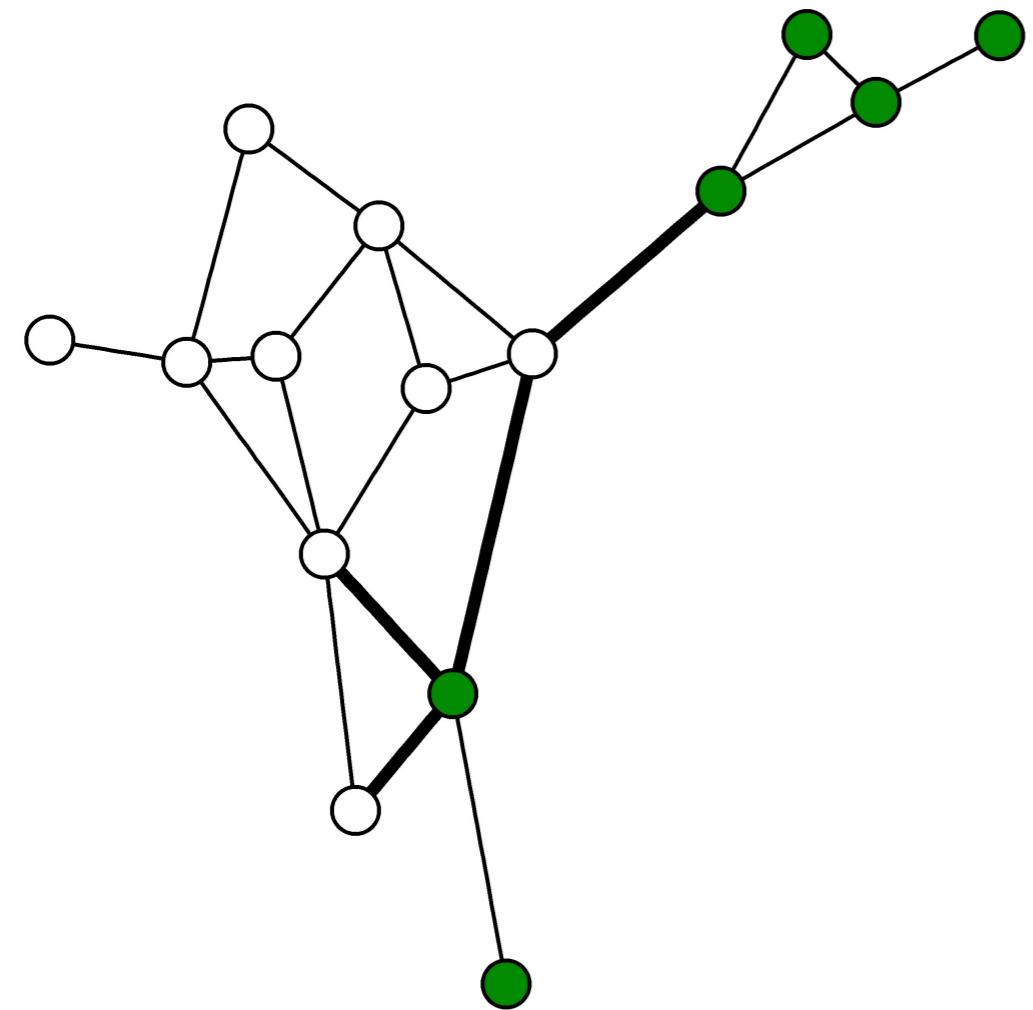
PREDICTIVE ANALYTICS USING NETWORKED DATA IN R



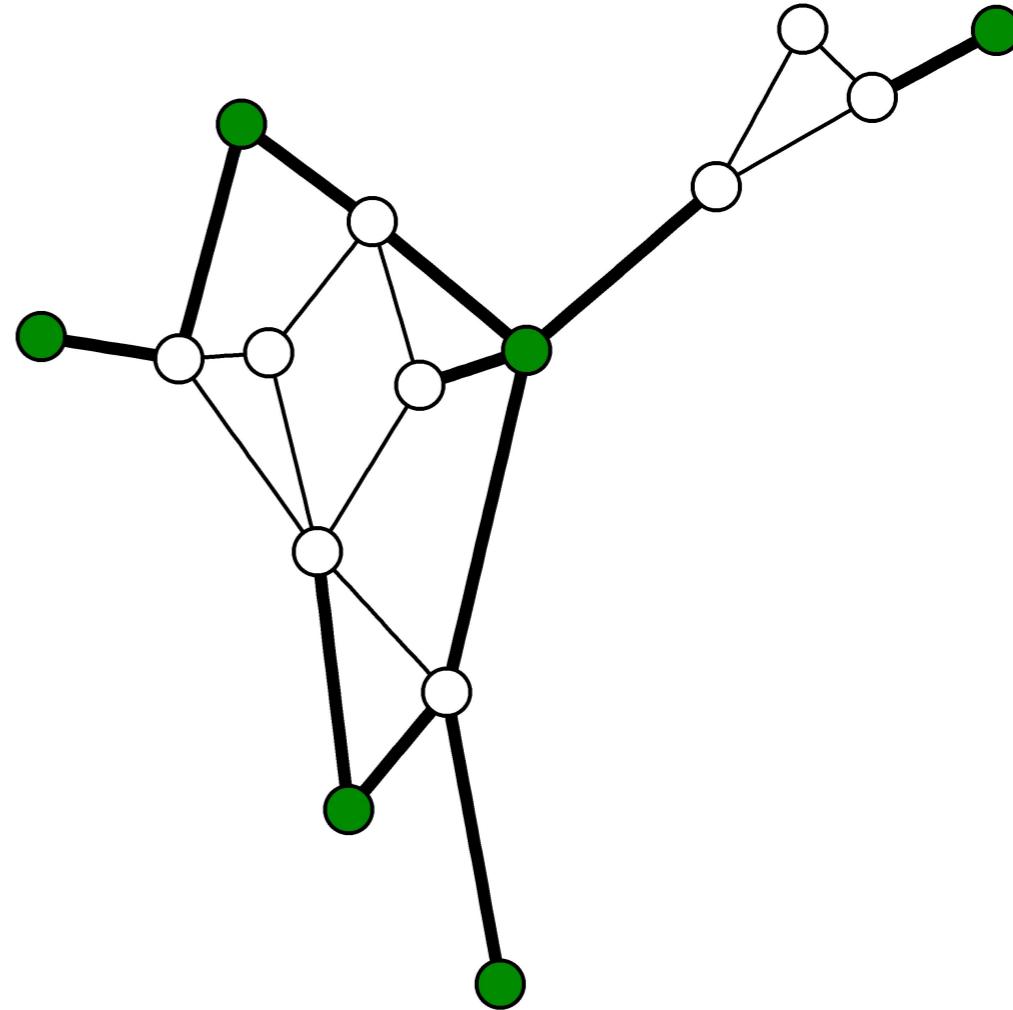
María Óskarsdóttir, Ph.D.

Post-doctoral researcher

Heterophilicity



4 cross label edges



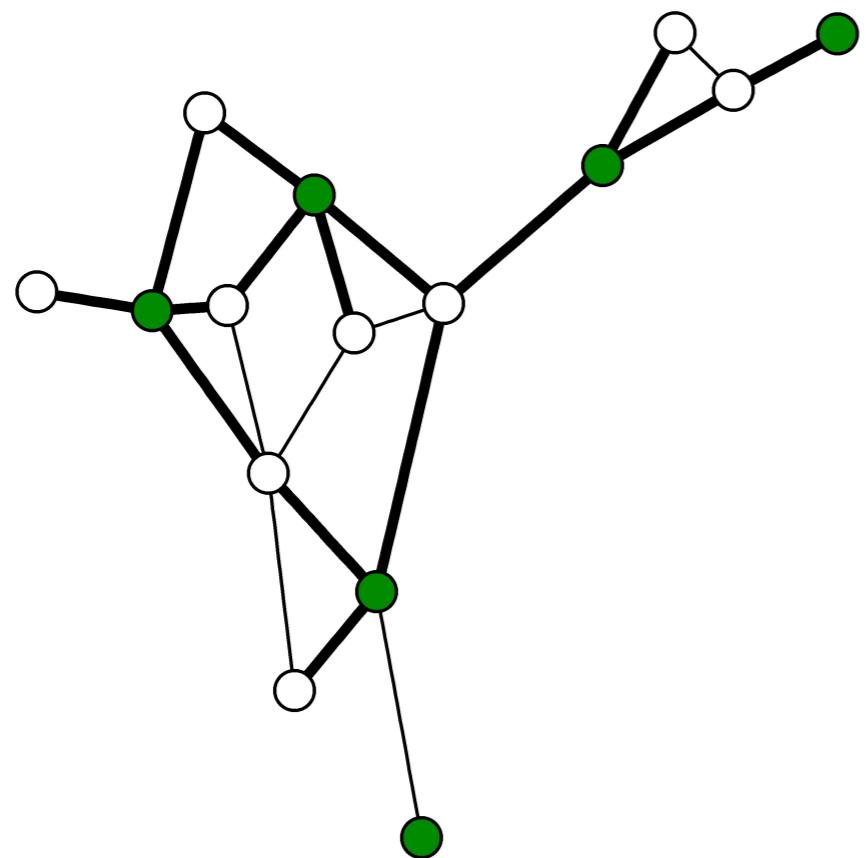
11 cross label edges

Heterophilicity

Connectedness between nodes with **different** labels compared to what is expected for a random configuration of the network

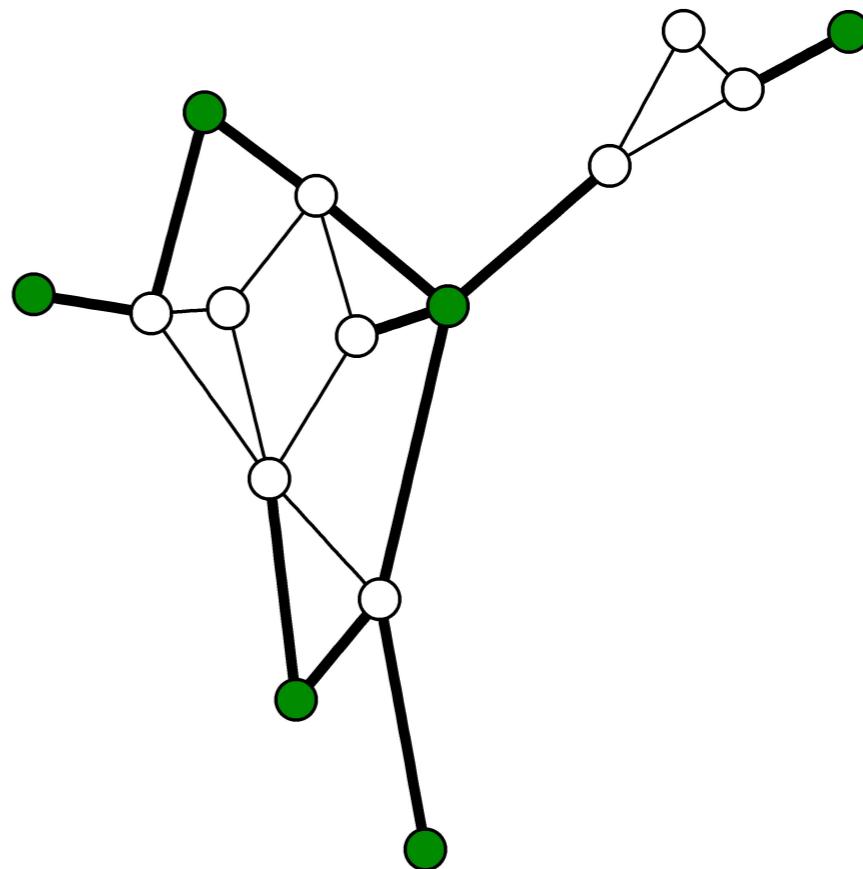
- Expected number of cross label edges = $n_w n_g p$
- Example:
 - Network with 9 white nodes, 6 green nodes, 21 edges, and connectance $p = 0.2$
 - Expected number of cross label edges is 11 ($= 9 \cdot 6 \cdot p$)
- Heterophilicity equals the actual number of cross label edges divided by the expected number of cross label edges
 - $$H = \frac{\text{number of cross label edges}}{\text{expected number of cross label edges}}$$

Heterophilicity



15 cross label edges

- $H = 15/11 = 1.39$



11 cross label edges

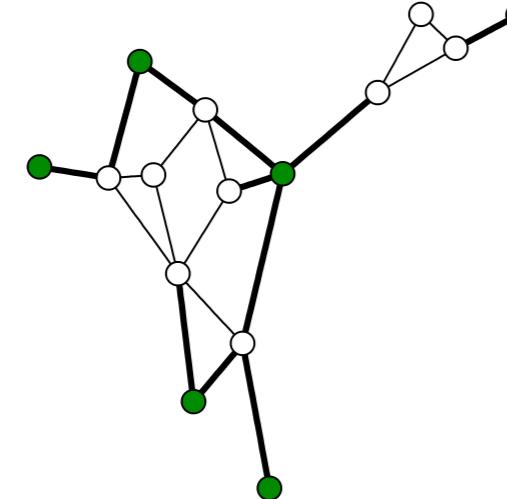
- $H = 11/11 = 1.02$

Types of Heterophilicity

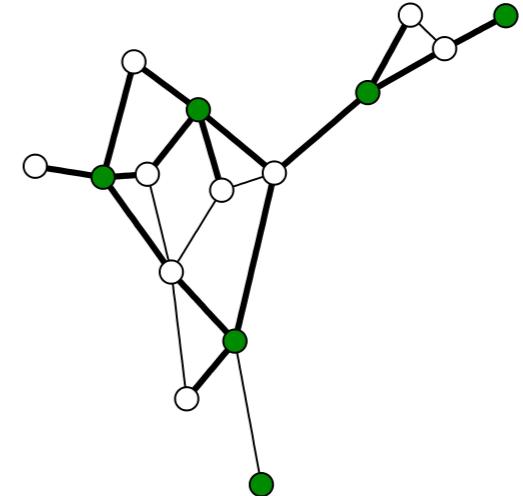
Three scenarios

1. $H > 1 \Rightarrow$ Heterophilic
2. $H \approx 1 \Rightarrow$ Random
3. $H < 1 \Rightarrow$ Heterophobic

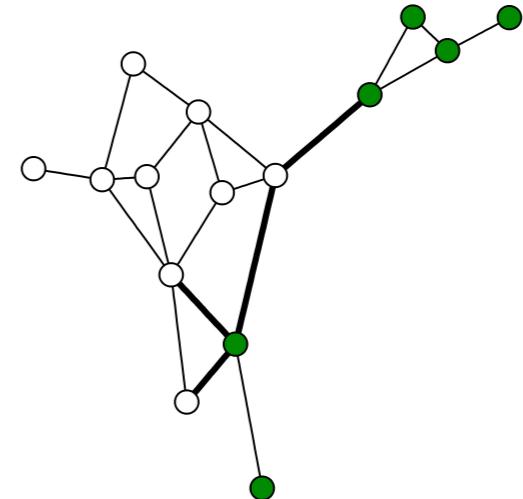
$H = 1.02$



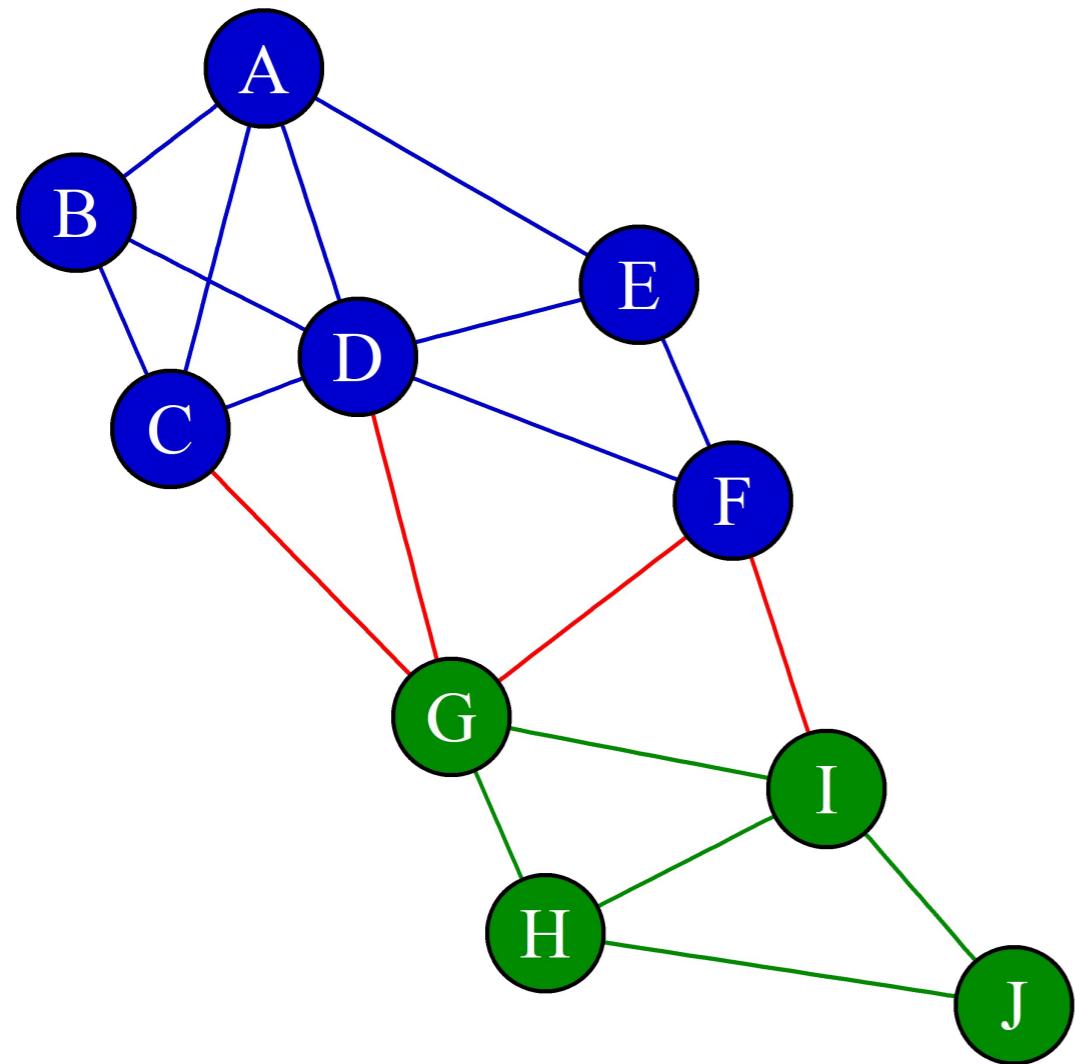
$H = 1.39$



$H = 0.37$



Heterophilicity in the network of data scientists



$p < -2 * 19 / (10 * 9)$

$m_rp < -6 * 4 * p$

$H_rp < -edge_rp / m_rp$

H_rp

0.3947368

Let's practice!

PREDICTIVE ANALYTICS USING NETWORKED DATA IN R

Summary of homophily

PREDICTIVE ANALYTICS USING NETWORKED DATA IN R

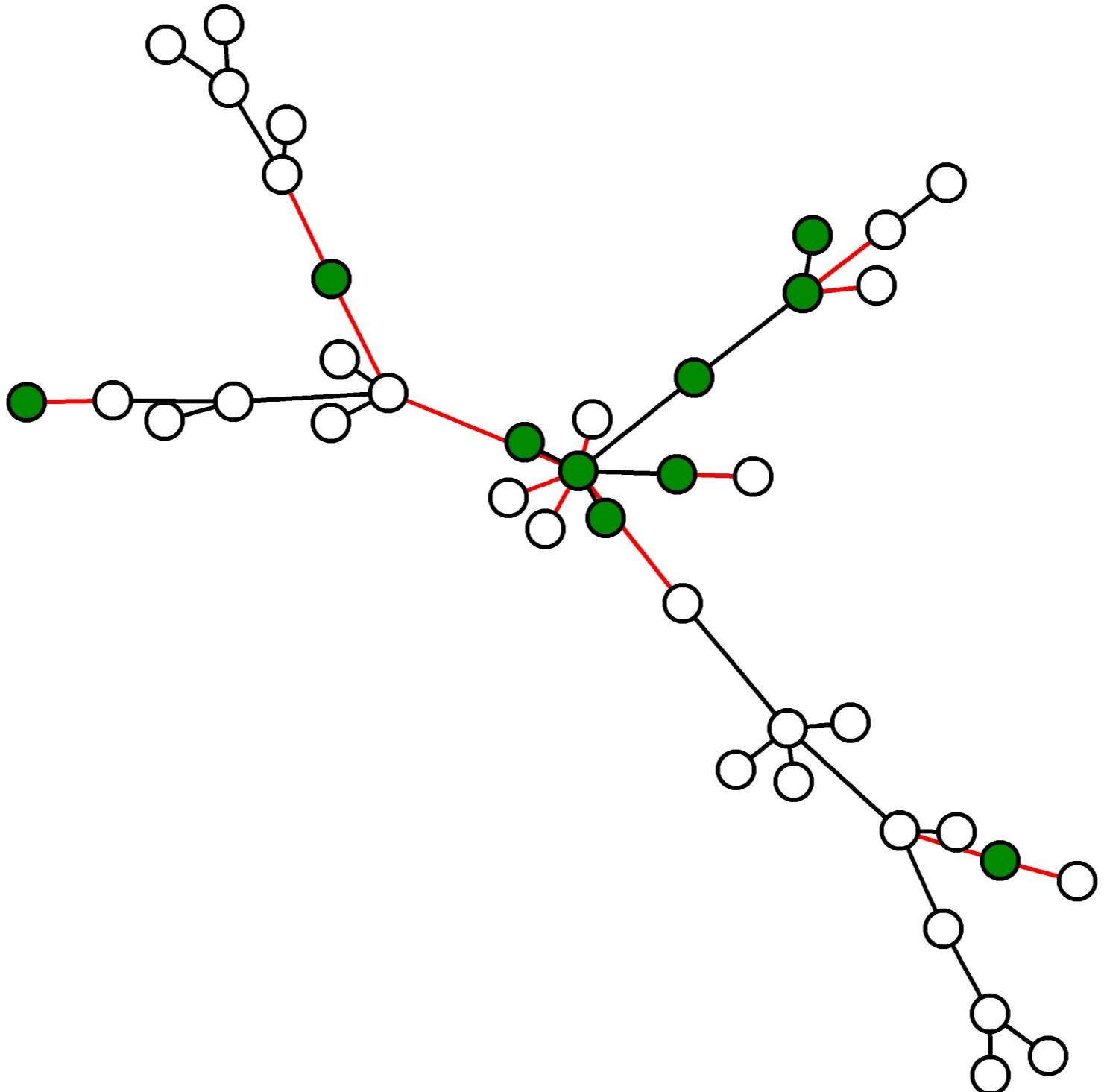


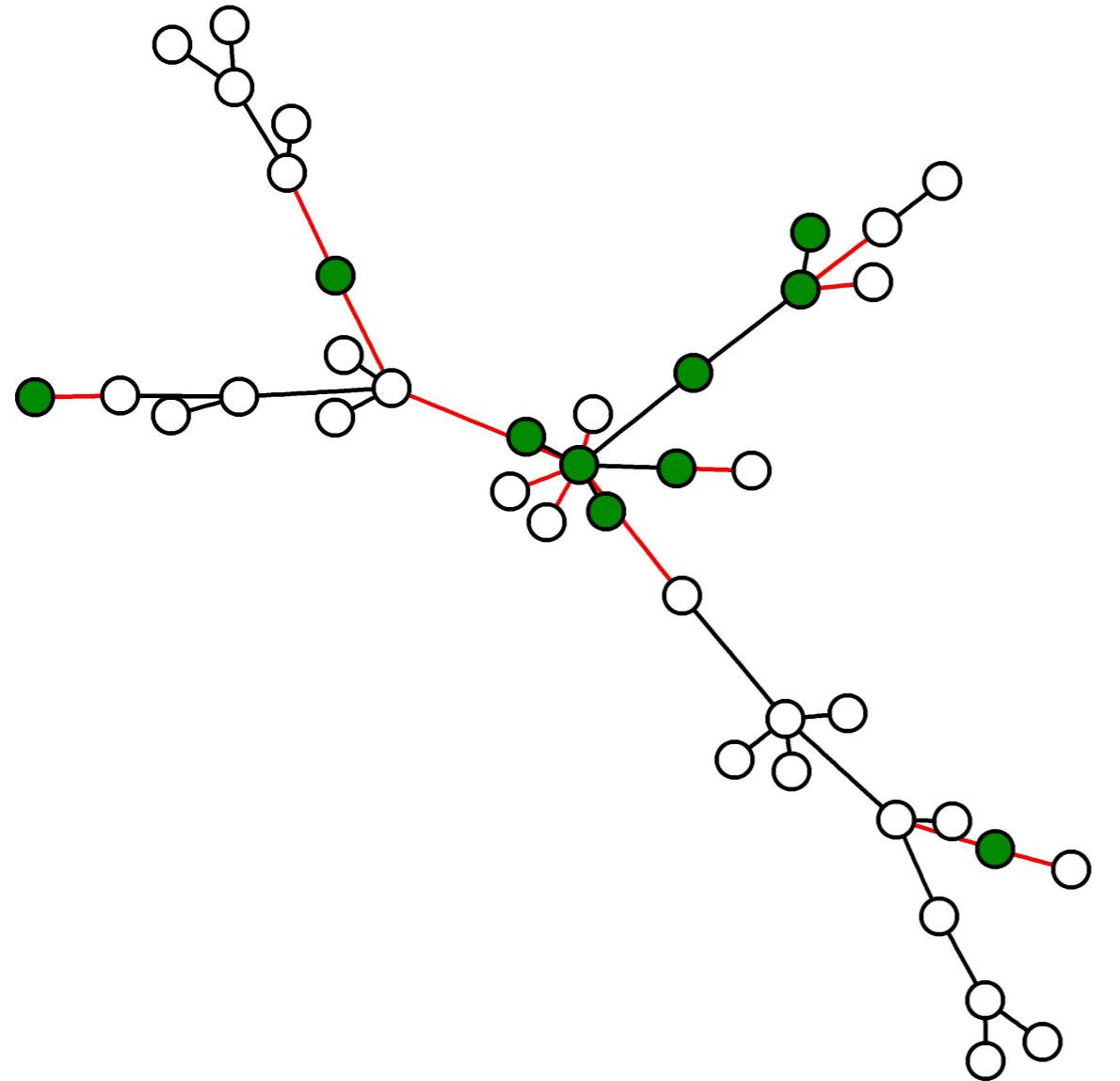
María Óskarsdóttir, Ph.D.

Postdoctoral researcher

Can I do predictive analytics with my network?

- Are the relationships between nodes important?
- Are the labels randomly spread through the network or is there some structure?
- Is the network homophilic?**





```
# Heterophilicity
e_mixed / m_mixed
```

```
N <- 40
E <- 39
n_green <- 10
n_white <- 30
e_green <- 6
e_mixed <- 13
p <- 2 * E / N / (N-1)
m_green <- n_green * (n_green-1)/2 * p
m_mixed <- n_green * n_white * p
# Dyadicity
e_green / m_green
```

2.666667

0.866667

⇒ Homophilic

Let's practice!

PREDICTIVE ANALYTICS USING NETWORKED DATA IN R

Basic Network features

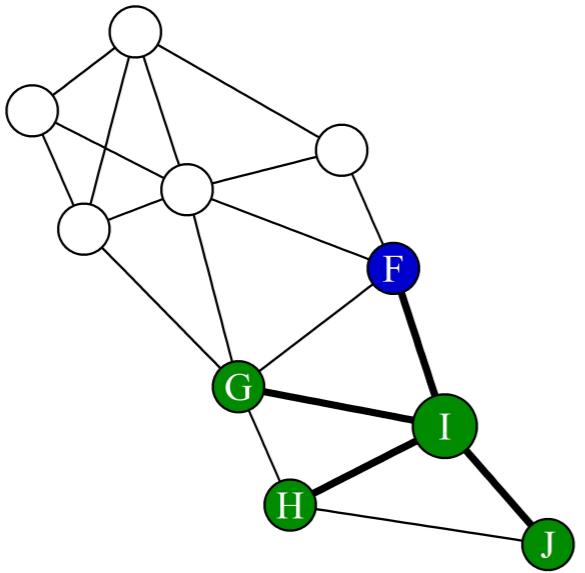
PREDICTIVE ANALYTICS USING NETWORKED DATA IN R



Bart Baesens, Ph.D.

Professor of Data Science, KU Leuven
and University of Southampton

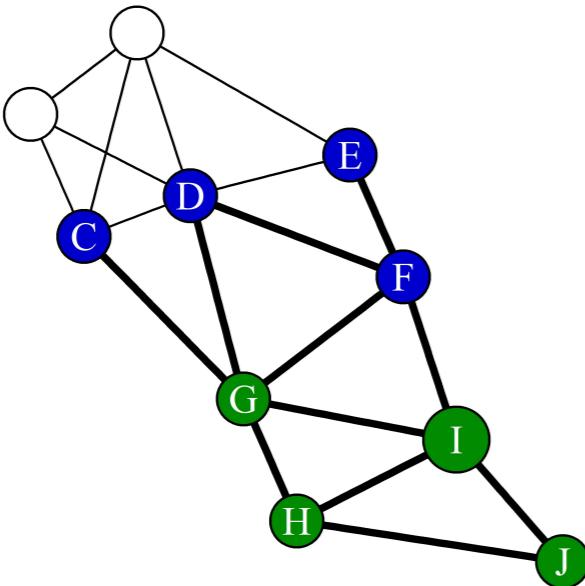
Neighborhood features



- First order degree
 - Number of connected nodes

```
degree(g)
```

A	B	C	D	E	F	G	H	I	J
4	3	4	6	3	4	5	3	4	2

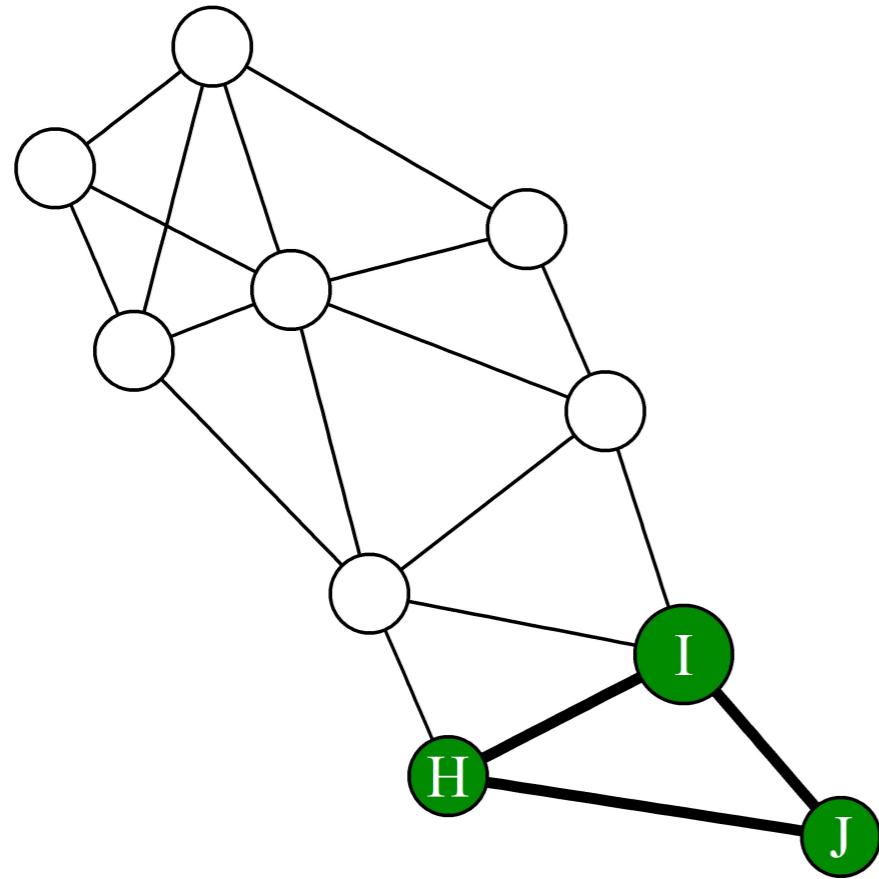


- Second order degree
 - Number of connected nodes that are two or less edges away

```
neighborhood.size(g, order = 2)
```

7	7	9	9	8	10	10	7	8	5
---	---	---	---	---	----	----	---	---	---

Neighborhood features - triangles

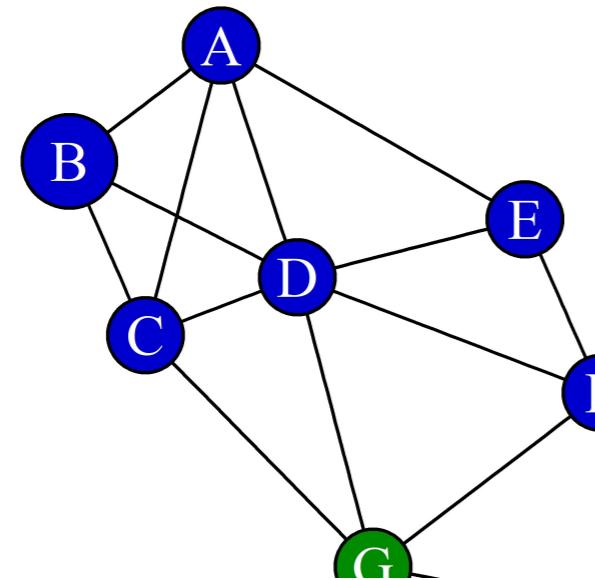


```
count_triangles(g)
```

```
4 3 4 7 2 3 4 2 3 1
```

Centrality features

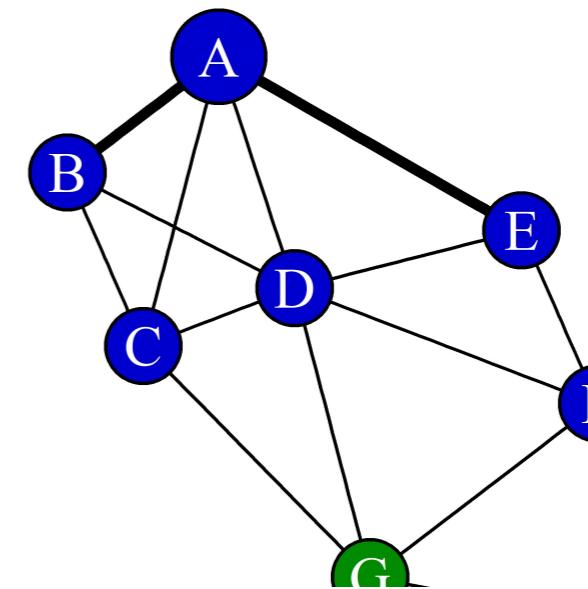
- Betweenness



`betweenness(g)`

A	B	C	D	E	F	G	H	I	J
1.00	0.00	3.32	8.10	0.92	5.37	11.47	2.07	5.77	0.00

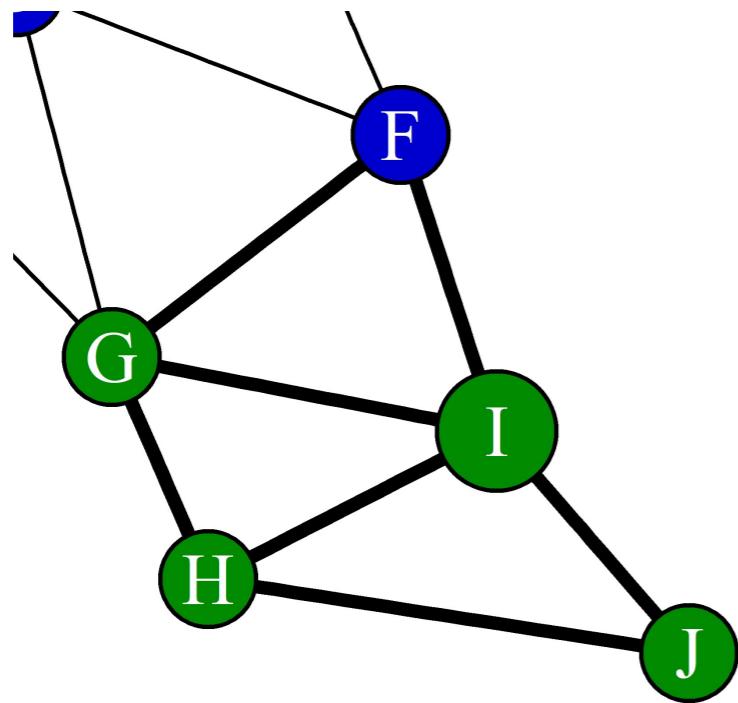
- Closeness



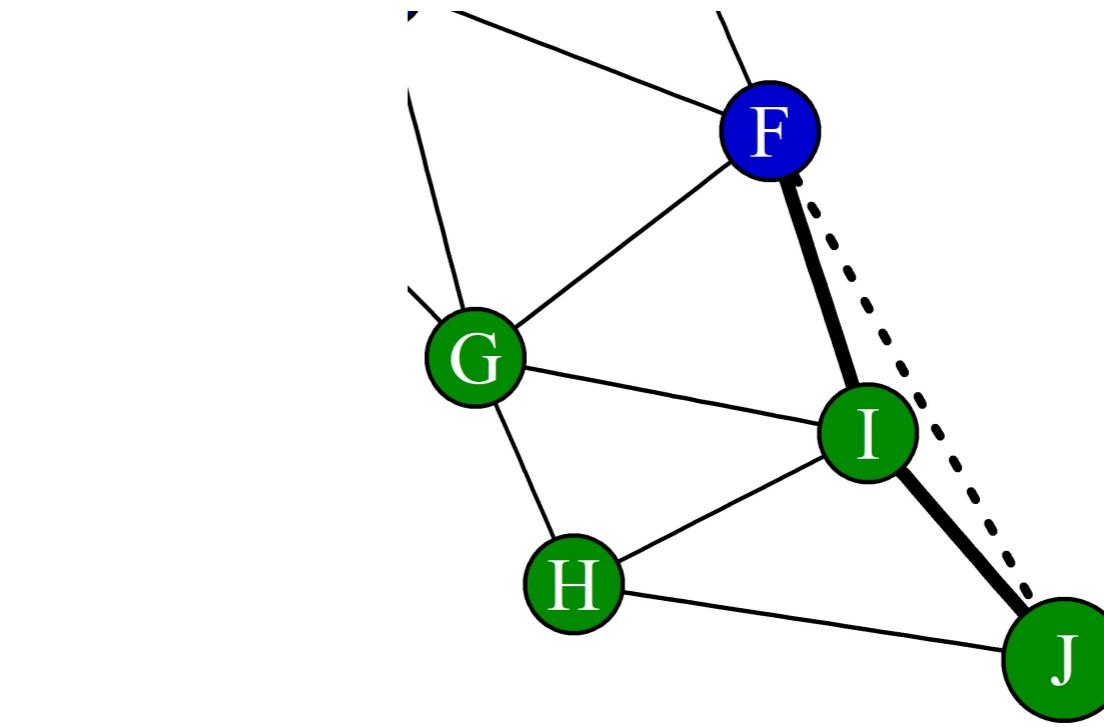
`closeness(g)`

A	B	C	D	E	F	G	H	I	J
0.06	0.05	0.07	0.08	0.06	0.07	0.08	0.06	0.06	0.04

Transitivity



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



```
0.67 1.00 0.67 0.47 0.67 0.50 0.40 0.67 0.50 1.00
```

Let's practice!

PREDICTIVE ANALYTICS USING NETWORKED DATA IN R

Link Based Features

PREDICTIVE ANALYTICS USING NETWORKED DATA IN R

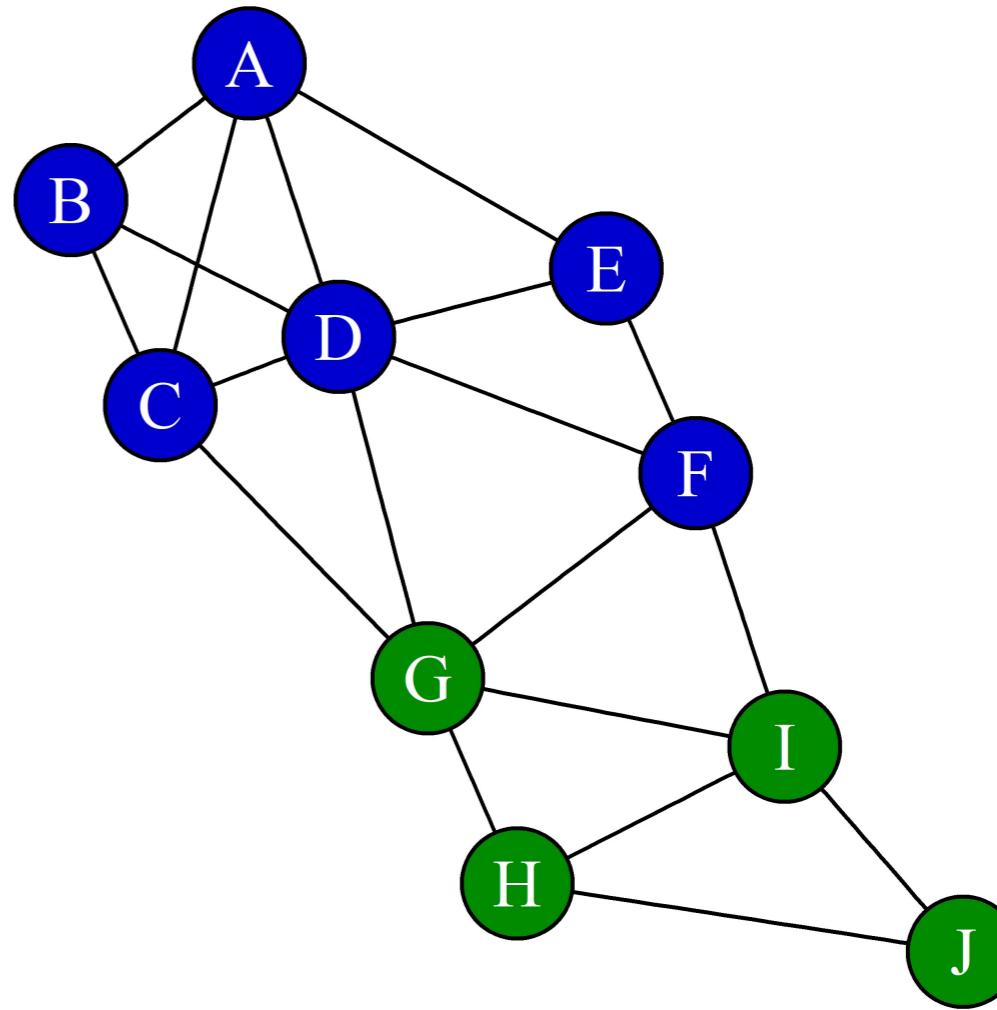


María Óskarsdóttir, Ph.D.

Post-doctoral researcher

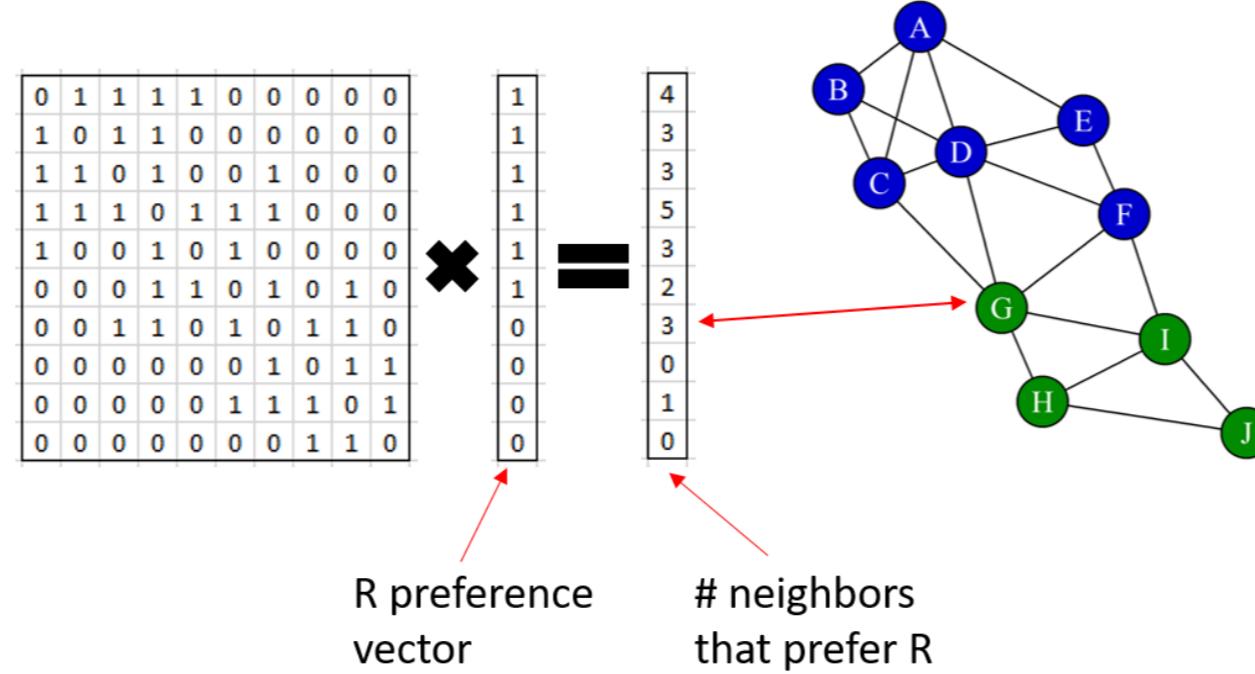
Adjacency matrices

	A	B	C	D	E	F	G	H	I	J
A	0	1	1	1	1	0	0	0	0	0
B	1	0	1	1	0	0	0	0	0	0
C	1	1	0	1	0	0	1	0	0	0
D	1	1	1	0	1	1	1	0	0	0
E	1	0	0	1	0	1	0	0	0	0
F	0	0	0	1	1	0	1	0	1	0
G	0	0	1	1	0	1	0	1	1	0
H	0	0	0	0	0	0	1	0	1	1
I	0	0	0	0	1	1	1	0	1	0
J	0	0	0	0	0	0	0	1	1	0



```
A <- get.adjacency(g)
```

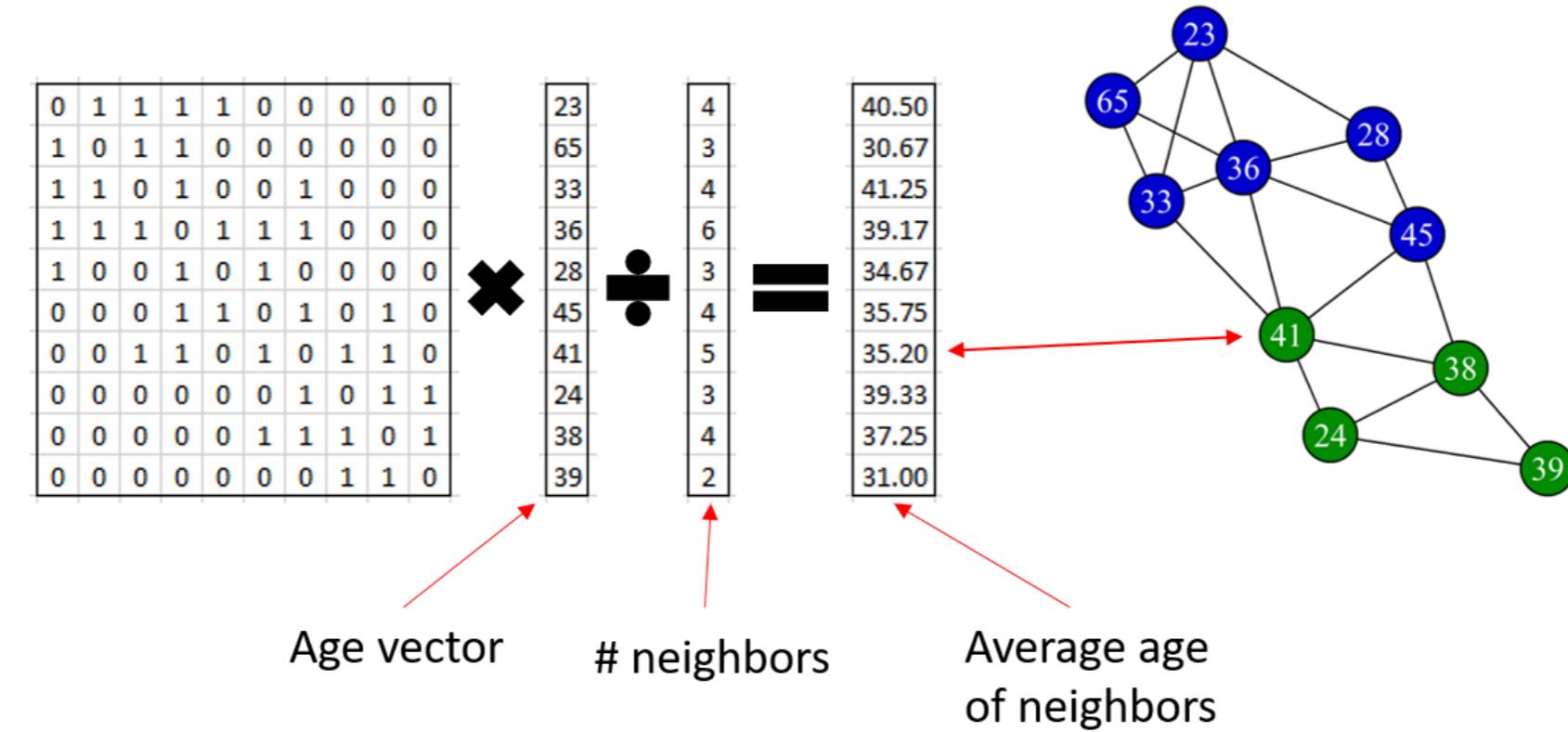
Link based features



```
preference <- c(1,1,1,1,1,1,0,0,0,0)
rNeighbors <- A %*% preference
as.vector(rNeighbors)
```

4 3 3 5 3 2 3 0 1 0

Neighborhood features



```
age <- c(23, 65, 33, 36, 28, 45, 41, 24, 38, 39)
degree <- degree(g)
averageAge <- A %*% age / degree
```

Let's practice!

PREDICTIVE ANALYTICS USING NETWORKED DATA IN R

PageRank

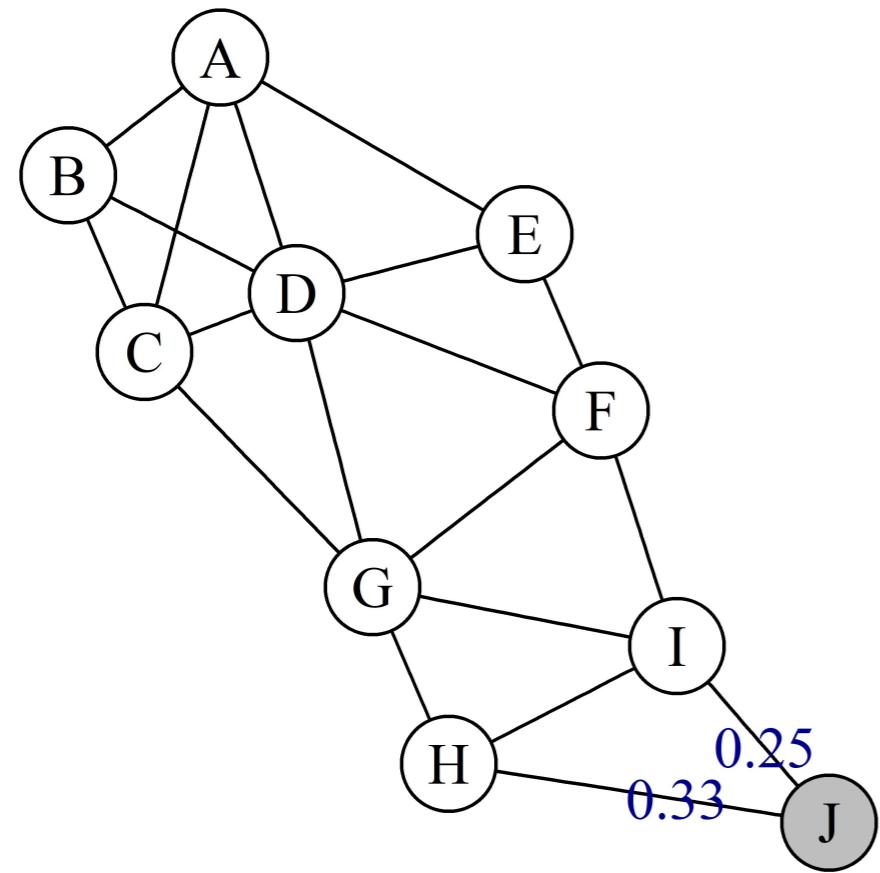
PREDICTIVE ANALYTICS USING NETWORKED DATA IN R



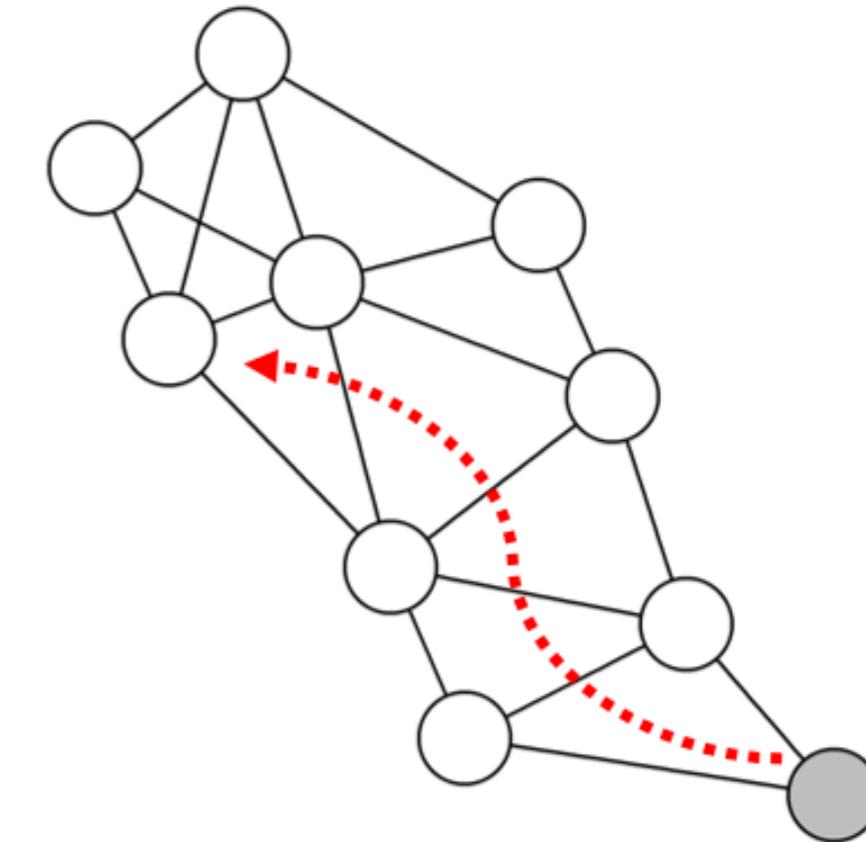
María Óskarsdóttir, Ph.D.

Post-doctoral researcher

The PageRank algorithm



$$\text{PageRank}_J = \alpha \cdot \left(\frac{1}{3} \cdot \text{PageRank}_H + \frac{1}{4} \cdot \text{PageRank}_I \right) + (1 - \alpha) \cdot e_J$$



$$+ \frac{1}{4} \cdot \text{PageRank}_I \Big) + (1 - \alpha) \cdot e_J$$

The PageRank algorithm

$$\vec{PR} = \alpha \cdot A \cdot \vec{PR} + (1 - \alpha) \cdot \vec{e}$$

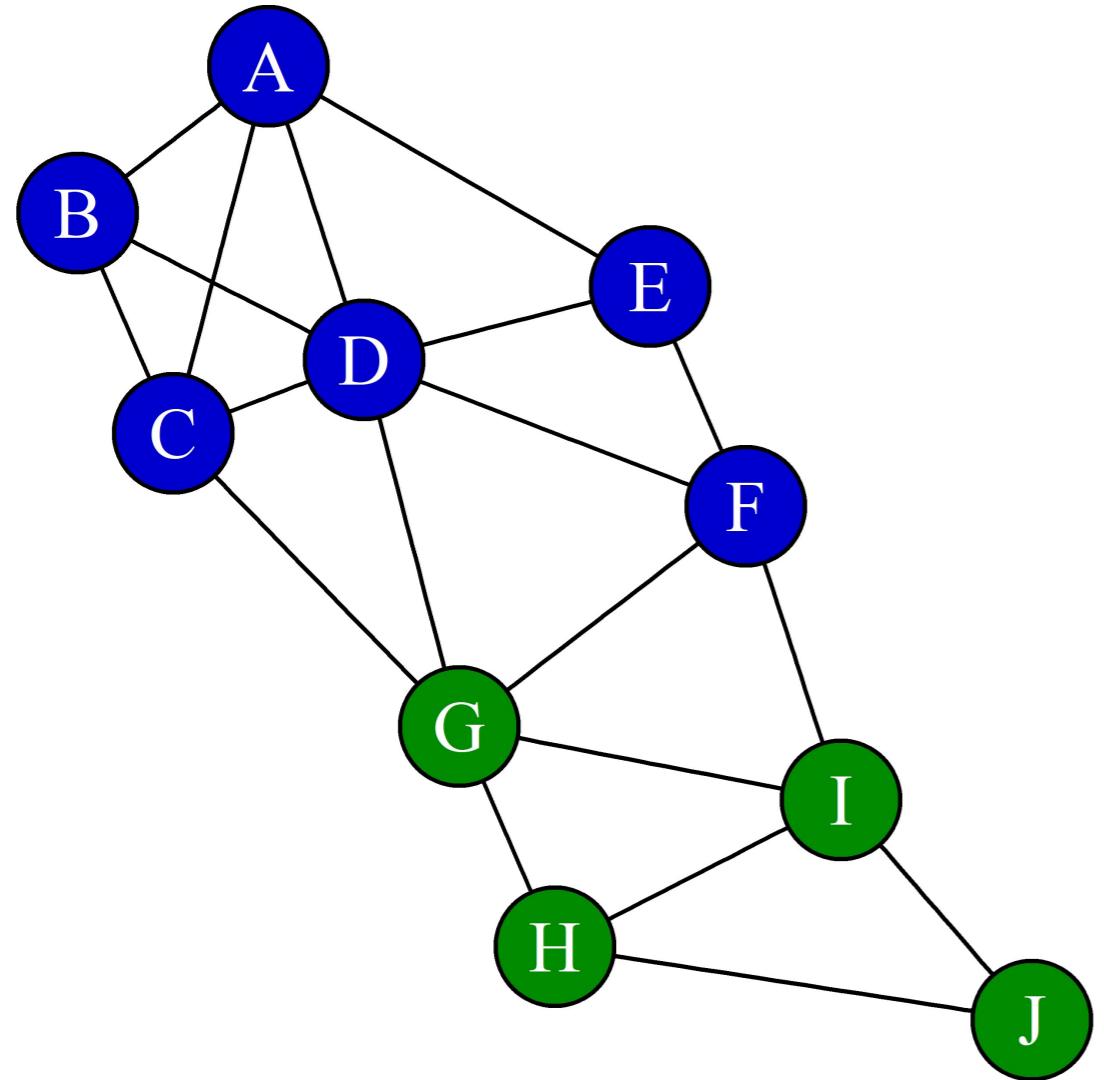
```
page.rank(g)
```

```
$vector
  A          B          C          D          E          F          G
0.10238312 0.07917232 0.10164910 0.14693274 0.07953551 0.10335821 0.12732387
  H          I          J
0.08675903 0.10994175 0.06294435
```

```
$value
[1] 1
```

```
$options
NULL
```

Personalized PageRank



```
page.rank(g,  
personalized = c(1,0,0,0,0,0,0,0,0,0))
```

```
$vector  
A          B          C          D          E  
0.25528911 0.10363533 0.12156935 0.16625582 0.09366836  
F          G          H          I          J  
0.07466596 0.08473039 0.03285162 0.04785657 0.01947748  
  
$value  
[1] 1  
  
$options  
NULL
```

Let's practice!

PREDICTIVE ANALYTICS USING NETWORKED DATA IN R

Extract a dataset

PREDICTIVE ANALYTICS USING NETWORKED DATA IN R



María Óskarsdóttir, Ph.D.

Post-doctoral researcher

```
V(g)$degree<-degree(g)
V(g)$triangles<-count_triangles(g)
V(g)$betweeness<-betweeness(g,normalized=TRUE)
V(g)$transitivity<-transitivity(g,type='local',isolates='zero')
A <- get.adjacency(g)
preference <- c(1,1,1,1,1,0,0,0,0)
age <- c(23,65,33,36,28,45,41,24,38,39)
V(g)$rNeighbors <- as.vector(A%*%preference)
V(g)$averageAge <- as.vector(A%*%age/V(g)$degree)
V(g)$pageRank<-page.rank(g)$vector
V(g)$personalizePageRank<-page.rank(g,
  personalized = c(1,0,0,0,0,0,0,0,0))$vector
g
```

```
IGRAPH UN-- 10 19 --
attr: name (v/c), degree (v/n), triangles (v/n), transitivity
| (v/n), rNeighbors (v/n), averageAge (v/n), pageRank (v/n),
| pPageRank (v/n), label (e/c)
edges (vertex names):
A--B A--C A--D A--E B--C B--D C--D C--G D--E D--F D--G E--F F--G F--I G--I G--H H--I H--J I--J
```

```
IGRAPH UN-- 10 19 --
attr: name (v/c), degree (v/n), triangles (v/n), transitivity
| (v/n), rNeighbors (v/n), averageAge (v/n), pageRank (v/n),
| pPageRank (v/n), label (e/c)
edges (vertex names):
[1] A--B A--C A--D A--E B--C B--D C--D C--G D--E D--F D--G E--F F--G F--I G--I G--H H--I H--J I--J
```

```
as_data_frame(g,what='vertices')
```

		name	degree	triangles	transitivity	rNeighbors	averageAge	pageRank	pPageRank
A	A	A	4	4	0.6666667	4	40.50000	0.10238312	0.25528911
B	B	B	3	3	1.0000000	3	30.66667	0.07917232	0.10363533
C	C	C	4	4	0.6666667	3	41.25000	0.10164910	0.12156935
D	D	D	6	7	0.4666667	5	39.16667	0.14693274	0.16625582
E	E	E	3	2	0.6666667	3	34.66667	0.07953551	0.09366836
F	F	F	4	3	0.5000000	2	35.75000	0.10335821	0.07466596
G	G	G	5	4	0.4000000	3	35.20000	0.12732387	0.08473039
H	H	H	3	2	0.6666667	0	39.33333	0.08675903	0.03285162
I	I	I	4	3	0.5000000	1	37.25000	0.10994175	0.04785657
J	J	J	2	1	1.0000000	0	31.00000	0.06294435	0.01947748

Preprocessing - missing values

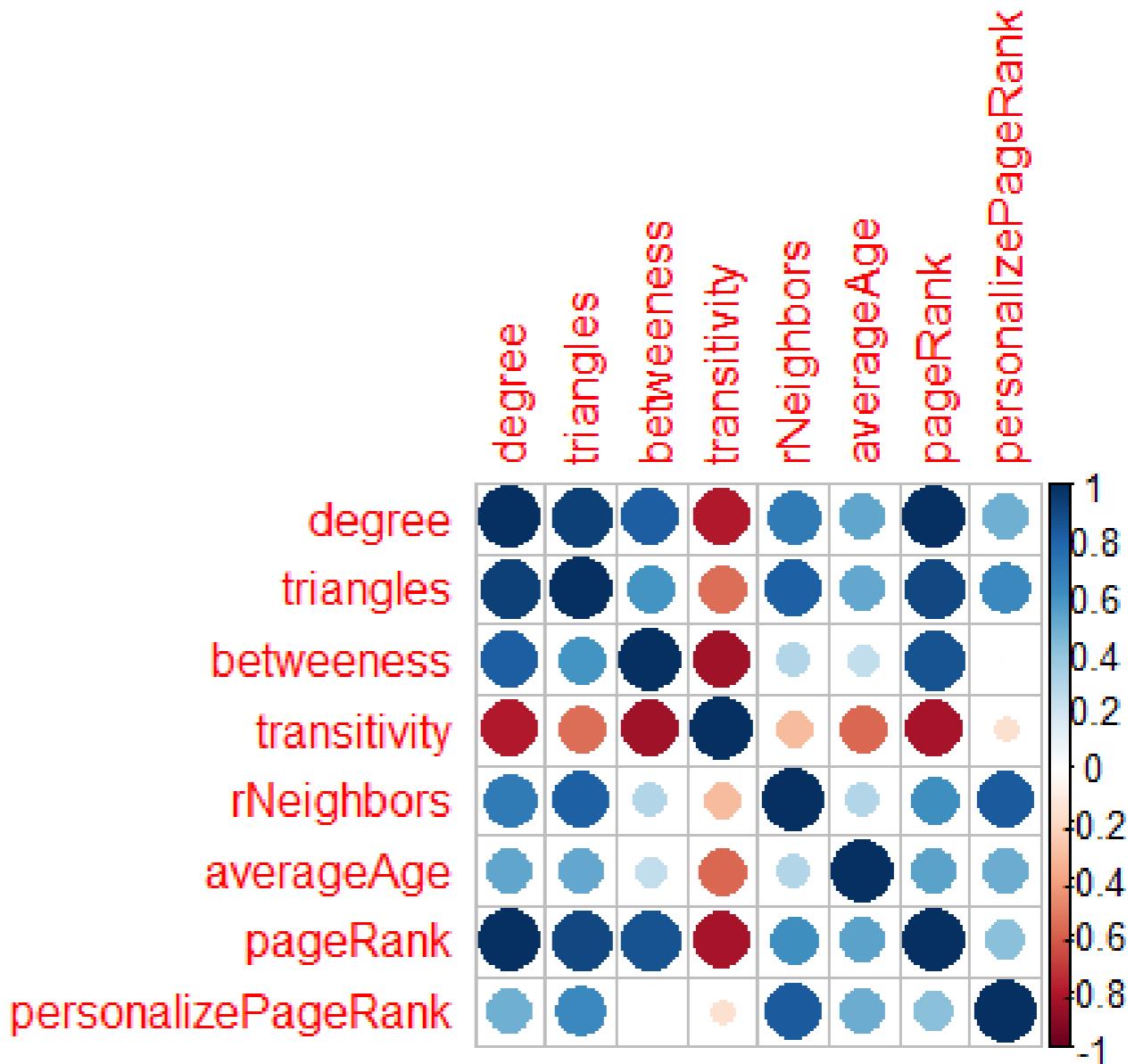
name	degree	triangles	transitivity	rNeighbors	averageAge	pageRank	pPageRank
A	4	4	0.6666667	4	40.50000	0.10238312	0.25528911
B	3	3	1.0000000	3	30.66667	0.07917232	0.10363533
C	NA	4	0.6666667	3	41.25000	0.10164910	0.12156935
D	6	7	0.4666667	5	39.16667	0.14693274	0.16625582
E	3	2	0.6666667	3	34.66667	0.07953551	0.09366836
F	4	3	0.5000000	2	35.75000	0.10335821	0.07466596
G	5	4	0.4000000	3	35.20000	0.12732387	0.08473039
H	2	2	0.6666667	0	39.33333	0.08675903	0.03285162
I	NA	3	0.5000000	1	37.25000	0.10994175	0.04785657
J	2	1	1.0000000	0	31.00000	0.06294435	0.01947748

```
sum(is.na(dataset$degree))
```

2

Preprocessing - correlated variables

```
library(corrplot)  
  
M <- cor(dataset[,-1])  
  
corrplot(M, method = 'circle')
```



Let's practice!

PREDICTIVE ANALYTICS USING NETWORKED DATA IN R

Building a predictive model

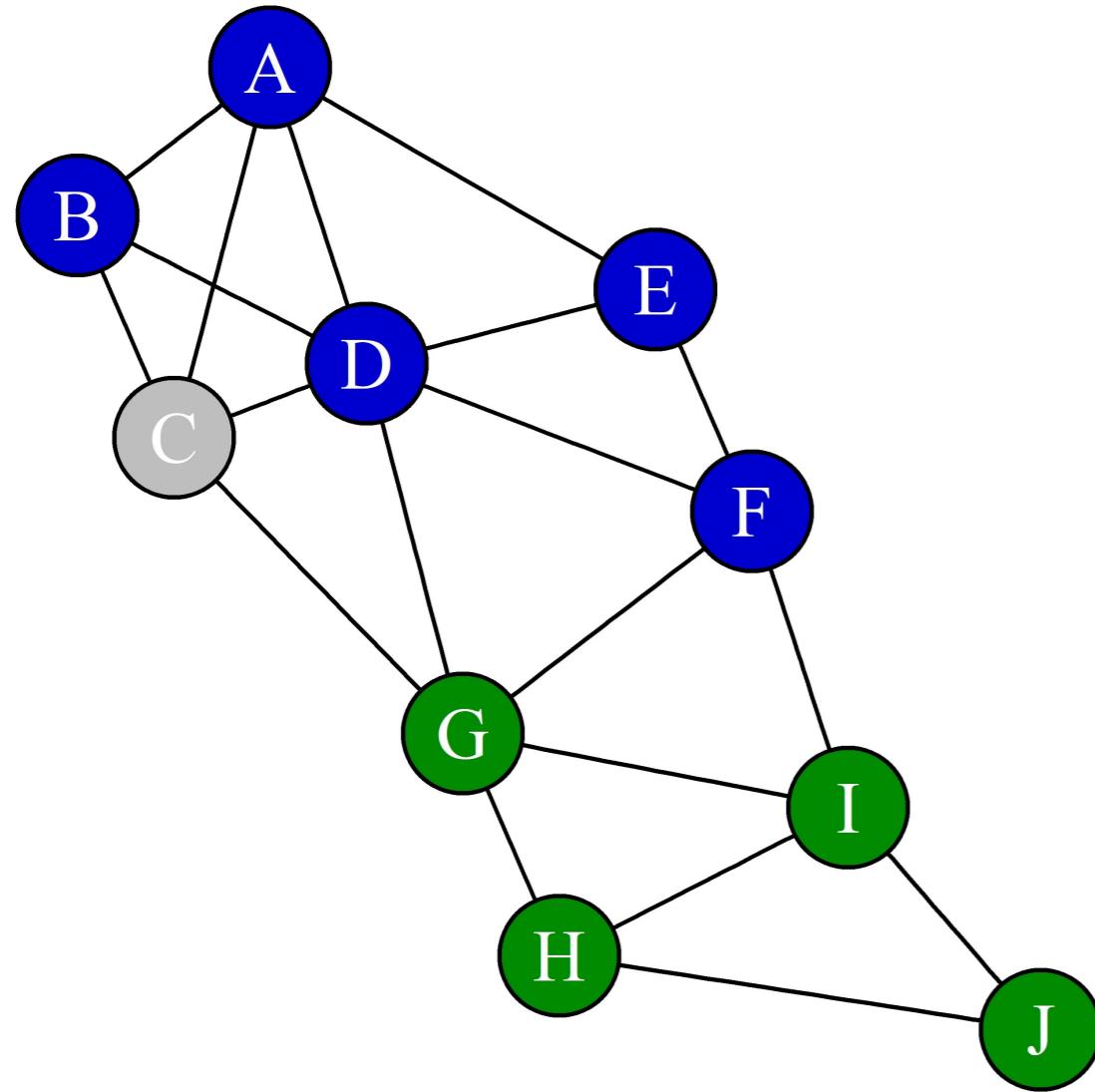
PREDICTIVE ANALYTICS USING NETWORKED DATA IN R



María Óskarsdóttir, Ph.D.

Post-doctoral researcher

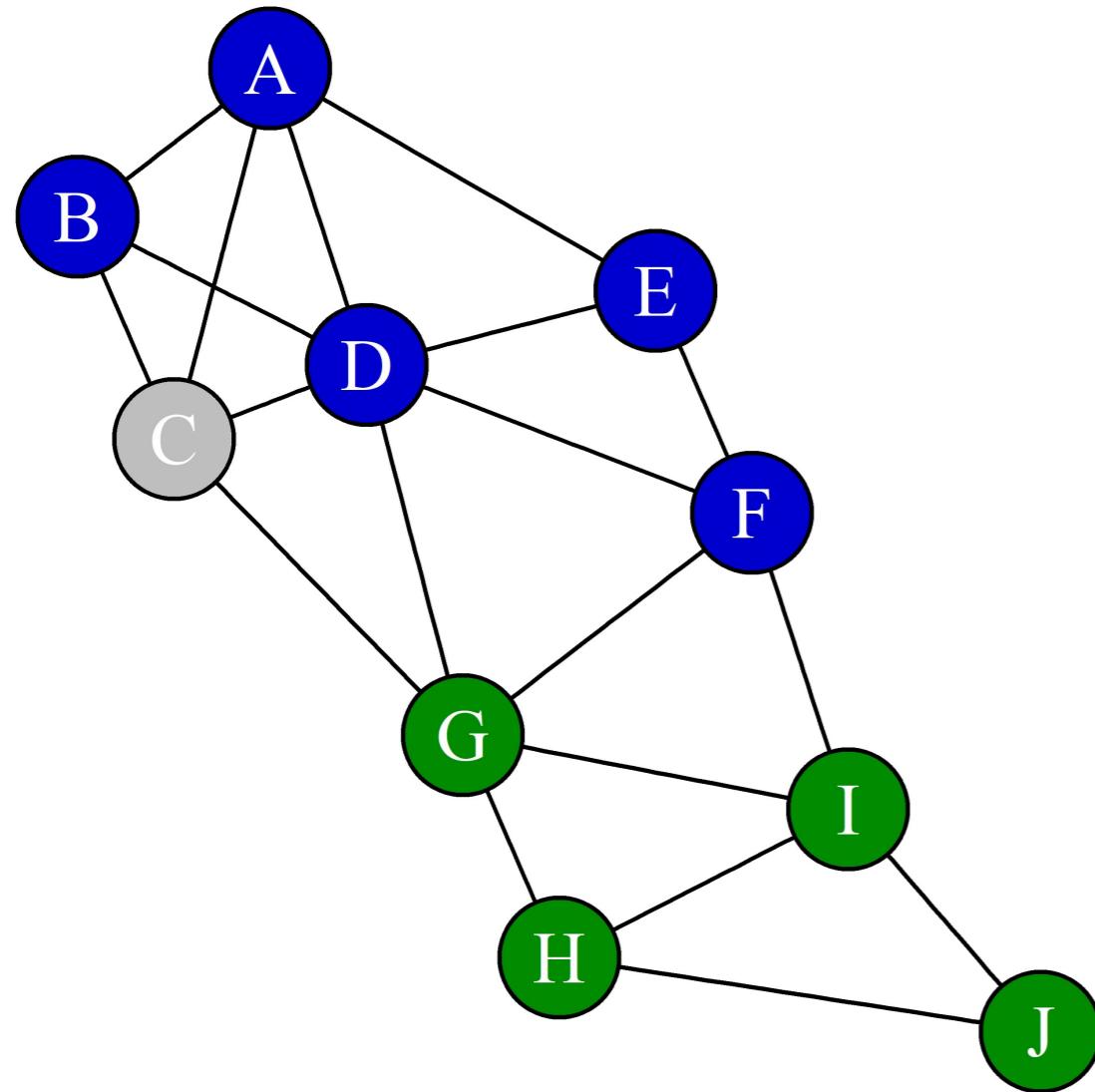
Predictive modeling



```
dataset$preference<-c(rep('R',2), '?',  
rep('R',3),rep('P',4))  
dataset[,c(1,9)]
```

	name	preference
A	A	R
B	B	R
C	C	?
D	D	R
E	E	R
F	F	R
G	G	P
H	H	P
I	I	P
J	J	P

Predictive modeling

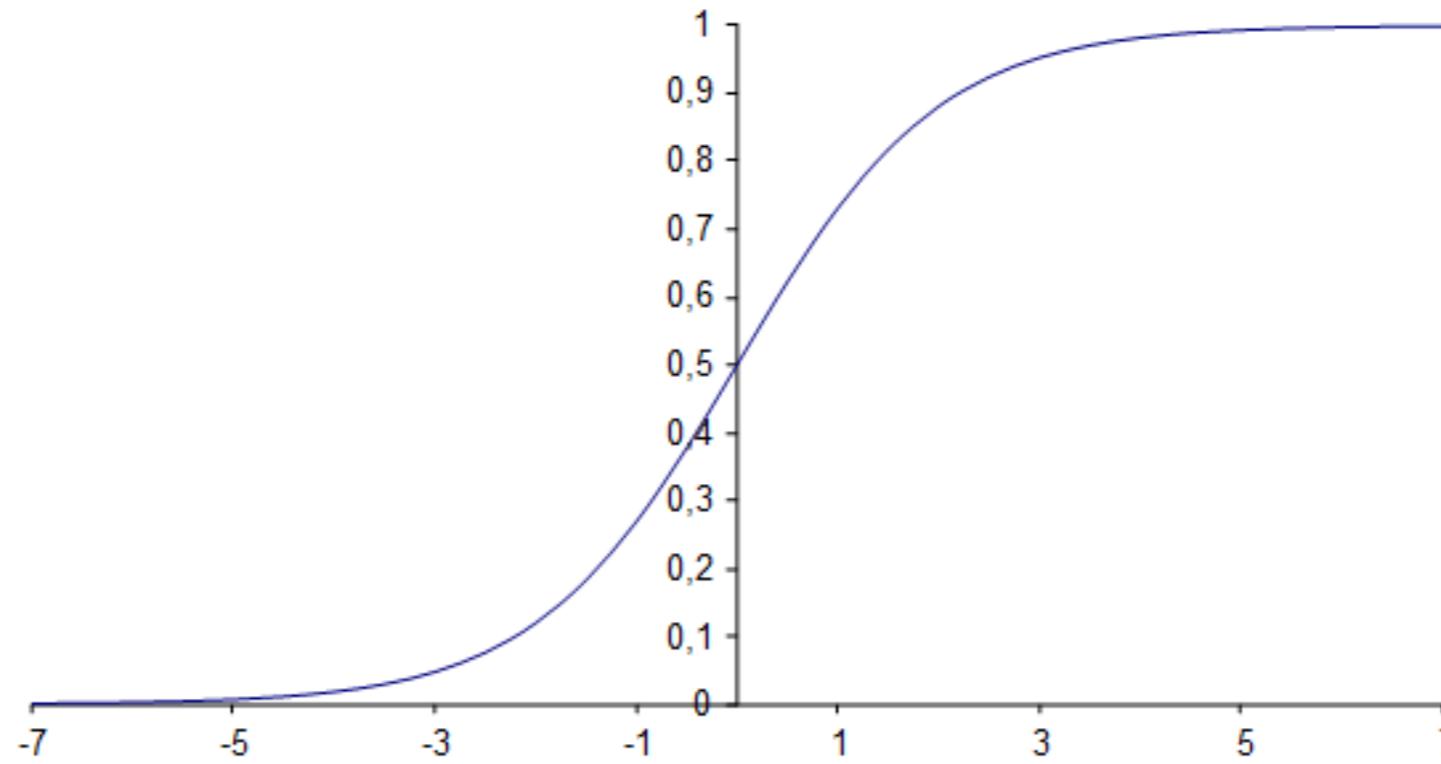


```
dataset$R<-c(1,1,'?',1,1,1,0,0,0,0)  
dataset[,c(1,9,10)]
```

	name	preference	R
A	A		R 1
B	B		R 1
C	C		? ?
D	D		R 1
E	E		R 1
F	F		R 1
G	G		P 0
H	H		P 0
I	I		P 0
J	J		P 0

```
training_set<-dataset[-3,-9]  
test_set<-dataset[3,-9]
```

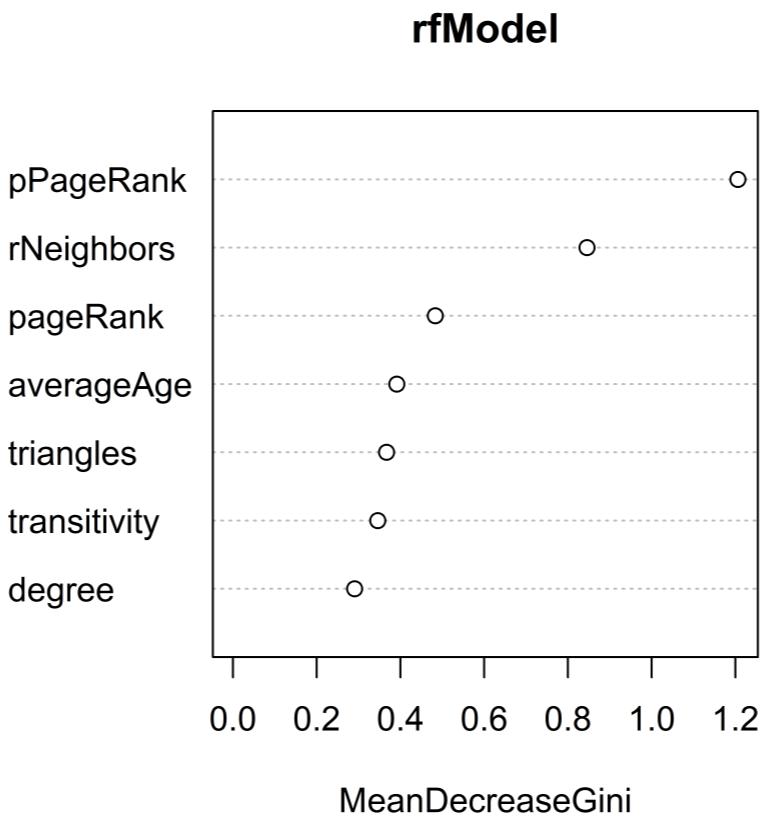
Logistic regression



```
glm(R~degree+pageRank, dataset=training_set,family='binomial')  
glm(R~., dataset=training_set,family='binomial')
```

Random forests

```
library(randomForest)  
rfModel<-randomForest(R~., dataset=training_set)  
varImpPlot(rfModel)
```



Let's practice!

PREDICTIVE ANALYTICS USING NETWORKED DATA IN R

Evaluating model performance

PREDICTIVE ANALYTICS USING NETWORKED DATA IN R



María Óskarsdóttir, Ph.D.

Post-doctoral researcher

Making predictions

```
library(pROC)
```

- Logistic regression

```
logPredictions <- predict(logModel, newdata = test_set, type = "response")
```

- Random forest

```
rfPredictions<- predict(rfModel, newdata = test_set, type='prob')
```

```
rfPredictions
```

```
attr(,"class")
```

```
0      1  
C 0.136 0.864  
"matrix" "votes"
```

AUC

- Probability that a randomly chosen churner gets a higher score than a randomly chosen non-churner
- Displays the trade-off between the model's sensitivity and specificity
- A number between:
 - **0.5**: random model
 - **1**: perfect model

```
library(pROC)  
auc(test_set$label, logPredictions)
```

Top decile lift

- How much better is the prediction model at identifying churners, compared to a random sample of customers
- Computes the proportion of actual churners amongst the 10% of customers with the highest predicted churn probability
- Lift value greater than 1 means that the model is better than a random model
- If, in the top 10% of the highest scores there are **60%** churners and in the whole population there are **10%** churners, then the lift is $60/10 = 6$

```
library(lift)  
TopDecileLift(test_set$label, predictions, plot=TRUE)
```

Let's practice!

PREDICTIVE ANALYTICS USING NETWORKED DATA IN R

Summary and final thoughts

PREDICTIVE ANALYTICS USING NETWORKED DATA IN R



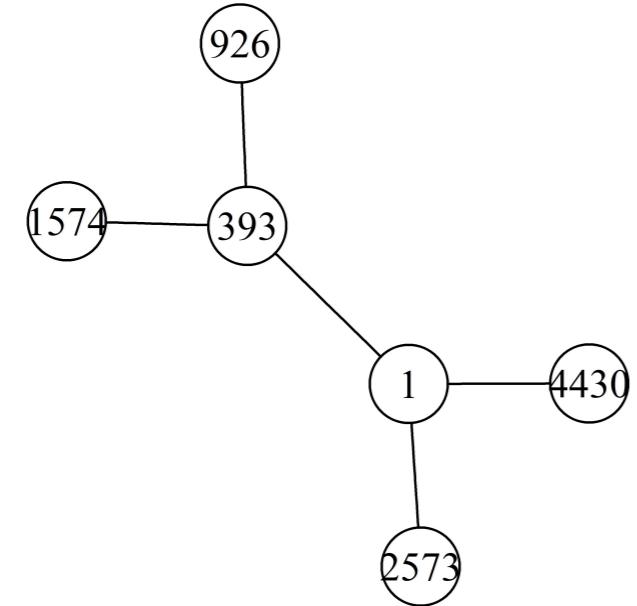
Bart Baesens, Ph.D.

Professor of Data Science, KU Leuven
and University of Southampton

Labeled networks

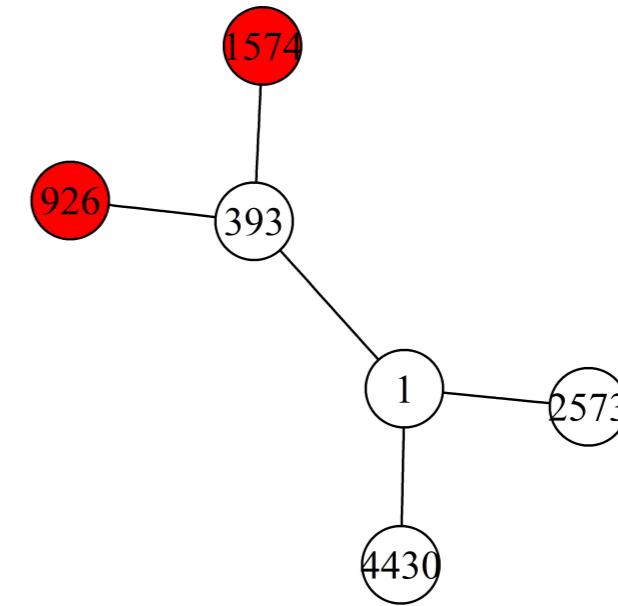
edgeList

```
from      to
1       1    393
2       1   2573
3       1   4430
4     393    926
5     393   1574
```

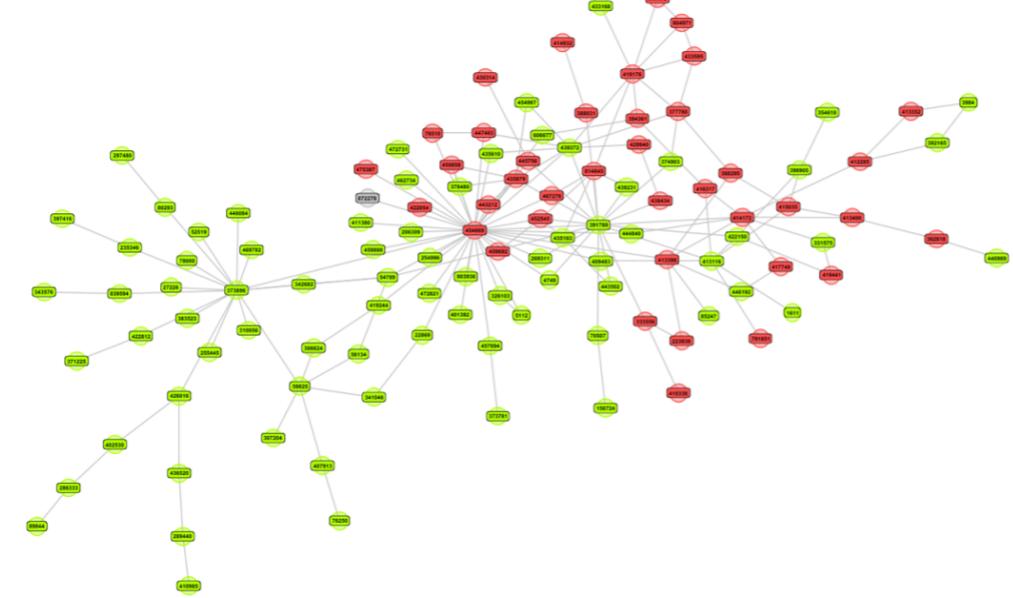


customers

```
id  churn
1   0
2   0
3   0
4   0
5   1
6   1
```



Homophily



Dyadicity: connectedness between nodes with same label

Birds of a feather flock together

Heterophilycty: connectedness between nodes with opposite labels

Network Featurization

```
g
IGRAPH UN-- 10 19 --
attr: name (v/c), label (e/c)
edges (vertex names):
A--B A--C A--D A--E B--C B--D C--D C--G D--E D--F D--G E--F F--G F--I G--I G--H H--I H--J I--J
```

```
V(g)$degree<-degree(g)
```

```
g
IGRAPH UN-- 10 19 --
attr: name (v/c), degree (v/n), triangles (v/n), transitivity
| (v/n), rNeighbors (v/n), averageAge (v/n), pageRank (v/n),
| pPageRank (v/n), label (e/c)
edges (vertex names):
A--B A--C A--D A--E B--C B--D C--D C--G D--E D--F D--G E--F F--G F--I G--I G--H H--I H--J I--J
```

1. Extract dataframe:

```
dataset <- as_data_frame(g, what='vertices')
```

2. Preprocess data set:

- Missing values, outliers, correlated variables, and normalization

3. Build model:

```
glm(R~., dataset=training_set, family='binomial')
```

4. Make predictions:

```
logPredictions <- predict(logModel, newdata=test_set, type="response")
```

5. Measure performance:

```
auc(test_set$label, logPredictions)  
TopDecileLift(test_set$label, predictions, plot=TRUE)
```

Congratulations!

PREDICTIVE ANALYTICS USING NETWORKED DATA IN R

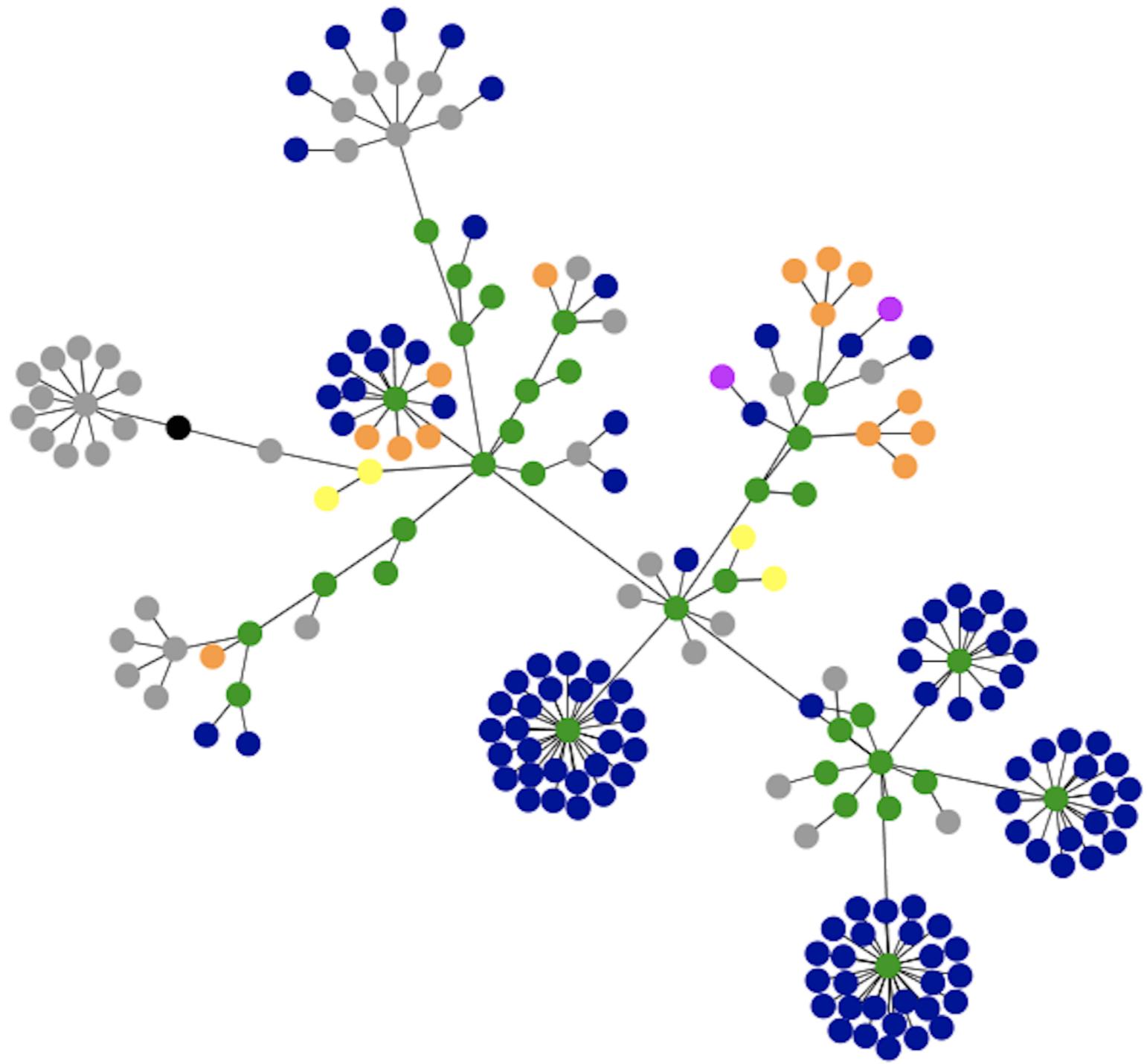
Network analysis in R: A tidy approach

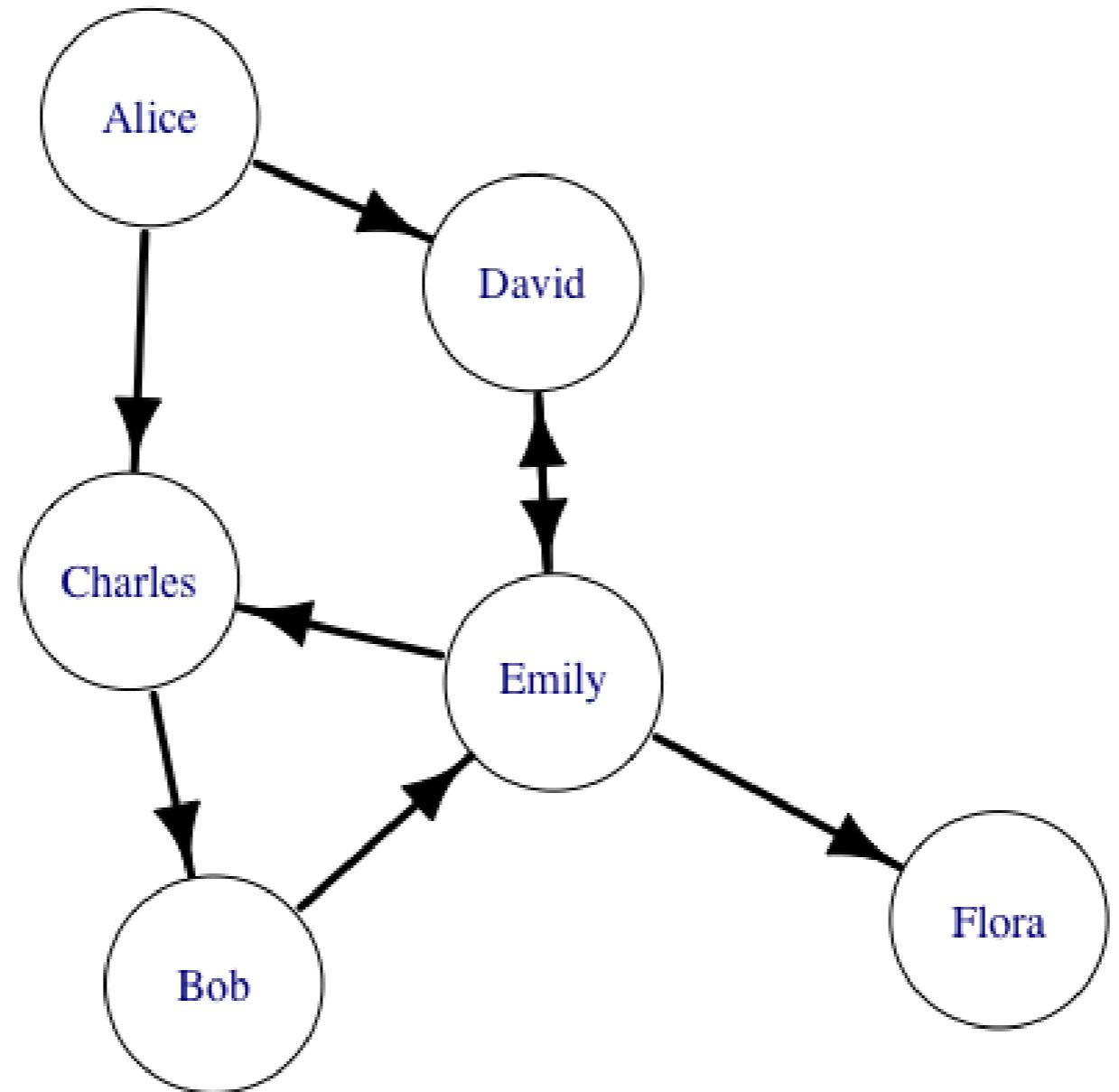
NETWORK ANALYSIS IN THE TIDYVERSE

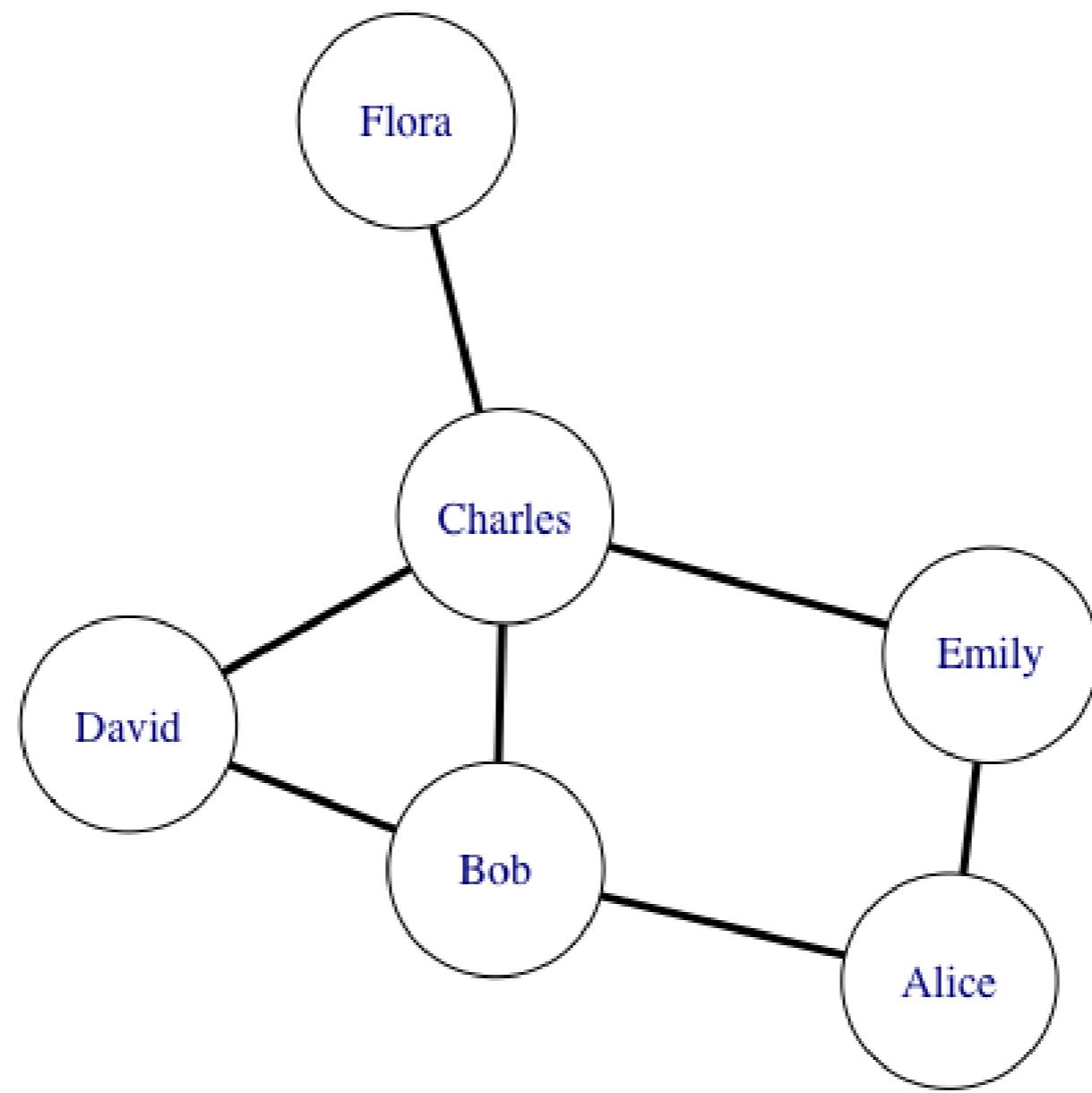


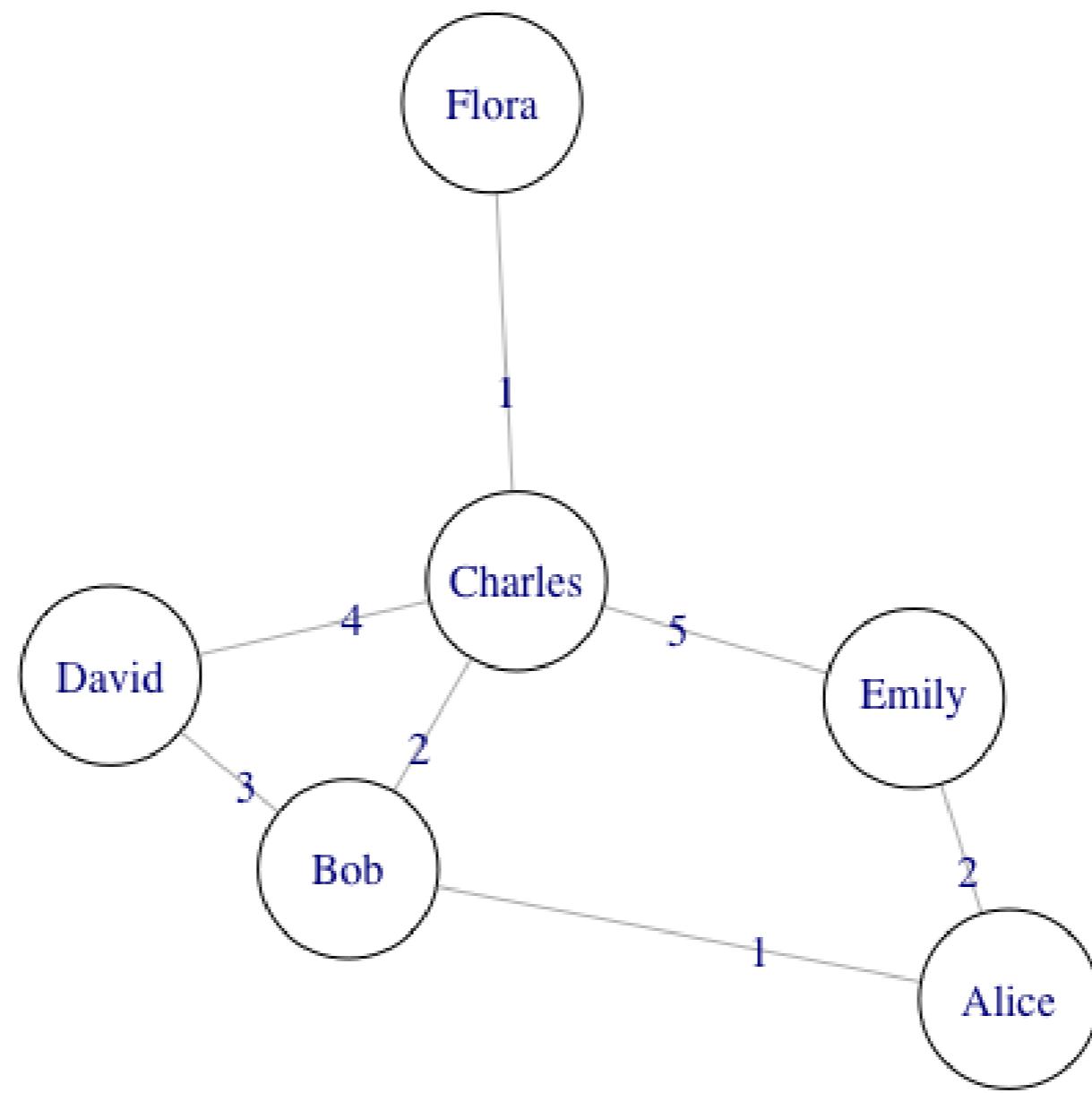
Massimo Franceschet

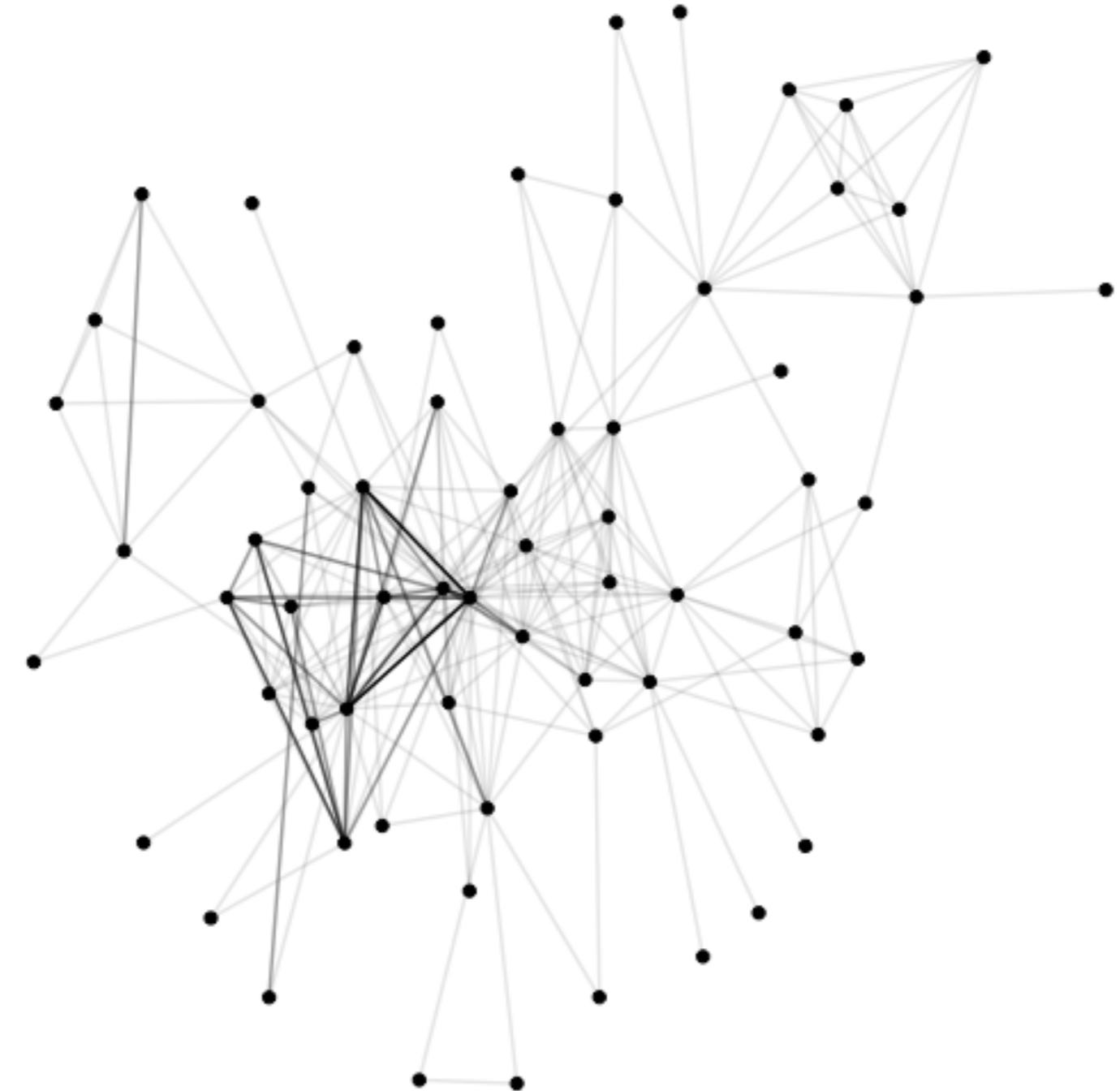
Prof. of Data Science, University of
Udine (Italy)











Building the network

```
# Load packages for network exploration
library(readr)
library(igraph)

# read nodes and ties data into variables
nodes <- read_csv("nodes.csv")
ties <- read_csv("ties.csv")

# build a network from data frames
g <- graph_from_data_frame(d = ties,
                           directed = FALSE,
                           vertices = nodes)
```

Exploring the network

```
# explore the set of nodes and print the number of nodes  
V(g)  
vcount(g)  
  
# explore the set of ties and print the number of ties  
E(g)  
ecount(g)  
# add the name attribute "Madrid network" to the network and print it  
g$name <- "Madrid network"  
g$name  
# add node attribute id and print the node `id` attribute  
V(g)$id <- 1:vcount(g)  
  
# print the tie `weight` attribute  
E(g)$weight
```

Let's start the investigation!

NETWORK ANALYSIS IN THE TIDYVERSE

Visualizing networks

NETWORK ANALYSIS IN THE TIDYVERSE



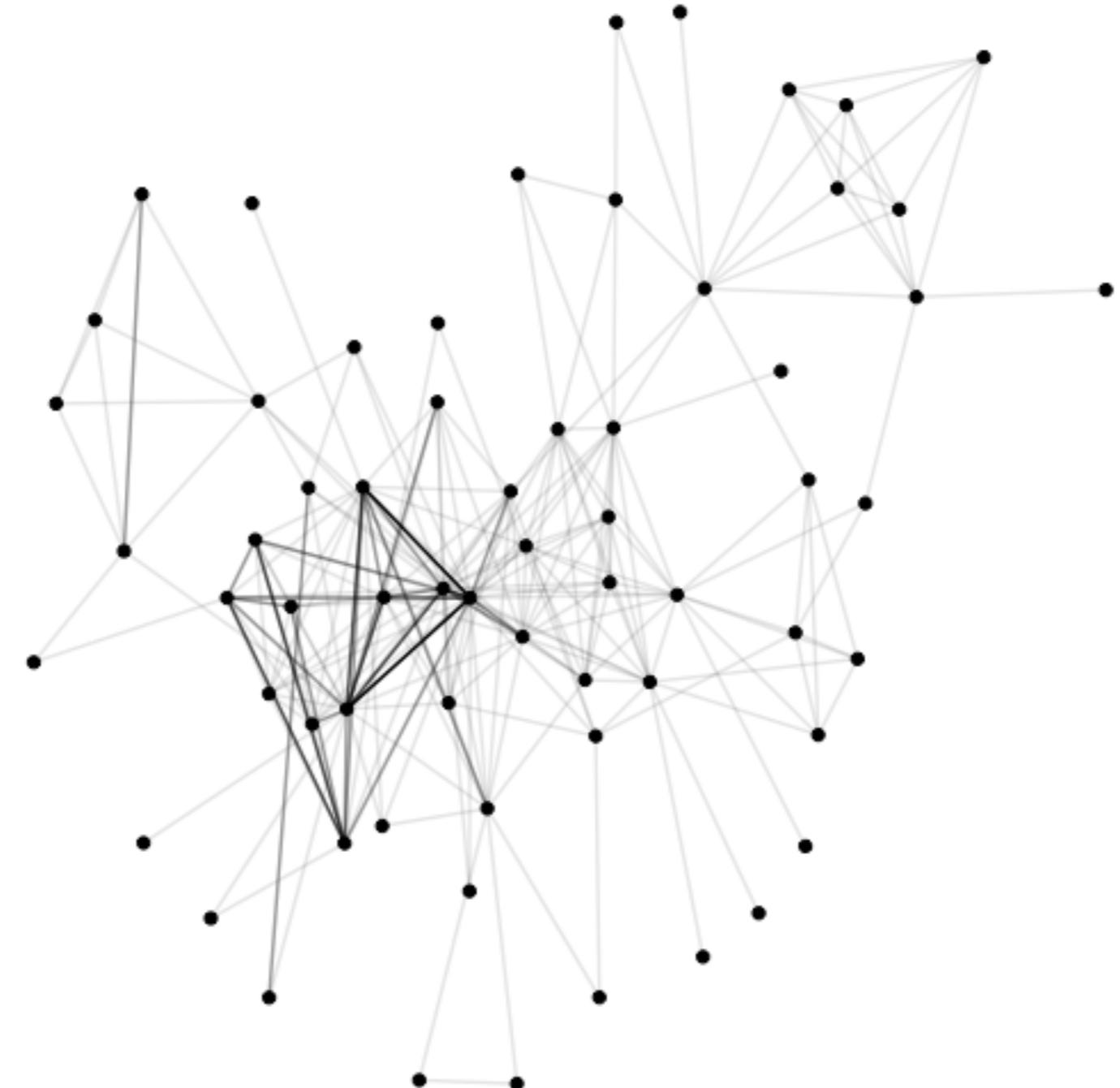
Massimo Franceschet

Prof. of Data Science, University of
Udine (Italy)

ggraph()

```
# Load packages for data manipulation and visualization
library(igraph)
library(dplyr)
library(ggplot2)
library(ggraph)
```

```
# visualize the network
ggraph(g, layout = "with_kk") +
  geom_edge_link(aes(alpha = weight)) +
  geom_node_point()
```



Let's practice!

NETWORK ANALYSIS IN THE TIDYVERSE

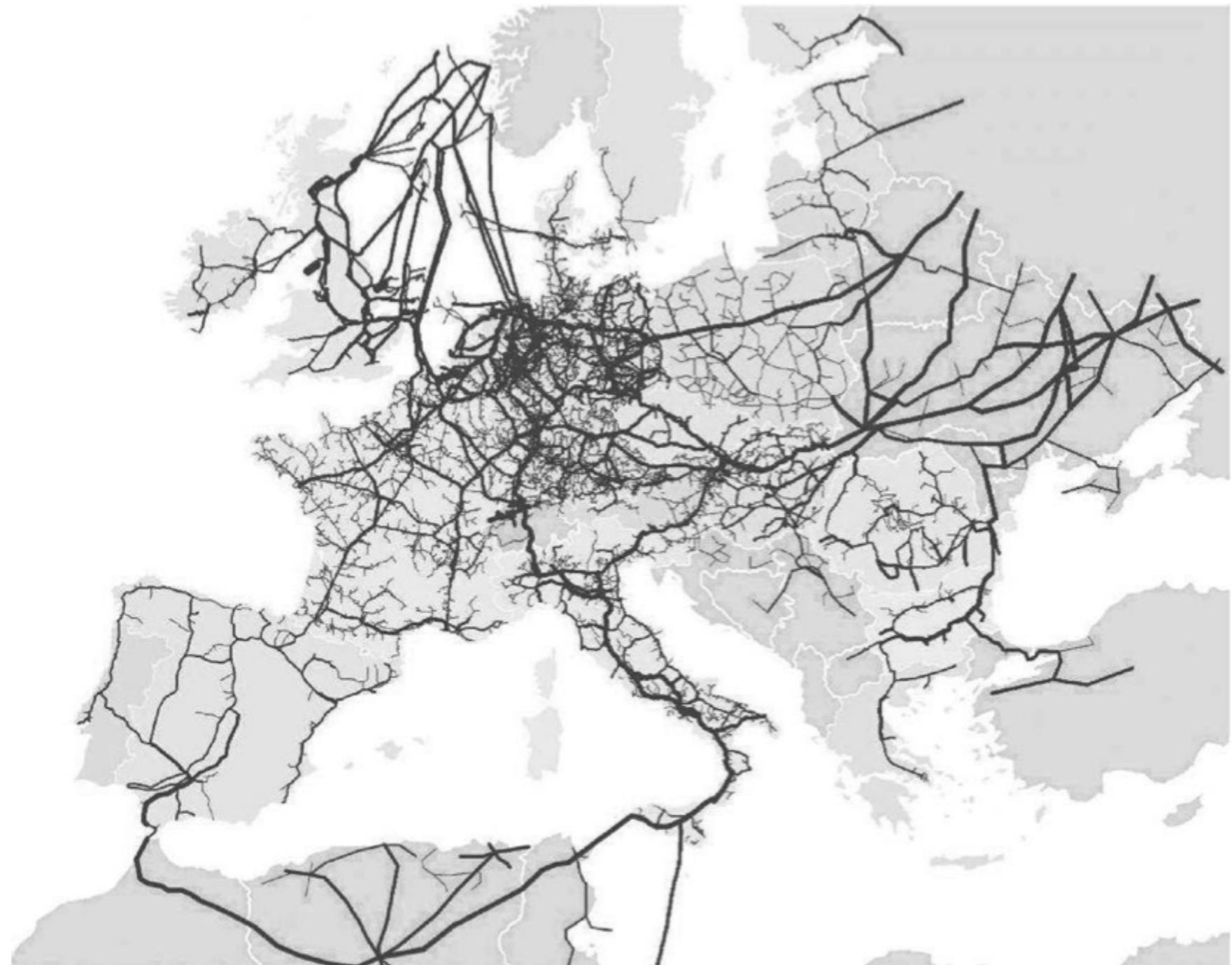
Centrality measures

NETWORK ANALYSIS IN THE TIDYVERSE



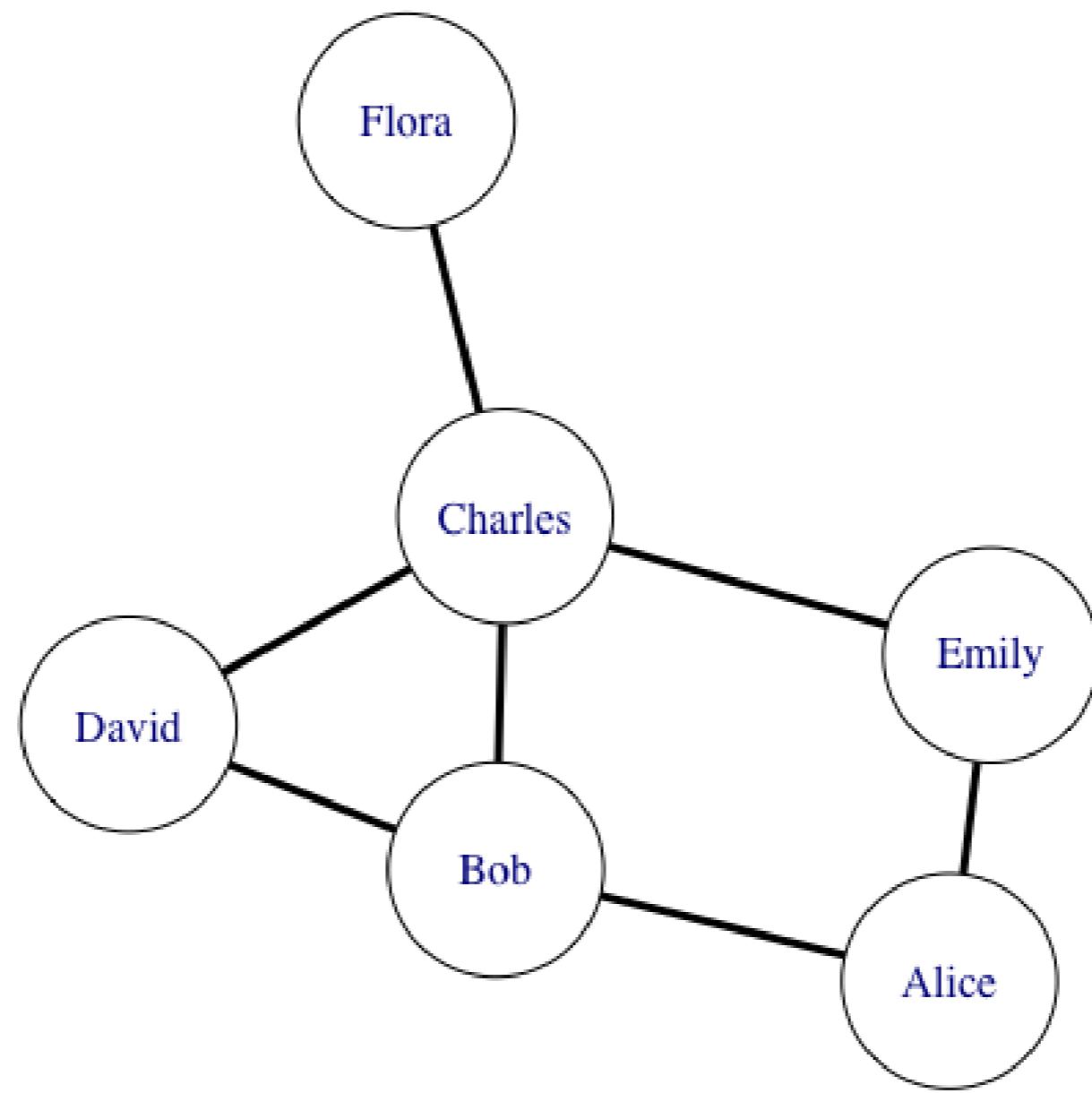
Massimo Franceschet

Prof. of Data Science, University of
Udine (Italy)



Node centrality

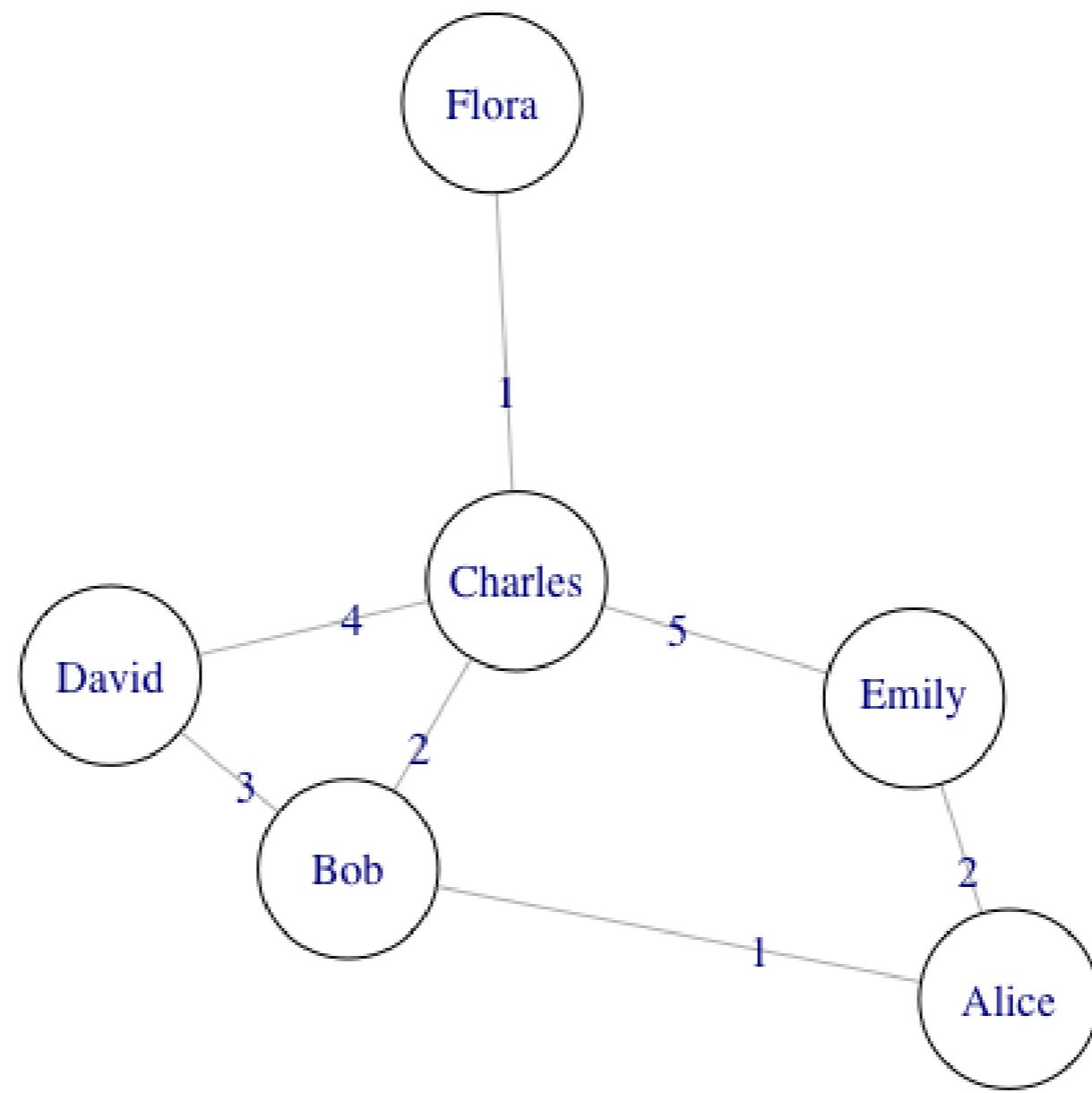
- Which are the most important nodes in a network?
 - Important web pages about a certain topic
 - Influential academic papers covering a given issue
 - Internet routers whose failure would greatly affect network connectivity



Computing degree

```
# compute node degrees  
degree(g)
```

Jamal Zougam	Mohamed Bekkali	Mohamed Chaoui
29	2	27
Vinay Kholy	Suresh Kumar	Mohamed Chedadi
10	10	7
Imad Eddin Barakat	Abdelaziz Benyaich	Abu Abderrahame
22	6	4
Omar Dhegaye	Amer Azizi	Abu Musad Alsakaoui
2	18	10
Mohamed Atta	Ramzi Binalshibh	Mohamed Belfatmi
10	10	11
Said Bahaji	Galeb Kalaje	Abderrahim Zbakh
11	16	15



Computing strength

```
# compute node strengths  
strength(g)
```

Jamal Zougam	Mohamed Bekkali	Mohamed Chaoui
43	2	34
Vinay Kholy	Suresh Kumar	Mohamed Chedadi
10	10	7
Imad Eddin Barakat	Abdelaziz Benyaich	Abu Abderrahame
35	7	4
Omar Dhegaye	Amer Azizi	Abu Musad Alsakaoui
3	27	10
Mohamed Atta	Ramzi Binalshibh	Mohamed Belfatmi
12	14	19
Said Bahaji	Galeb Kalaje	Abderrahim Zbakh
17	21	15

**Let's find the most
central terrorists in
the network!**

NETWORK ANALYSIS IN THE TIDYVERSE

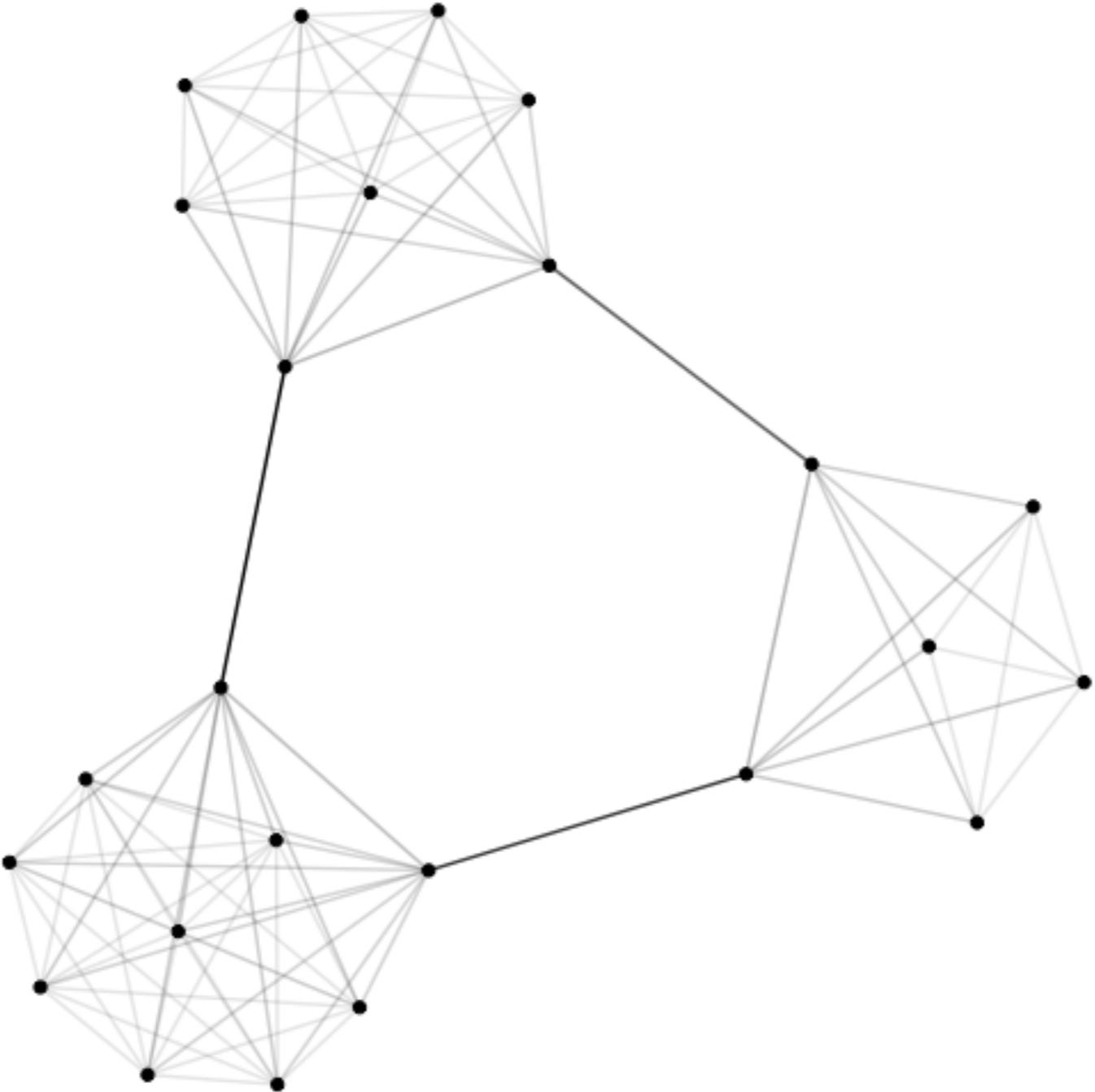
Betweenness on ties

NETWORK ANALYSIS IN THE TIDYVERSE

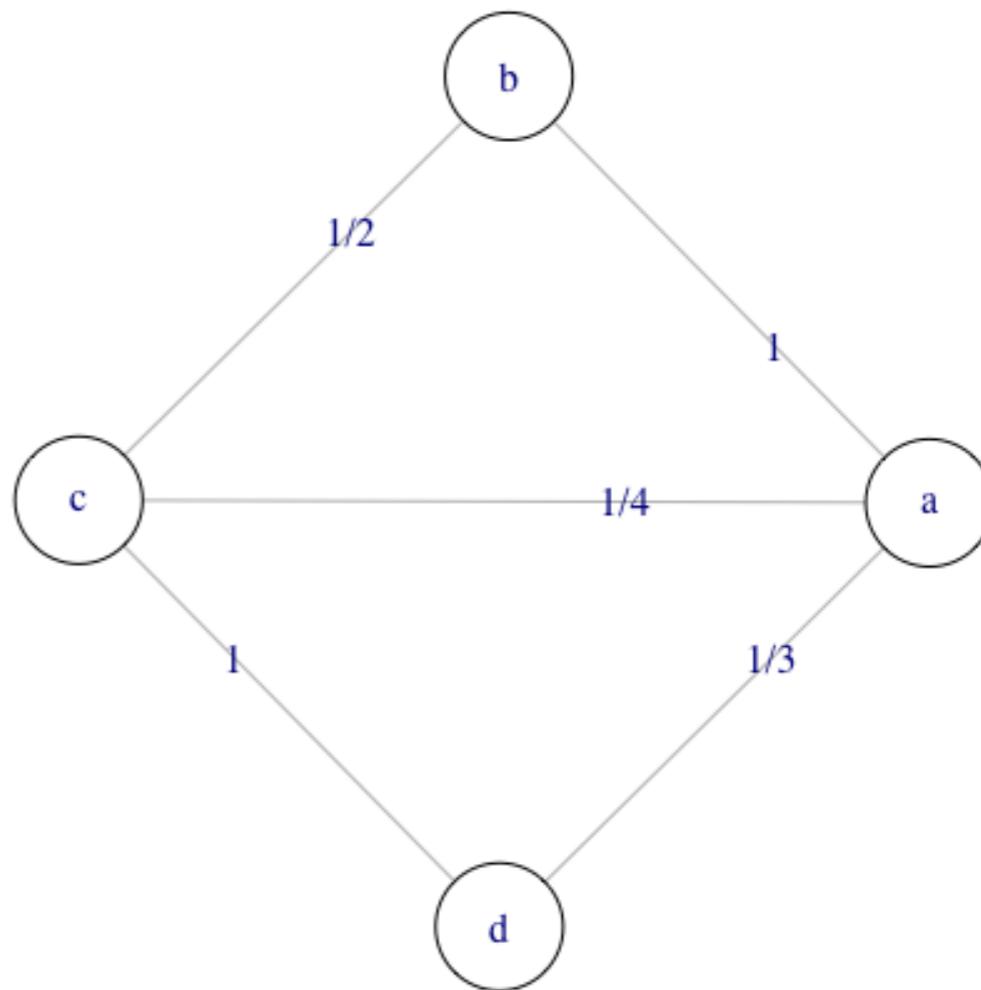
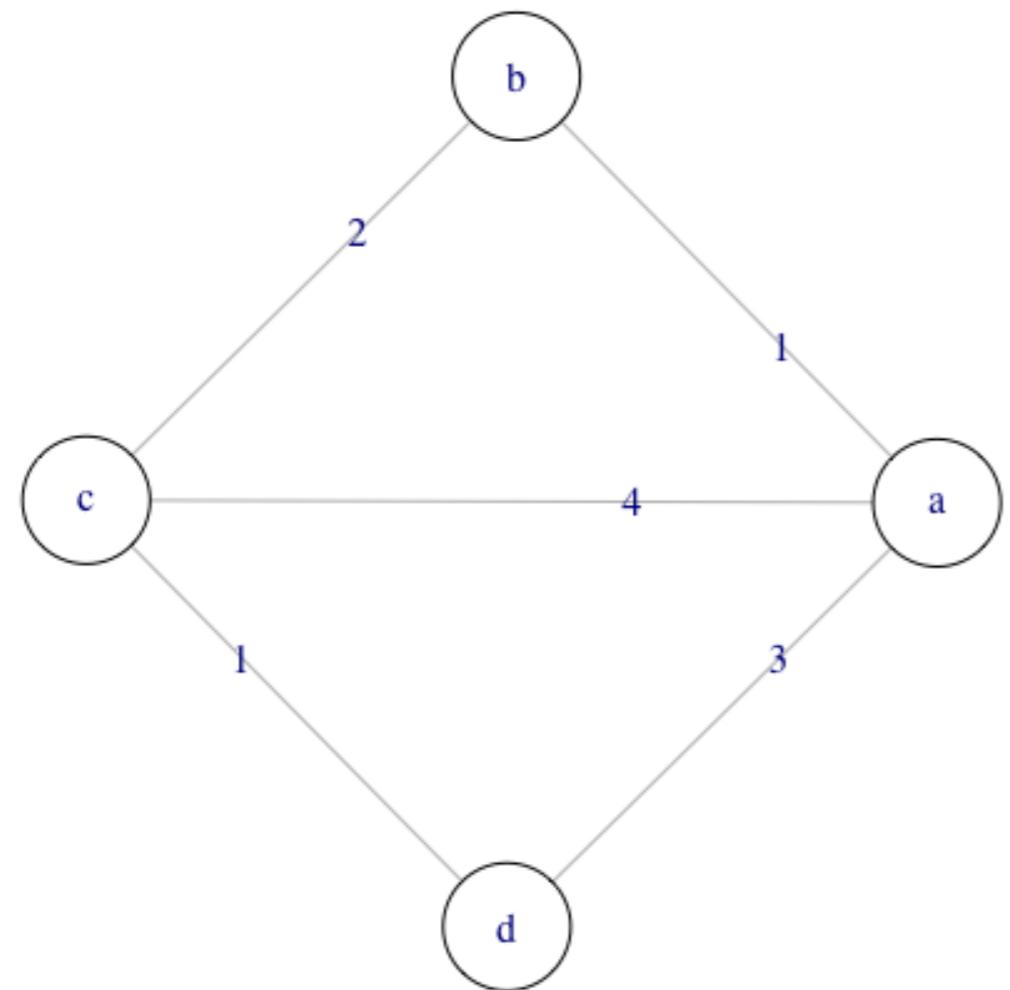


Massimo Franceschet

Prof. of Data Science, University of
Udine (Italy)

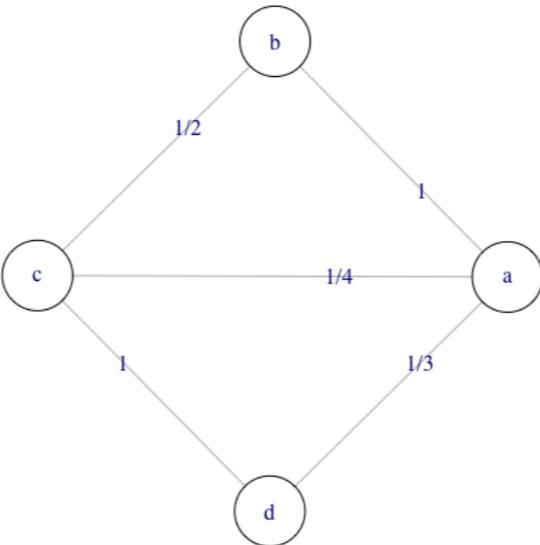


Weighted betweenness



Computing betweenness

```
# compute distance weights for ties  
dist_weight = 1 / E(g)$weight  
  
# compute weighted betweenness on ties  
edge_betweenness(g, weights = dist_weight)
```



**Let's start practicing
with tie
betweenness!**

NETWORK ANALYSIS IN THE TIDYVERSE

Visualizing centrality measures

NETWORK ANALYSIS IN THE TIDYVERSE

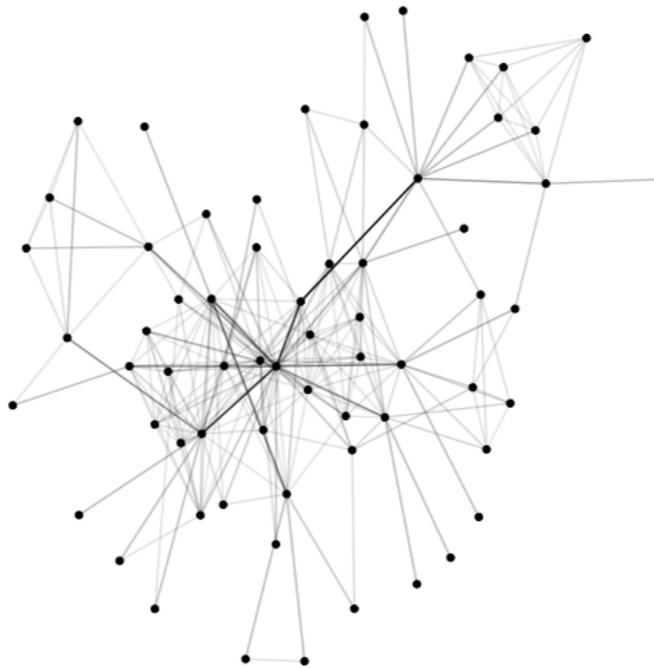


Massimo Franceschet

Prof. of Data Science, University of
Udine (Italy)

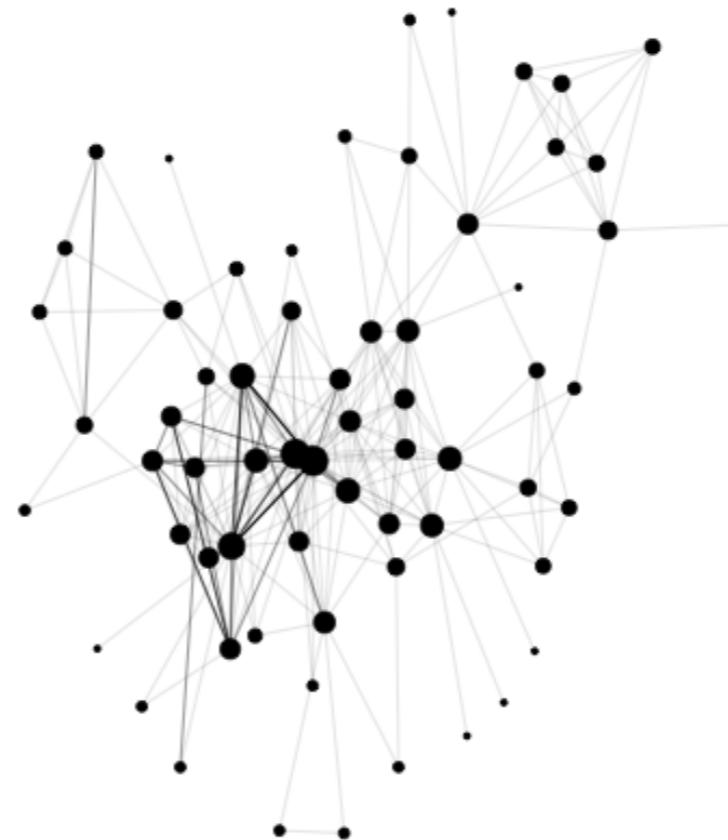
Visualizing betweenness

```
# visualize the network with tie transparency proportional to betweenness  
ggraph(g, layout = "with_kk") +  
  geom_edge_link(aes(alpha = betweenness)) +  
  geom_node_point()
```



Visualizing weight and degree

```
# visualize tie weight and node degree  
ggraph(g, layout = "with_kk") +  
  geom_edge_link(aes(alpha = weight)) +  
  geom_node_point(aes(size = degree))
```



Let's practice!

NETWORK ANALYSIS IN THE TIDYVERSE

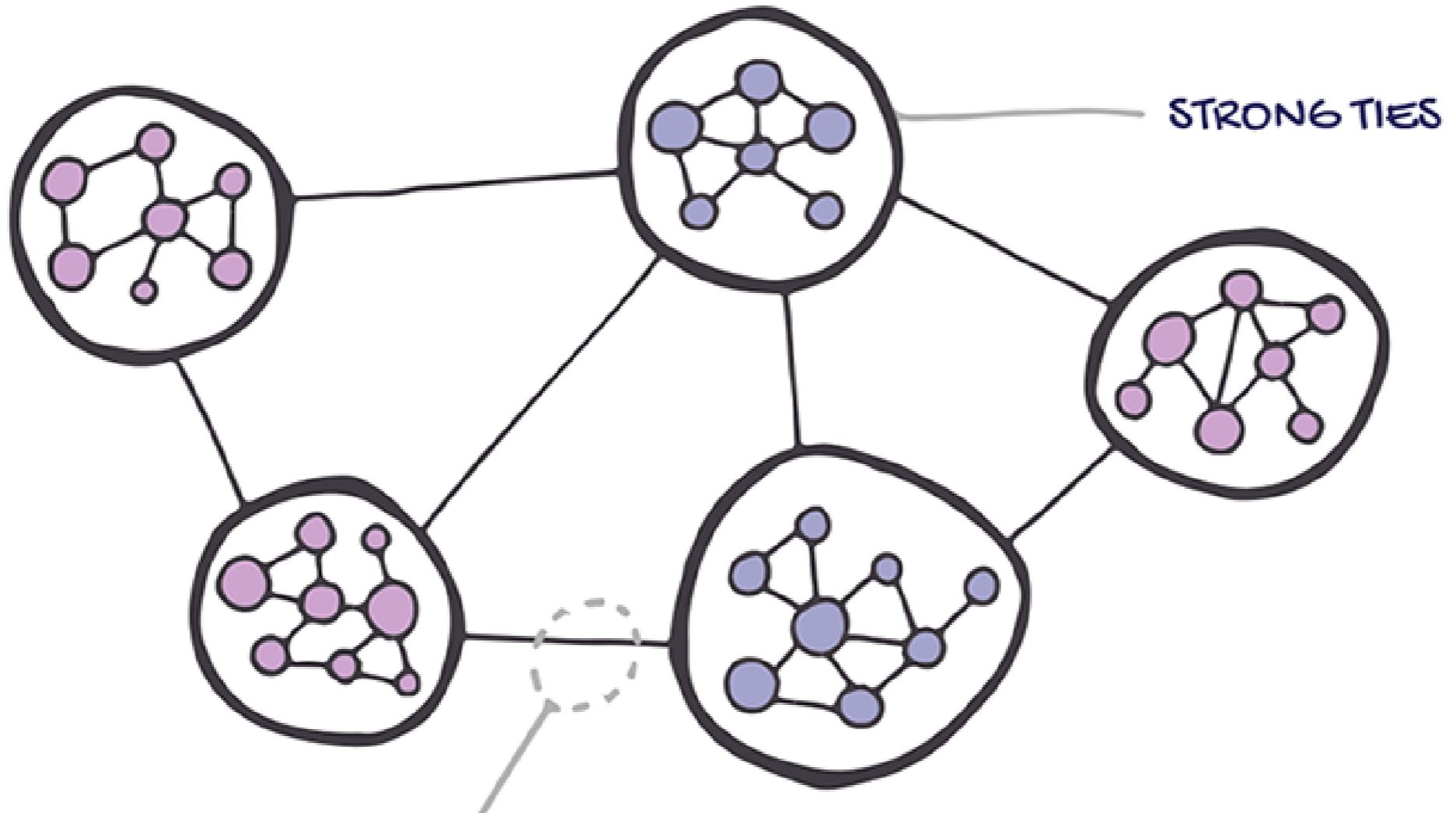
The strength of weak ties

NETWORK ANALYSIS IN THE TIDYVERSE



Massimo Franceschet

Prof. of Data Science, University of
Udine (Italy)



STRONG TIES

THE CONNECTIONS BETWEEN
MEMBERS OF SEPARATE GROUPS

Weak ties

- **Weak ties** are relationships between members of different communities. They lead to a diversity of ideas



Strong ties

- **Strong ties** are relationships between people who live, work, or play together. They lead to similar and stagnant ideas



In its weakness lies its strength

- Unlike conventional armed groups, which are often hierarchical and centralized
 - Large terrorist networks use **dispersed forms of organization**
- Balances covertness with broader operational support
- Easier to reconstruct without dependencies on strong relationships

Finding weak ties

```
# find number and percentage of weak ties
ties %>%
  group_by(weight) %>%
  summarise(n = n(), p = n / nrow(ties)) %>%
  arrange(-n)
```

```
# A tibble: 4 x 3
  weight     n      p
  <int> <int>  <dbl>
1     1    214 0.881
2     2     21 0.0864
3     3      6 0.0247
4     4      2 0.00823
```

**Let's find weak and
strong ties in our
network!**

NETWORK ANALYSIS IN THE TIDYVERSE

Connection patterns

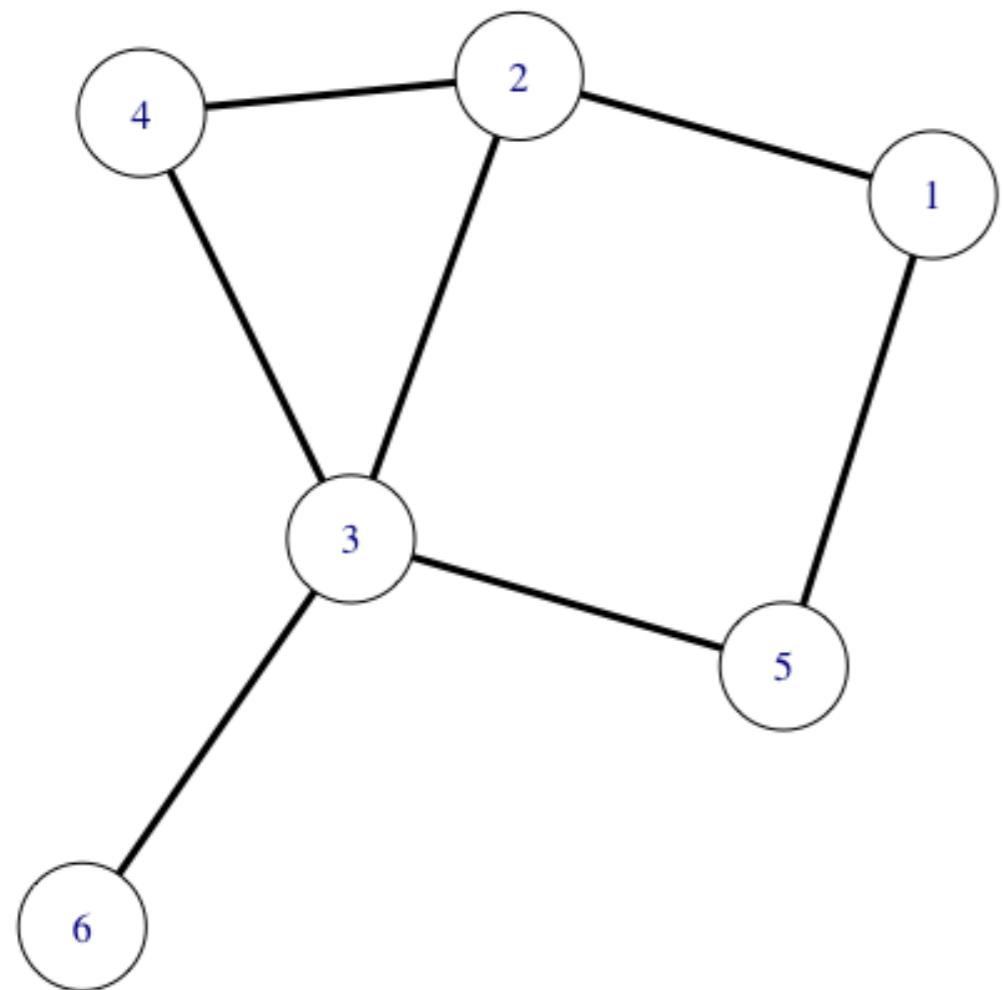
NETWORK ANALYSIS IN THE TIDYVERSE



Massimo Franceschet

Prof. of Data Science, University of
Udine (Italy)

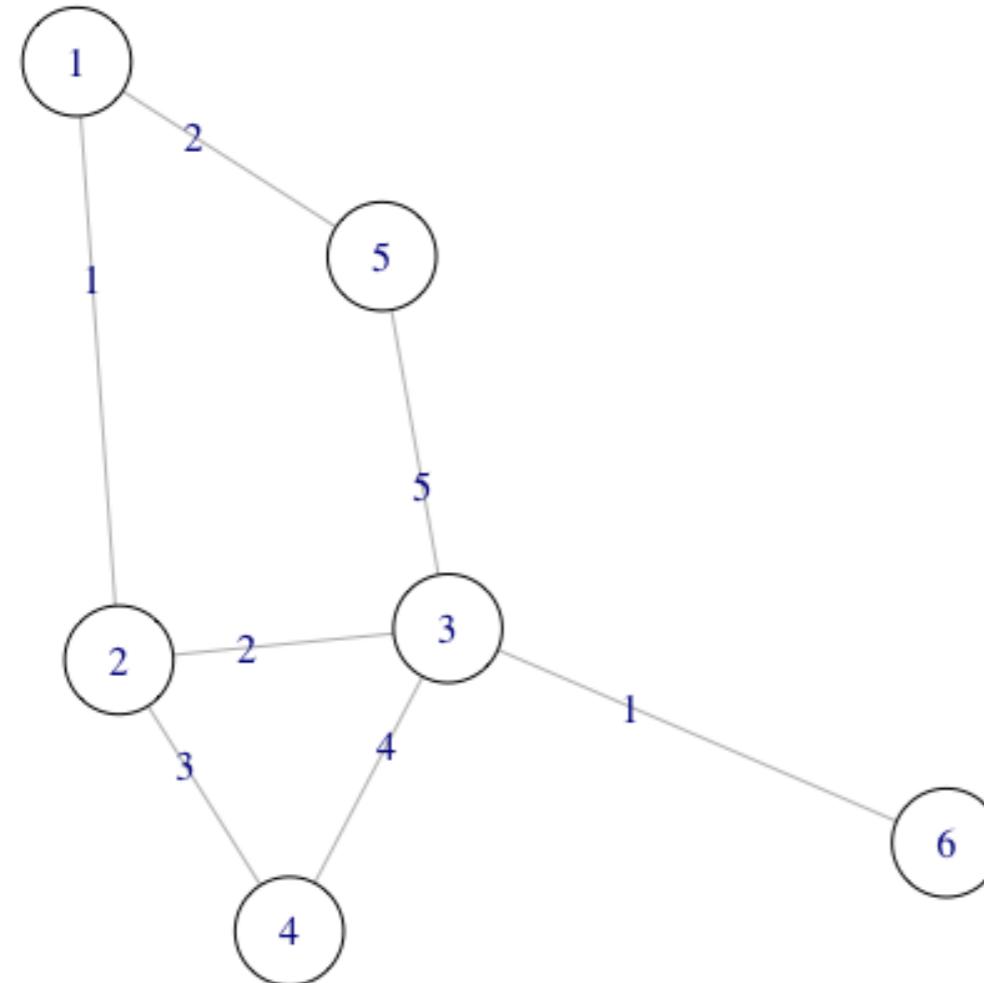
The adjacency matrix (part 1)



`as_adjacency_matrix(g)`

	1	2	3	4	5	6
1	0	1	0	0	1	0
2	1	0	1	1	0	0
3	0	1	0	1	1	1
4	0	1	1	0	0	0
5	1	0	1	0	0	0
6	0	0	1	0	0	0

The adjacency matrix (part 2)



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

	1	2	3	4	5	6
1	0	1	0	0	2	0
2	1	0	2	3	0	0
3	0	2	0	4	5	1
4	0	3	4	0	0	0
5	2	0	5	0	0	0
6	0	0	1	0	0	0

Working with adjacency matrices

```
# get the adjacency matrix of network g  
A = as_adjacency_matrix(g)  
# get the weighted adjacency matrix of weighted network g  
A = as_adjacency_matrix(g, attr = "weight")
```

```
# first row of matrix A
```

```
A[1, ]
```

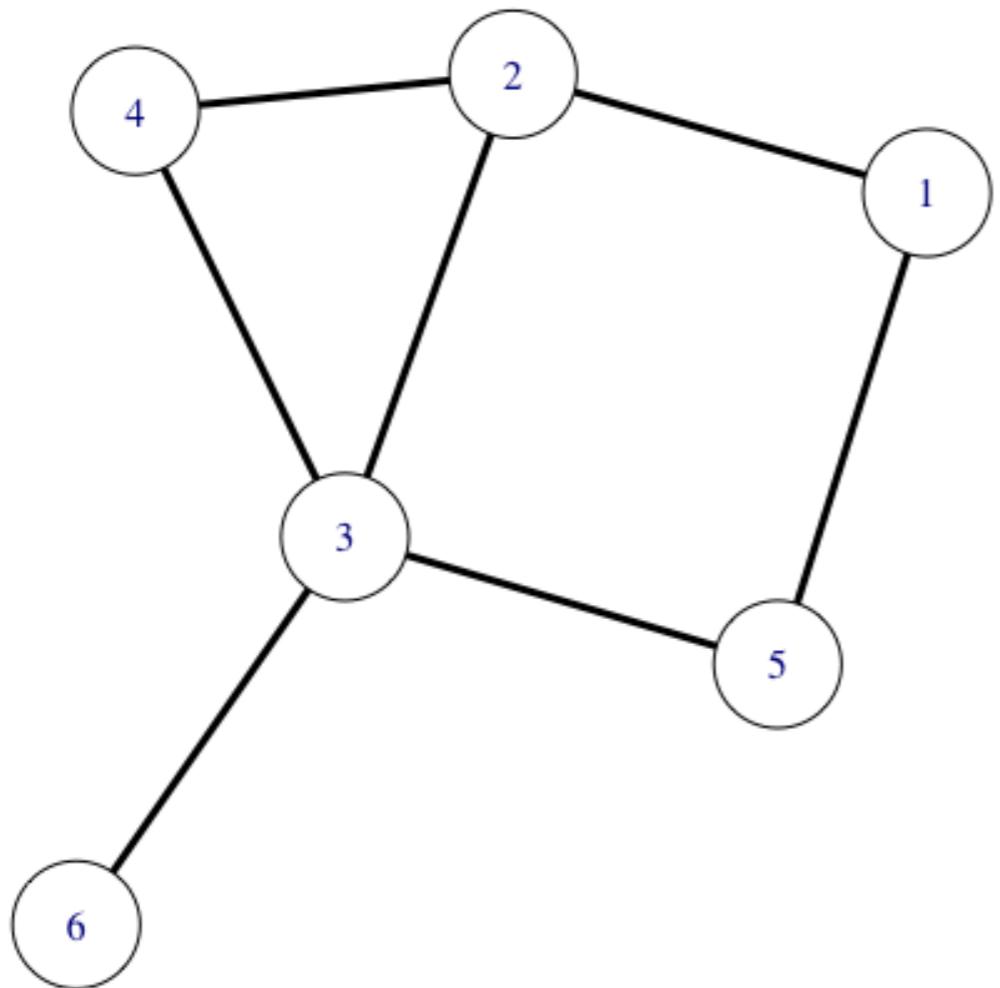
```
# first column of matrix A
```

```
A[, 1]
```

```
# diagonal of matrix A
```

```
diag(A)
```

Pearson similarity



```
as_adjacency_matrix(g)
```

	[,1]	[,2]	[,3]	[,4]	[,5]	[,6]
[1,]	0	1	0	0	1	0
[2,]	1	0	1	1	0	0
[3,]	0	1	0	1	1	1
[4,]	0	1	1	0	0	0
[5,]	1	0	1	0	0	0
[6,]	0	0	1	0	0	0

Let's try some examples!

NETWORK ANALYSIS IN THE TIDYVERSE

Pearson similarity

NETWORK ANALYSIS IN THE TIDYVERSE

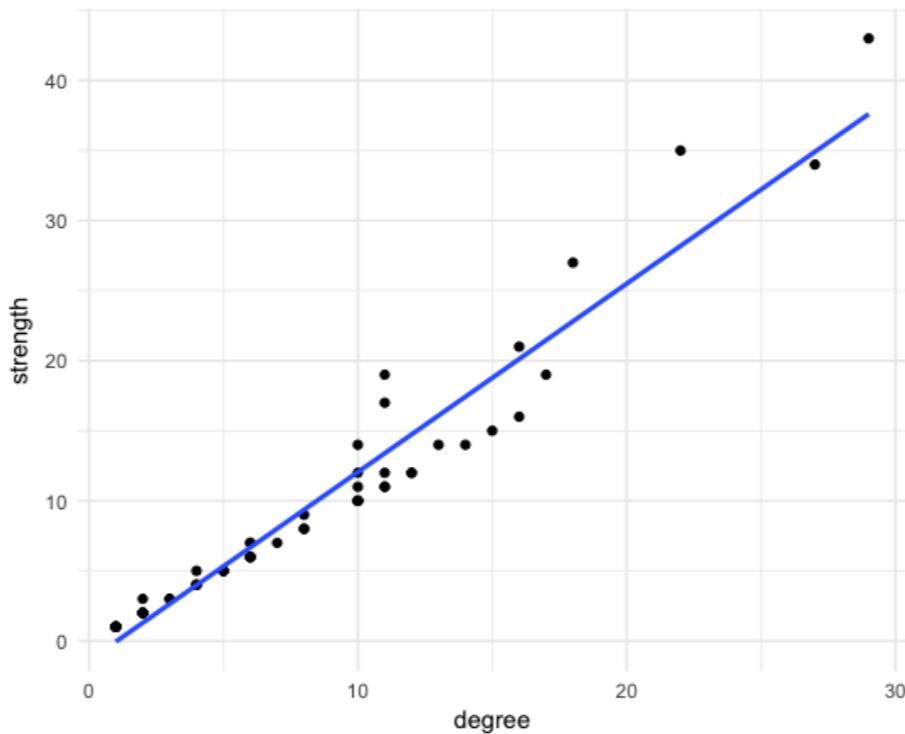


Massimo Franceschet

Prof. of Data Science, University of
Udine (Italy)

Visualizing correlation

```
# scatterplot of degree and strength
ggplot(data = nodes, mapping = aes(x = degree, y = strength)) +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE)
```



Computing correlation

- Positive values indicate positive correlation
- Negative values indicate negative correlation
- Null values indicate no correlation

```
# Pearson correlation coefficient  
cor(nodes$degree, nodes$strength)
```

```
0.9708946
```

Let's practice!

NETWORK ANALYSIS IN THE TIDYVERSE

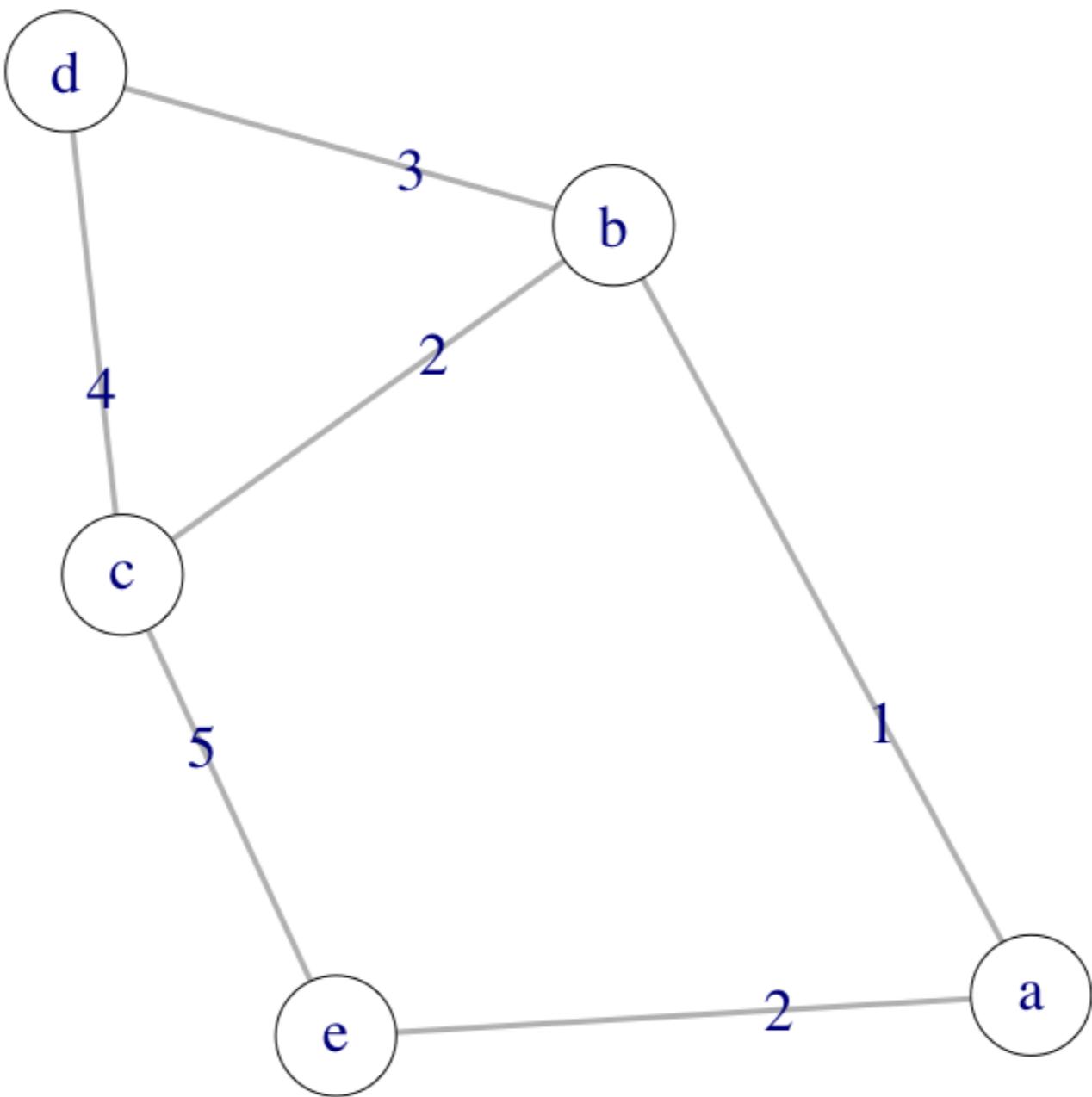
Most similar and most dissimilar nodes

NETWORK ANALYSIS IN THE TIDYVERSE

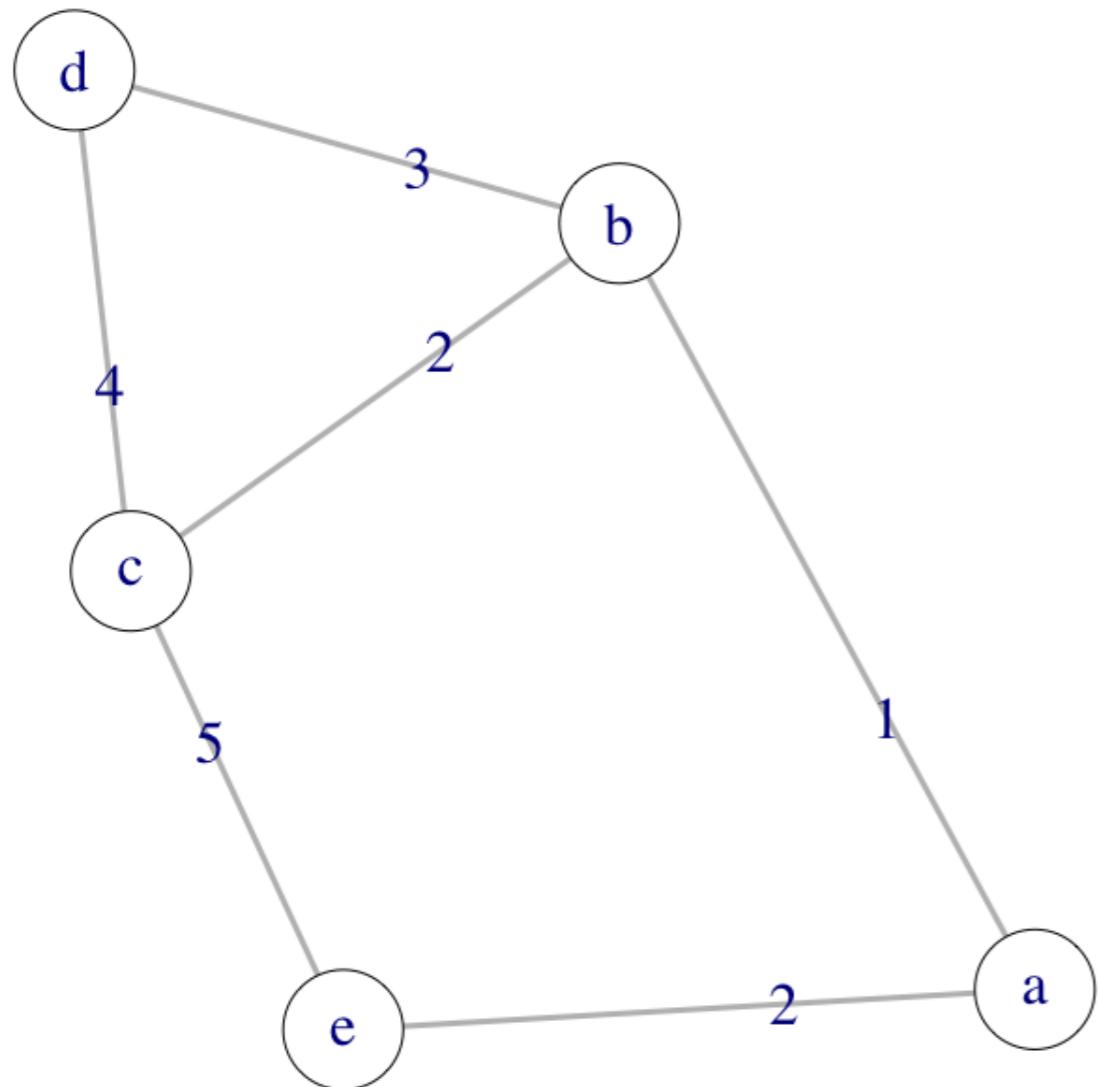
Massimo Franceschet

Prof. of Data Science, University of
Udine (Italy)





A network as a matrix



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

	a	b	c	d	e
a	0	1	0	0	2
b	1	0	2	3	0
c	0	2	0	4	5
d	0	3	4	0	0
e	2	0	5	0	0

A network as a data frame

```
as_data_frame(g, what = "both")
```

```
$nodes  
  name  
  a  
  b  
  c  
  d  
  e
```

```
$ties  
  from to weight  
  1   a   b     1  
  2   a   e     2  
  3   b   c     2  
  4   b   d     3  
  5   c   d     4  
  6   c   e     5
```

Mapping representations

```
# graph to matrix
A <- as_adjacency_matrix(g)

# matrix to graph
g <- graph_from adjacency_matrix(A)
# graph to data frame
df = as_data_frame(g, what = "both")

# data frame to graph
g <- graph_from_data_frame(df$ties, vertices = df$nodes)
# matrix to data frame
df = as_data_frame(graph_from adjacency_matrix(A), what = "both")

# data frame to matrix
A <- as_adjacency_matrix(graph_from_data_frame(df$ties,
vertices = df$nodes))
```

Let's try more examples!

NETWORK ANALYSIS IN THE TIDYVERSE

Hierarchical clustering

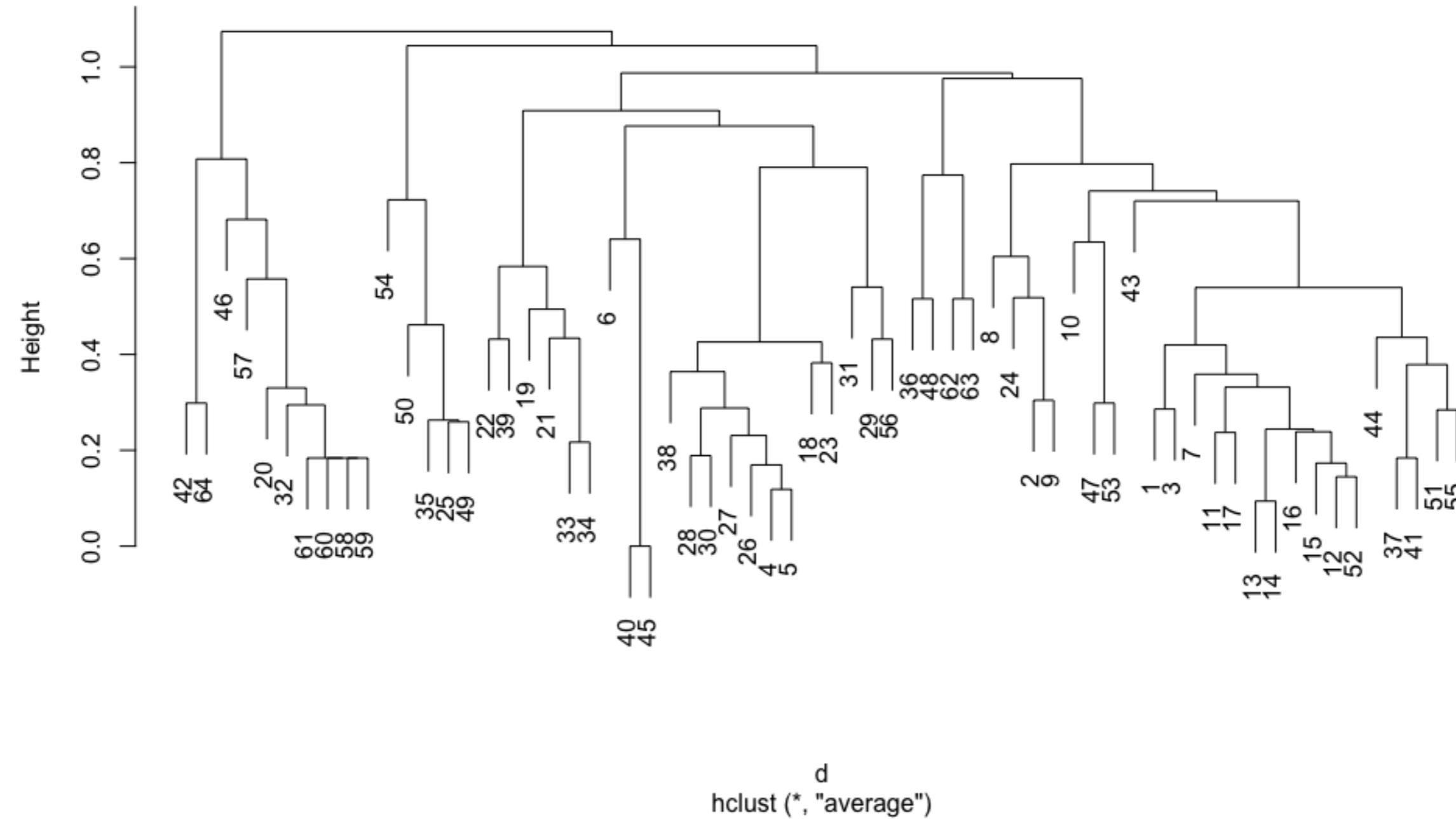
NETWORK ANALYSIS IN THE TIDYVERSE



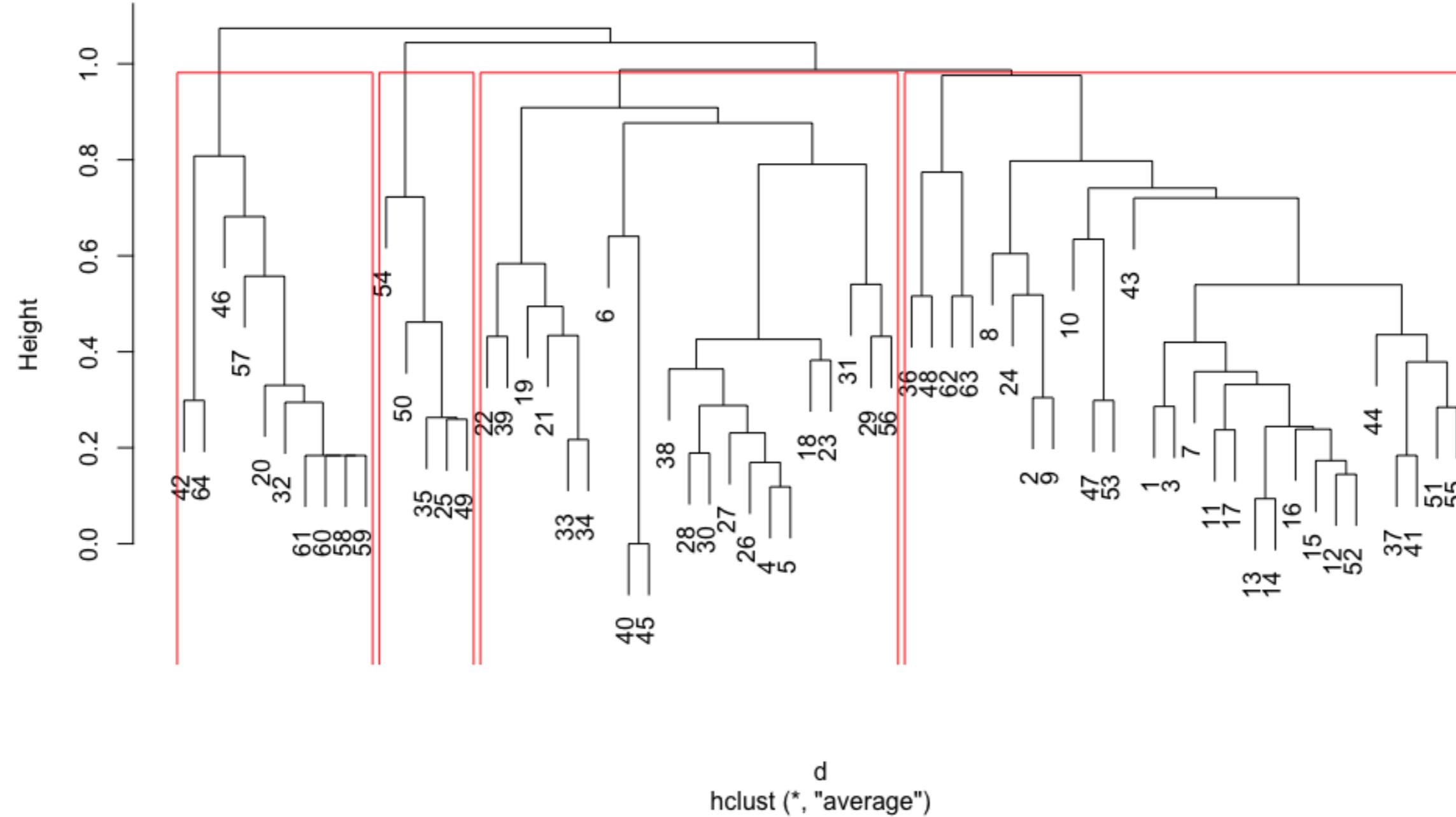
Massimo Franceschet

Prof. of Data Science, University of
Udine (Italy)

Cluster Dendrogram



Cluster Dendrogram



The similarity measure

- **Single-linkage:** the similarity between two groups is the maximum of the similarities between nodes of different groups.
- **Complete-linkage:** the similarity between two groups is the minimum of the similarities between nodes of different groups.
- **Average-linkage:** the similarity between two groups is the average of the similarities between nodes of different groups.

The clustering algorithm

1. Evaluate the similarity measures for all node pairs.
2. Assign each node to a group of its own.
3. Find the pair of groups with the highest similarity and join them together into a single group.
4. Calculate the similarity between the new composite group and all others.
5. Repeat steps 3 and 4 until all nodes have been joined into a single group.

Hierarchical clustering in R

```
# distance matrix from similarity matrix  
D <- 1-S  
  
# distance object from distance matrix  
d <- as.dist(D)  
  
# average-linkage clustering method  
cc <- hclust(d, method = "average")  
  
# cut dendrogram at 4 clusters  
hclust(d, method = "average")
```

```
[1] 1 1 1 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 3 2 2 2 1 4 2 2 2  
[29] 2 2 2 3 2 2 4 1 1 2 2 2 1 3 1 1 2 3 1 1 4 4 1 1 1 4 1 2  
[57] 3 3 3 3 3 1 1 3
```

Let's cluster our network!

NETWORK ANALYSIS IN THE TIDYVERSE

Interactive visualizations with visNetwork

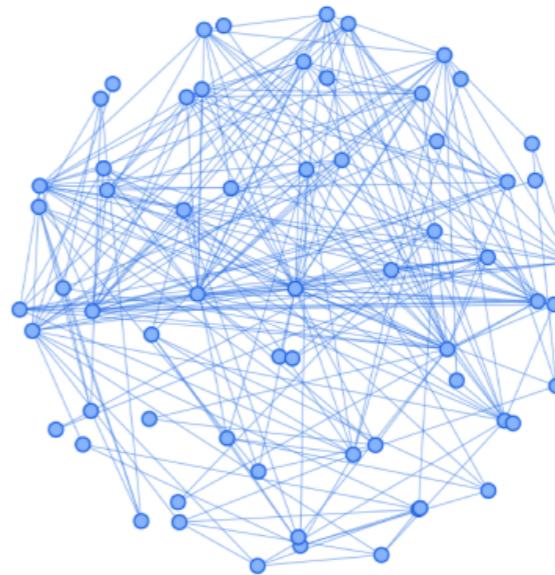
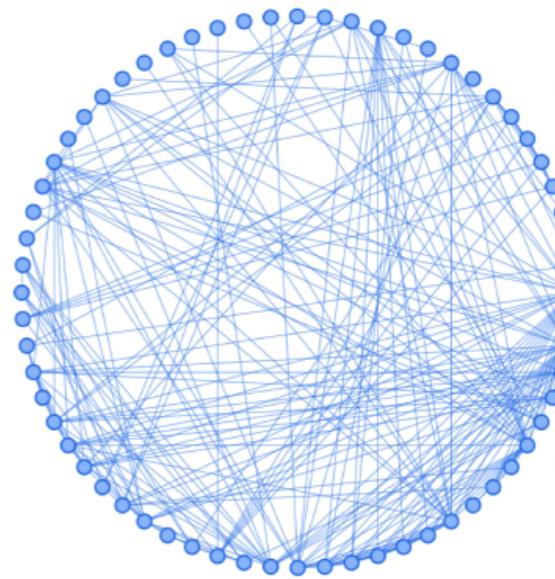
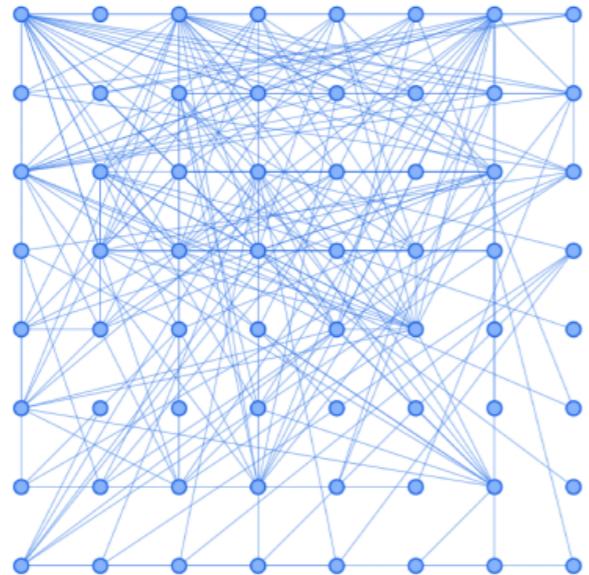
NETWORK ANALYSIS IN THE TIDYVERSE

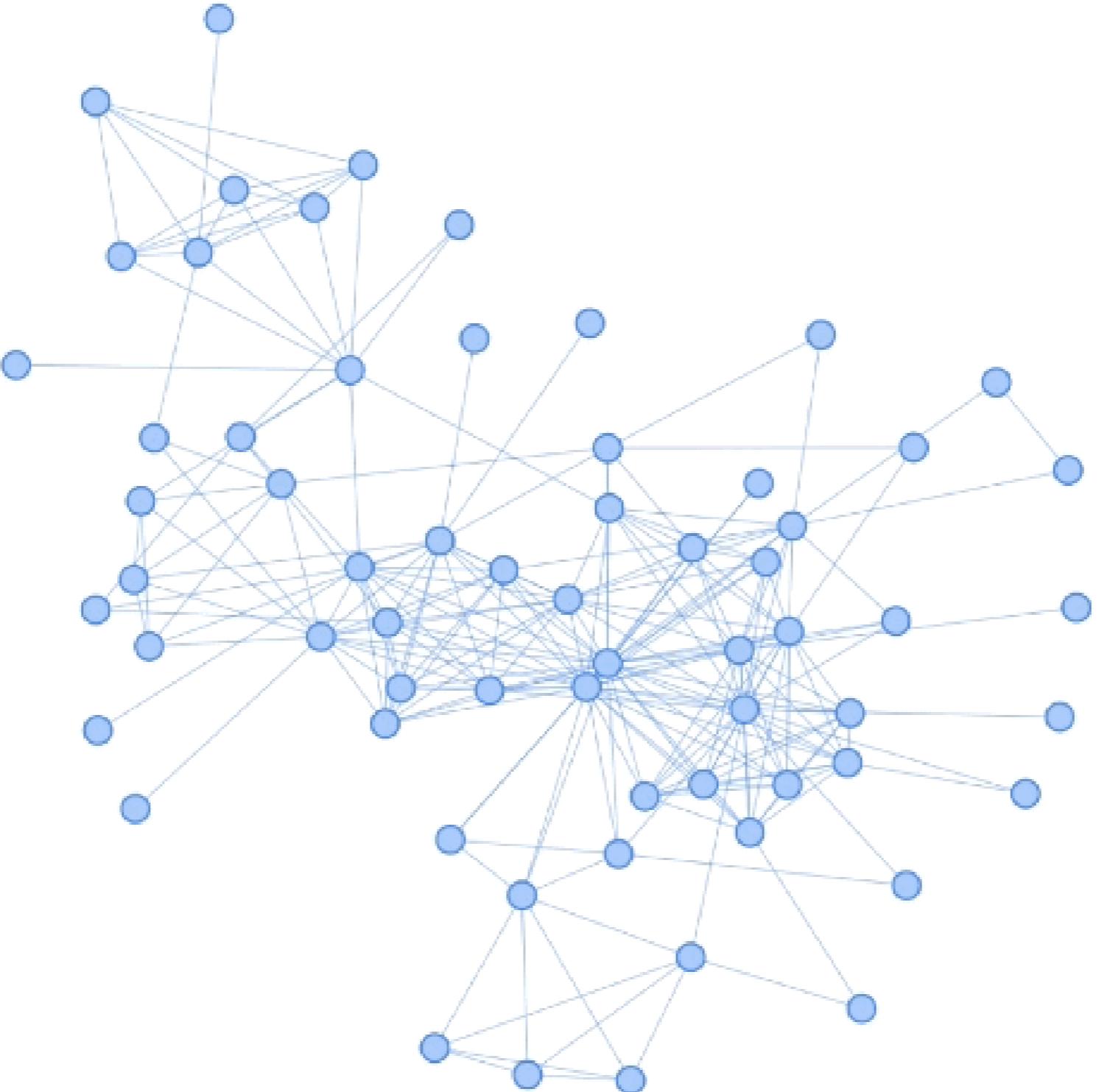
Massimo Franceschet

Prof. of Data Science, University of
Udine (Italy)



Different layouts





Select by id



Select by group



Let's interact!

NETWORK ANALYSIS IN THE TIDYVERSE

Congratulations!

NETWORK ANALYSIS IN THE TIDYVERSE



Massimo Franceschet

Prof. of Data Science, University of
Udine (Italy)

Deeper inside network science

You now know how to:

- Analyze any network with basic centrality and similarity measures
- Produce beautiful network visualizations, including interactive ones

For more information:

- [University of Udine Network Science Course](#)

Continue the journey!

NETWORK ANALYSIS IN THE TIDYVERSE

Exploring your data set

CASE STUDIES: NETWORK ANALYSIS IN R



Edmund Hart

Instructor

```
library(igraph)
library(dplyr)
amzn_raw <- read.csv("datasets/amazon_purchase_no_book.csv")
head(amzn_raw)
```

```
from to title.from group.from categories.from salesrank.from
1 1 44 42 The NBA's 100 Greatest Plays DVD 33124 5
2 2 179 71 Africa Screams/Jack & The Bean DVD 53825 21

totalreviews.from totalreviews.1.from
1 13 13
2 13 13

title.to group.to
1 Pixote DVD
2 Jonny Quest - Bandit in Adventures Best Friend Video

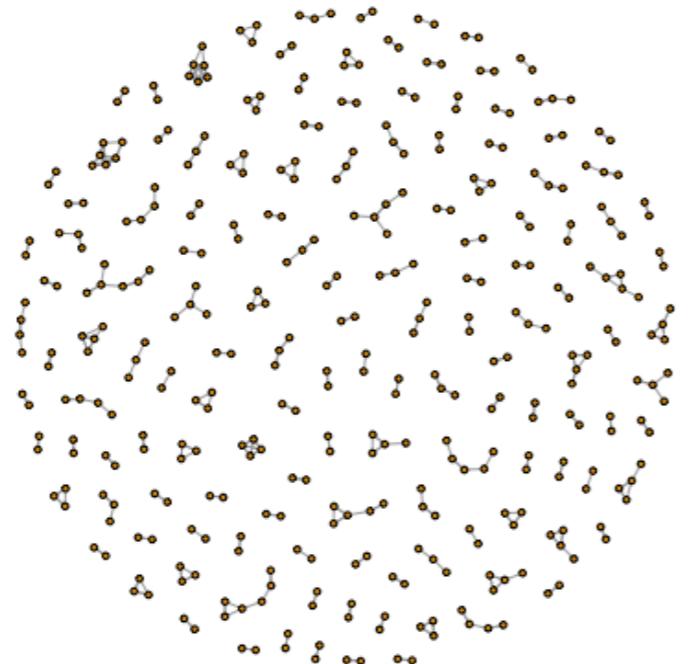
categories.to salesrank.to totalreviews.to totalreviews.1.to date
1 19685 15 24 24 2003-03-02
2 21571 5 2 2 2003-03-02
```

Creating the graph

```
amzn_g <- amzn_raw %>%  
  filter(date == "2003-03-02") %>%  
  select(from, to) %>%  
  graph_from_data_frame(directed = TRUE)  
gorder(amzn_g)  
gsize(amzn_g)
```

Visualize the graph

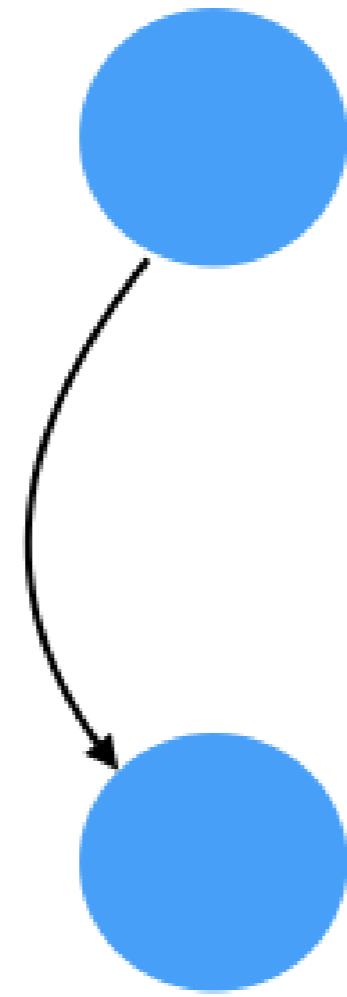
```
sg <- induced_subgraph(amzn_g, 1:500)
sg <- delete.vertices(sg, degree(sg) == 0)
plot(sg, vertex.label = NA, edge.arrow.width = 0,
     edge.arrow.size = 0, margin = 0, vertex.size = 2)
```



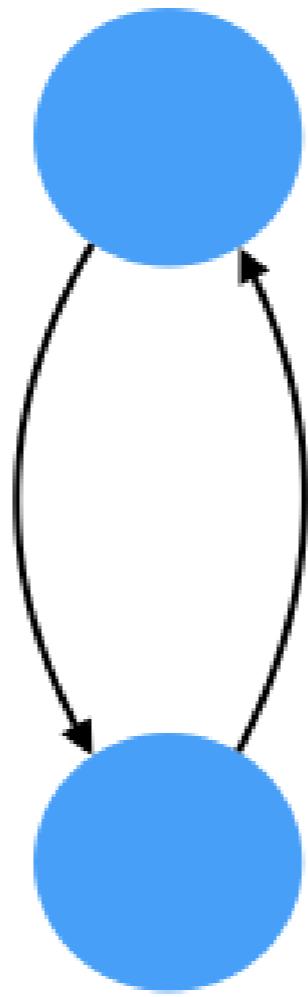
Null

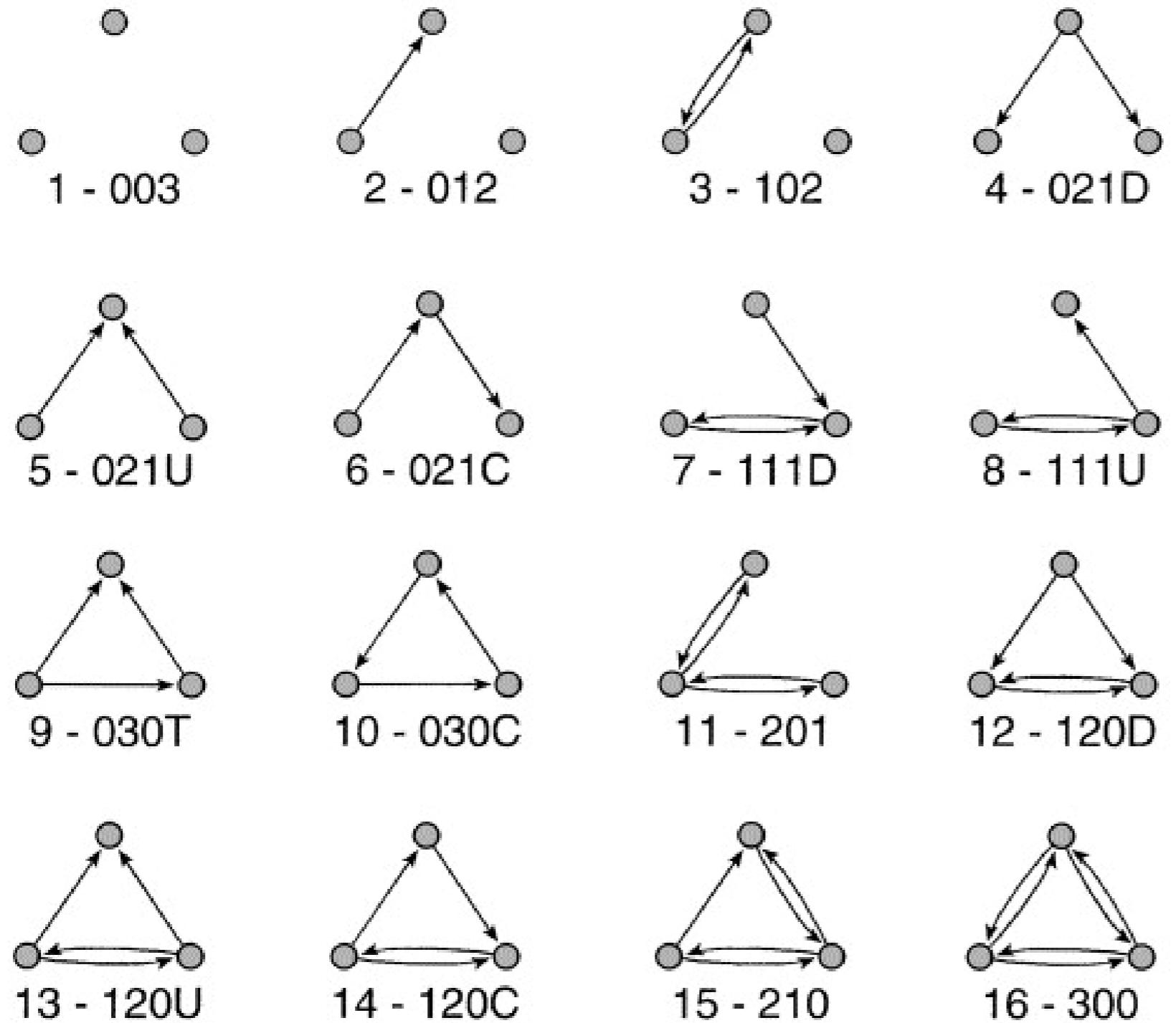


Asymmetric



Mutual





Let's practice

CASE STUDIES: NETWORK ANALYSIS IN R

Exploring temporal structure

CASE STUDIES: NETWORK ANALYSIS IN R



Edmund Hart

Instructor

Are important products always important?

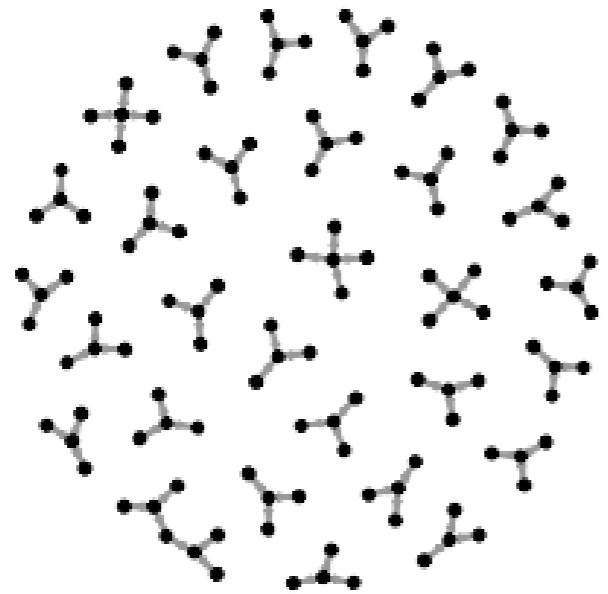
```
# Get unique Dates  
d <- sort(unique(amzn_raw$date))  
  
# Create graph from first date  
amzn_g <- graph_from_data_frame(  
  amzn_raw %>%  
    filter(date == d[1]) %>%  
    select(from, to), directed = TRUE  
)
```

Are important products always important?

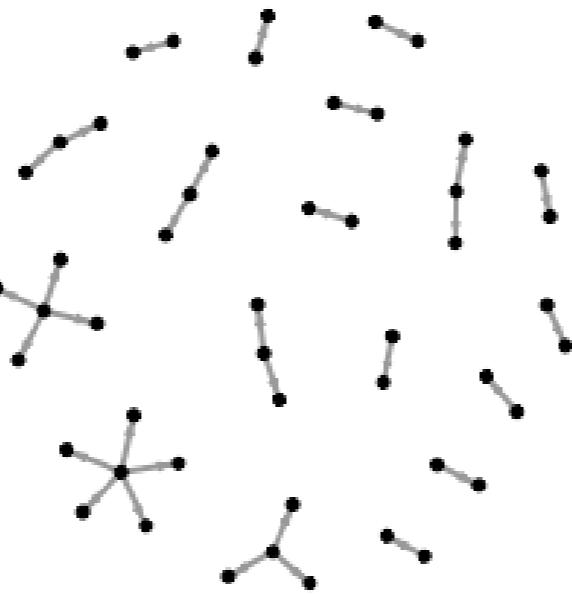
```
# Find products that are "important"  
high_out_degree <- degree(amzn_g, mode = "out") > 2  
  
low_in_degree <- degree(amzn_g, mode = "in") < 1  
  
important_nodes <- high_out_degree & low_in_degree  
  
imp_prod <- V(amzn_g)[important_nodes]  
  
# Store as a data frame to later join on  
tmp_df <- data.frame(imp_prod = as.numeric(names(imp_prod)))
```

```
## Create list to hold output
time_graph <- list()
## Create a 2x2 layout for plots and increase margins
par(mfrow = c(2, 2), mar = c(1.1, 1.1, 1.1, 1.1))
## Loop over the data to build
for(i in 1:length(d)){
  ## Create a data frame at each time stamp
  ip_df <- amzn_raw %>%
    filter(date == d[i]) %>%
    right_join(tmp_df, by = c("from" = "imp_prod")) %>%
    na.omit()
  ## Create an igraph object from that data frame
  time_graph[[i]] <- ip_df %>%
    select(from, to) %>%
    graph_from_data_frame(directed = TRUE)
  ## See what important vertices look like by date
  plot(time_graph[[i]], main = d[i]) }
```

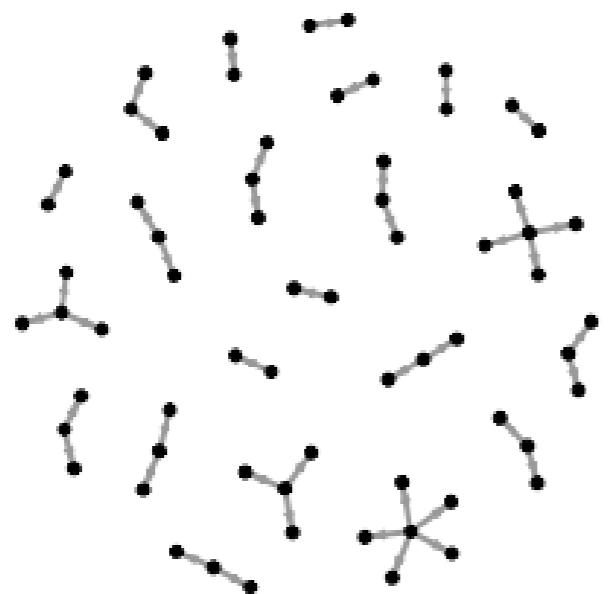
2003-03-02



2003-03-12



2003-05-05



2003-06-01



Let's practice!

CASE STUDIES: NETWORK ANALYSIS IN R

Building a graph from raw data

CASE STUDIES: NETWORK ANALYSIS IN R



Edmund Hart

Instructor

Exploring the data

- Data is several days of all the tweets mentioning #rstats
- Key attributes for building a graph are:
 - screen name
 - raw text of the tweet

Anatomy of a tweet

1. *ReecheshJC*: "Hey #rstats, how do I do fct_lump but where I lump based on count values in a column?"
2. *kom_256*: "RT @elenagbg: Retweeted R-Ladies Madrid (@RLadiesMAD):\n\nEn el #OCSummit17... Fast Talks sobre #rstats organizado por... <https://t.co/CKY5aG...>"

```
library(igraph)
library(stringr)
raw_tweets <- read.csv("datasets/rstatstweets.csv",
  stringsAsFactors = FALSE)
```

Data sample, single row

```
user_name:      Karen Millidine
screen_name:    KJMillidine
tweet_text:     RT @Rbloggers: RStudio v1.1 Released
https://t.co/kCMHc689nY #rstats #DataScience
favorites:      0
retweets:       96
location:       None
expanded_url:   https://wp.me/pMm6L-ExV
in_reply_to_tweet_id: NA
in_reply_to_user_id: NA
dt:             10/10/17
```

Building the graph

```
## Get all the screen names  
all_sn <- unique(raw_tweets$screen_name)  
  
## Create graph  
retweet_graph <- graph.empty()  
  
## Add screen names as vertices  
retweet_graph <- retweet_graph + vertices(all_sn)
```

Building the graph

```
## Extract name and add edges
for(i in 1:dim(raw_tweets)[1]){
  # Extract retweet name
  rt_name <- find_rt(raw_tweets$tweet_text[i])
  # If there is a name add an edge
  if(!is.null(rt_name)){
    # Check to make sure the vertex exists, if not, add it
    if(!rt_name %in% all_sn){
      retweet_graph <- retweet_graph + vertices(rt_name)
    }
    # add the edge
    retweet_graph <- retweet_graph +
      edges(c(raw_tweets$screen_name[i], rt_name))
  }
}
```

Cleaning the graph

```
## Size the number of degree 0 vertices  
sum(degree(retweet_graph) == 0)  
  
## Trim and simplify  
retweet_graph <- simplify(retweet_graph)  
retweet_graph <- delete.vertices(retweet_graph,  
degree(retweet_graph) == 0)
```

Let's practice!

CASE STUDIES: NETWORK ANALYSIS IN R

Building a mentions graph

CASE STUDIES: NETWORK ANALYSIS IN R



Edmund Hart

Instructor

Recall tweet anatomy

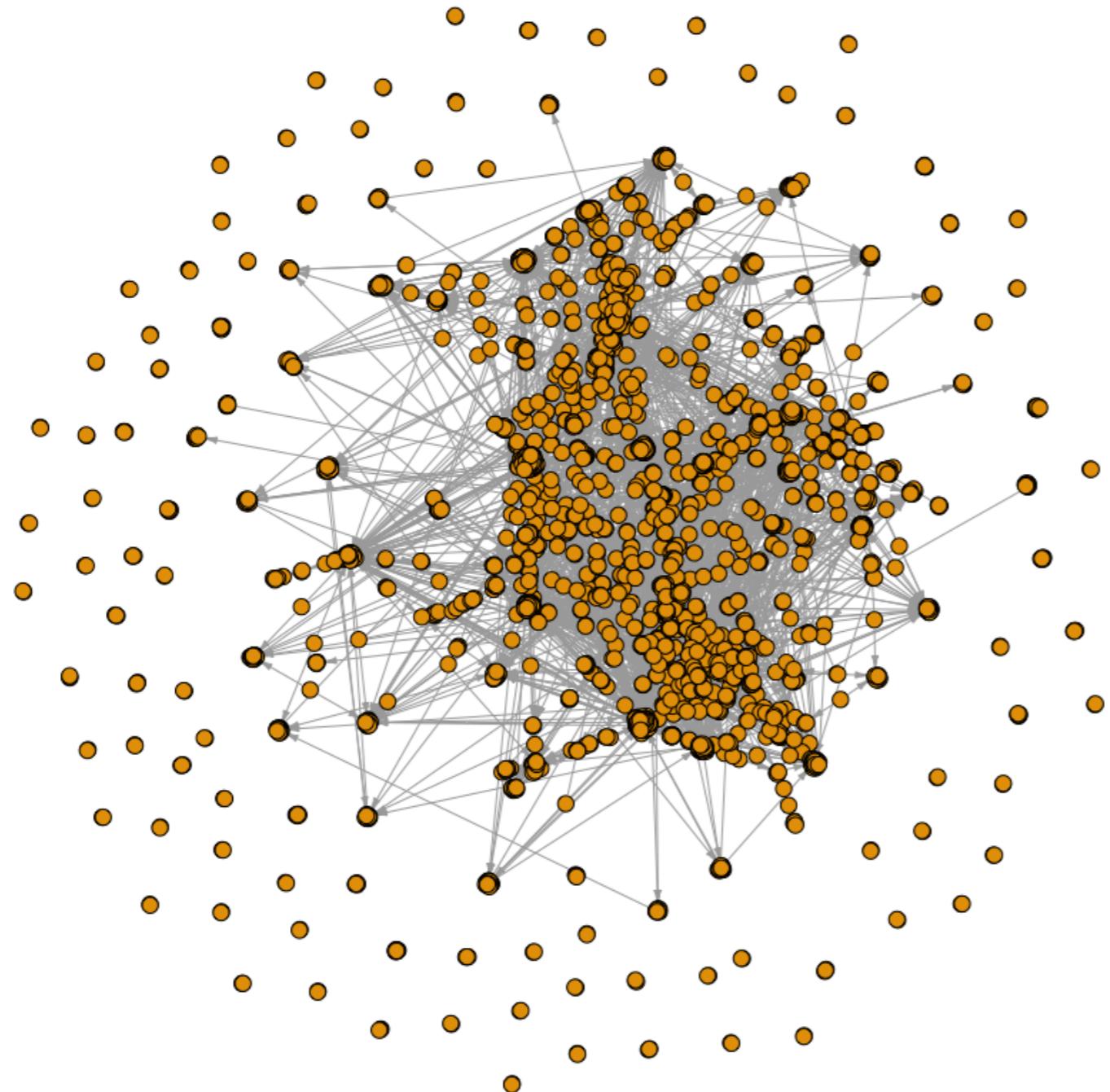
AlexisAchim: "@LAStools @Lees_Sandbox @jhollist
@LeahAWasser LidR is also available directly on CRAN #rstats"

timelyportfolio: "just might have a demo of @emeeks new
#reactjs/#d3js semiotic in #rstats in the works"

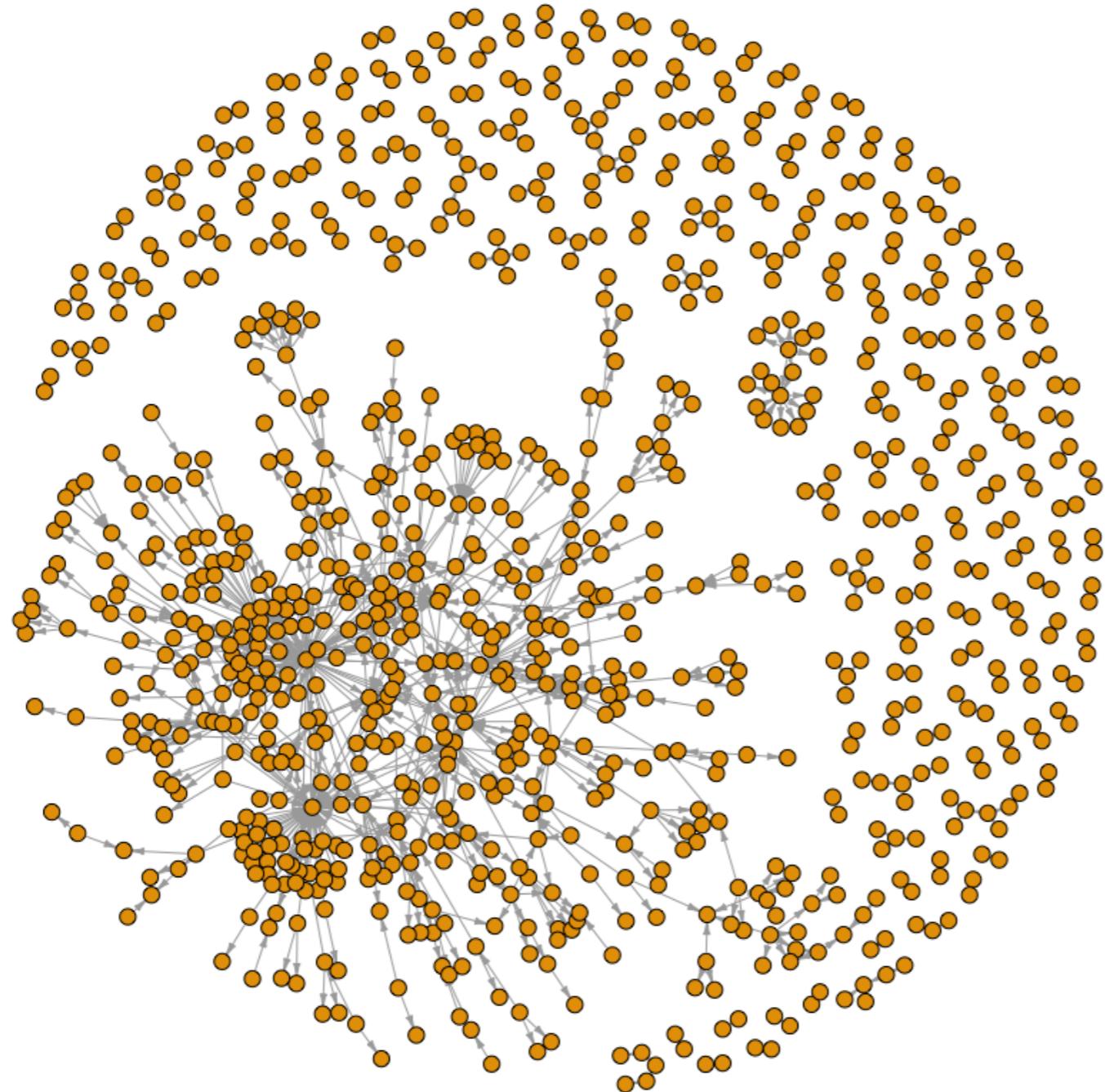
Build your mentions graph

```
ment_g <- graph.empty()
ment_g <- ment_g + vertices(all_sn)
for(i in 1:dim(raw_tweets)[1]) {
  ment_name <- mention_ext(raw_tweets$ tweet_text[i])
  if(length(ment_name) > 0 ) {
    # Add the edge(s)
    for(j in ment_name) {
      # Check to make sure the vertex exists, if not, add it
      if(!j %in% all_sn) {
        ment_g <- ment_g + vertices(j)  }
      ment_g <- ment_g + edges(c(raw_tweets$screen_name[i], j))
    }
  }
}
ment_g <- simplify(ment_g)
ment_g <- delete.vertices(ment_g, degree(ment_g) == 0)
```

Retweet Graph



Mentions Graph



Let's practice!

CASE STUDIES: NETWORK ANALYSIS IN R

Finding communities

CASE STUDIES: NETWORK ANALYSIS IN R



Edmund Hart
Instructor

Three different communities

```
undirected_ment_g <- as.undirected(ment_g)

ment_edg <- cluster_edge_betweenness(undirected_ment_g)
ment_eigen <- cluster_leading_eigen(undirected_ment_g)
ment_lp <- cluster_label_prop(undirected_ment_g)
```

Sizing the communities

```
length(ment_edg)  
length(ment_eigen)  
length(ment_lp)
```

```
173  
168  
212
```

```
table(sizes(ment_edg))
```

	2	3	4	5	6	7	8	9	11	12	18	19	20	23	24	26	28
103	21	14	7	3	3	1	2	1	2	2	1	1	1	1	2	1	1
31	33	38	40	41	52	58											
	1	1	1	1	1	1	1										

```
table(sizes(ment_eigen))
```

	2	3	4	5	6	7	9	10	12	18	23	26	29	30	32	34	35	58
103	22	14	7	4	3	1	1	1	1	1	1	1	1	1	1	1	1	1
64	66	101																
	1	1	1															

```
table(sizes(ment_lp))
```

	2	3	4	5	6	7	8	9	10	11	12	13	16	25	26	67	70
103	32	22	19	8	5	4	3	5	1	2	3	1	1	1	1	1	1

Comparing communities

```
compare(ment_edg, ment_eigen, method = 'vi')
```

```
0.9761792
```

```
compare(ment_eigen, ment_lp, method = 'vi')
```

```
1.192238
```

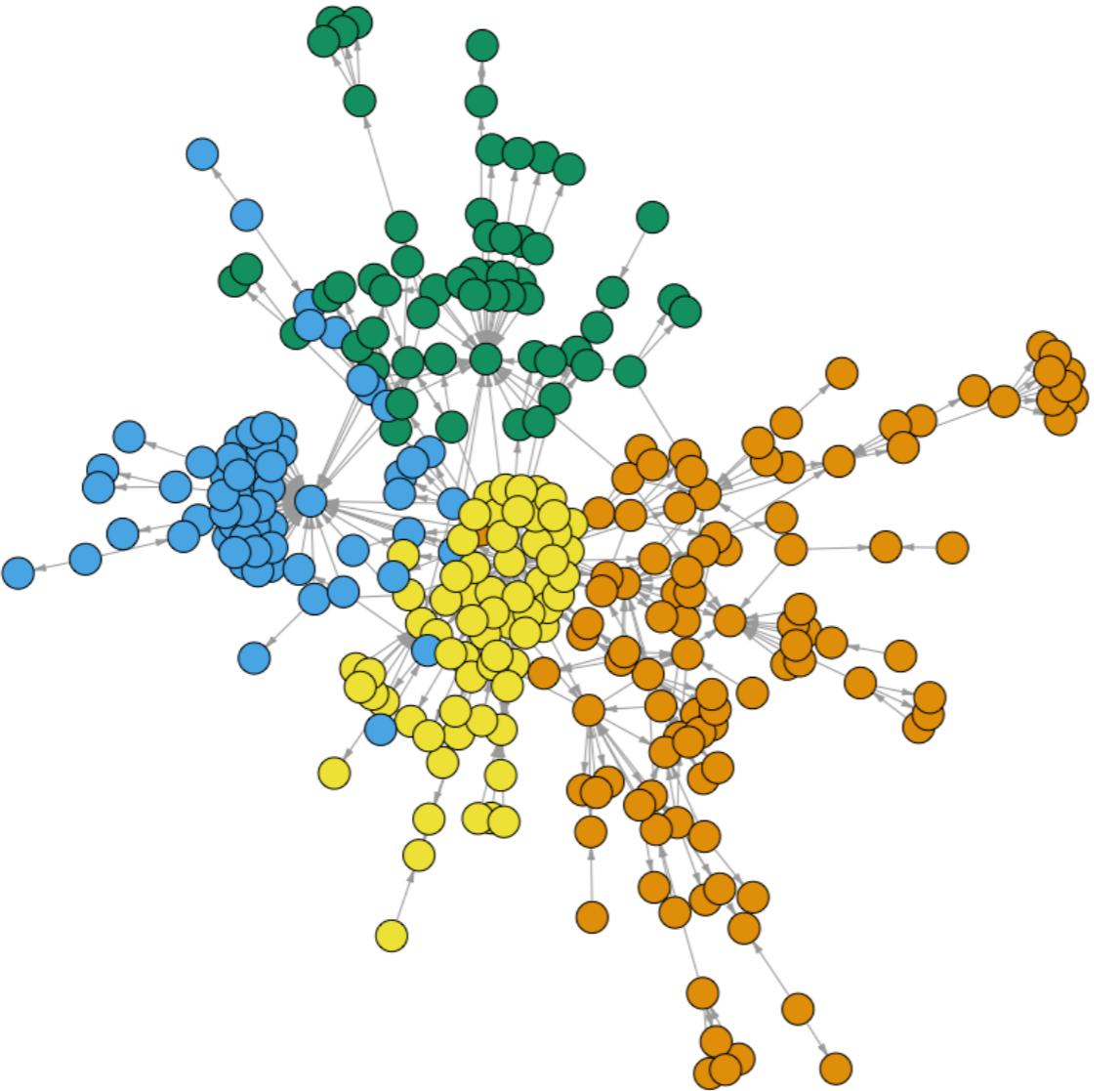
```
compare(ment_lp, ment_edg, method = 'vi')
```

```
0.9631608
```

Plotting community structure

```
lrg_eigen <- as.numeric(  
  names(ment_eigen[which(sizes(ment_eigen) > 45)])  
)  
  
eigen_sg <- induced.subgraph(ment_g,  
  V(ment_g)[ eigen %in% lrg_eigen])  
plot(eigen_sg, vertex.label = NA, edge.arrow.width = .8,  
  edge.arrow.size = 0.2,  
  coords = layout_with_fr(ment_sg), margin = 0,  
  vertex.size = 6, vertex.color =  
  as.numeric(as.factor(V(eigen_sg)$eigen)))
```

Mentions subgraph communities



Let's practice!

CASE STUDIES: NETWORK ANALYSIS IN R

Exploring our data

CASE STUDIES: NETWORK ANALYSIS IN R



Edmund Hart

Instructor

```
library(igraph)
library(dplyr)
library(lubridate)
bike_dat <- read.csv("datasets/bike2_test3.csv", stringsAsFactors = FALSE)
str(bike_dat)
```

```
'data.frame': 52800 obs. of 13 variables:
 $ tripduration : int 295 533 1570 2064 2257 296 412...
 $ from_station_id : int 49 165 25 300 85 174 75 45 85 99 ...
 $ from_station_name: chr "Dearborn St & Monroe St" ...
 $ to_station_id   : int 174 308 287 296 313 198 56 147 174 99 ...
 $ to_station_name : chr "Canal St & Madison St" ...
 $ usertype        : chr "Subscriber" "Subscriber" "Customer"...
 $ gender          : chr "Male" "Male" "" "" ...
 $ birthyear       : int 1964 1972 NA NA 1963 1973 1989 1965 1983 1983 ...
 $ from_latitude   : num 41.9 42 41.9 41.9 41.9 ...
 $ from_longitude  : num -87.6 -87.7 -87.6 -87.6 -87.6 ...
 $ to_latitude     : num 41.9 41.9 41.9 41.9 41.9 ...
 $ to_longitude    : num -87.6 -87.7 -87.6 -87.6 -87.6 ...
 $ geo_distance    : num 859 1882 2159 288 3044 ...
```

Creating the bike sharing graph

```
trip_df <- bike_dat %>%  
  group_by(from_station_id, to_station_id) %>%  
  summarize(weights = n())  
  
head(trip_df)
```

```
# A tibble: 6 x 3  
# Groups: from_station_id [1]  
  from_station_id to_station_id weights  
          <int>        <int>    <int>  
1             5            5        2  
2             5           14        1  
3             5           16        1  
4             5           25        3  
5             5           29        3  
6             5           33        1
```

Creating the bike sharing graph

```
trip_g <- graph_from_data_frame(trip_df[, 1:2])  
# add edge weights  
E(trip_g)$weight <- trip_df$weights  
# Quick exploration of our graph  
gsize(trip_g)
```

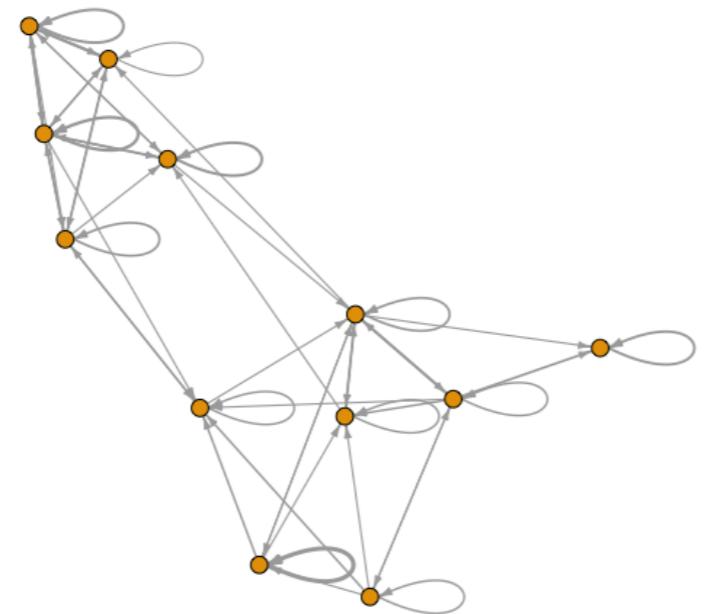
19052

```
gorder(trip_g)
```

300

Explore the graph

```
sg <- induced_subgraph(trip_g, 1:12)
plot(sg, vertex.label = NA, edge.arrow.width = 0.8,
     edge.arrow.size = 0.6,
     margin = 0,
     vertex.size = 6,
     edge.width = log(E(sg)$weight + 2))
```



Let's practice!

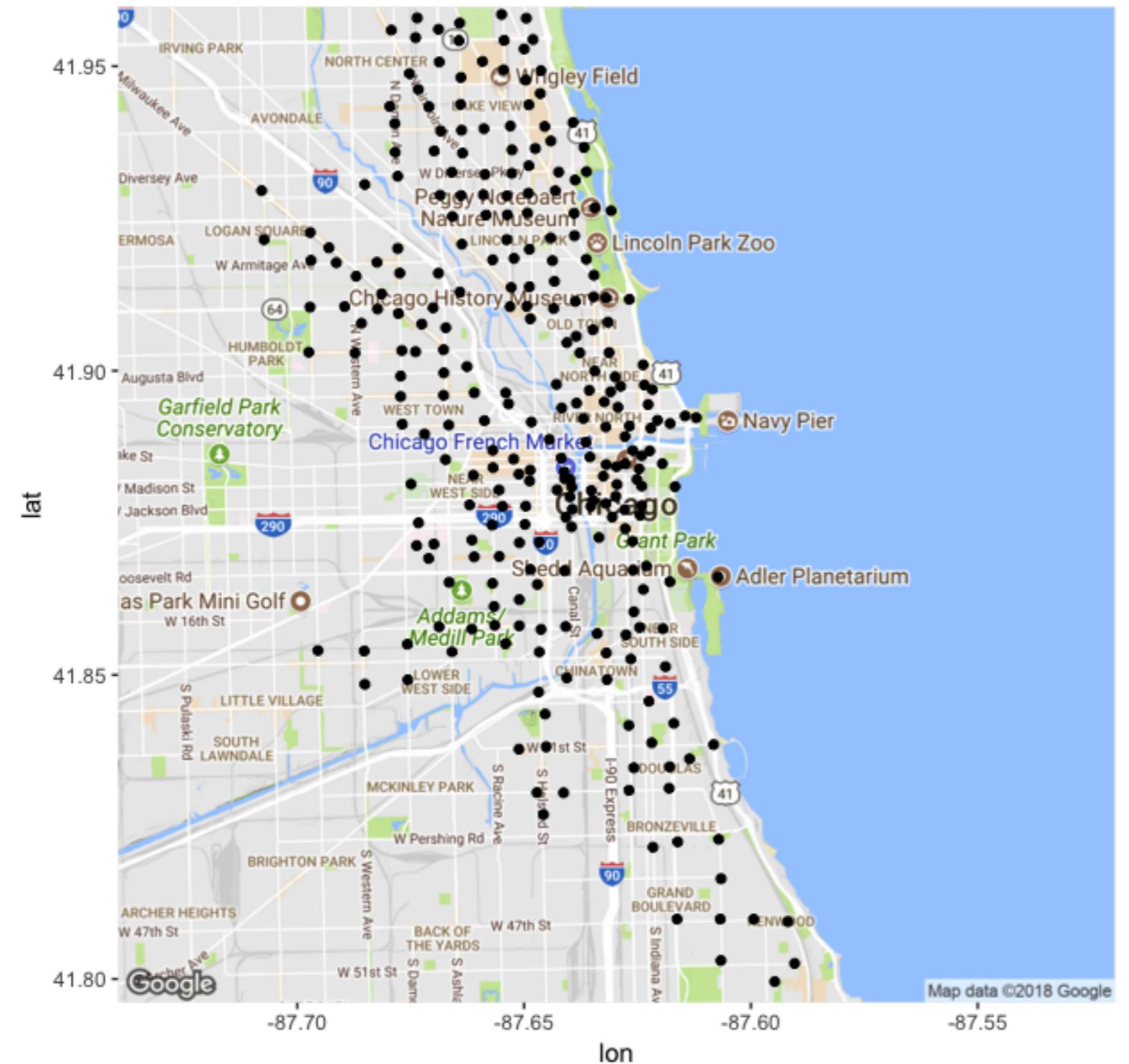
CASE STUDIES: NETWORK ANALYSIS IN R

Compare graph distance vs. geographic distance

CASE STUDIES: NETWORK ANALYSIS IN R



Edmund Hart
Instructor



Graph distance

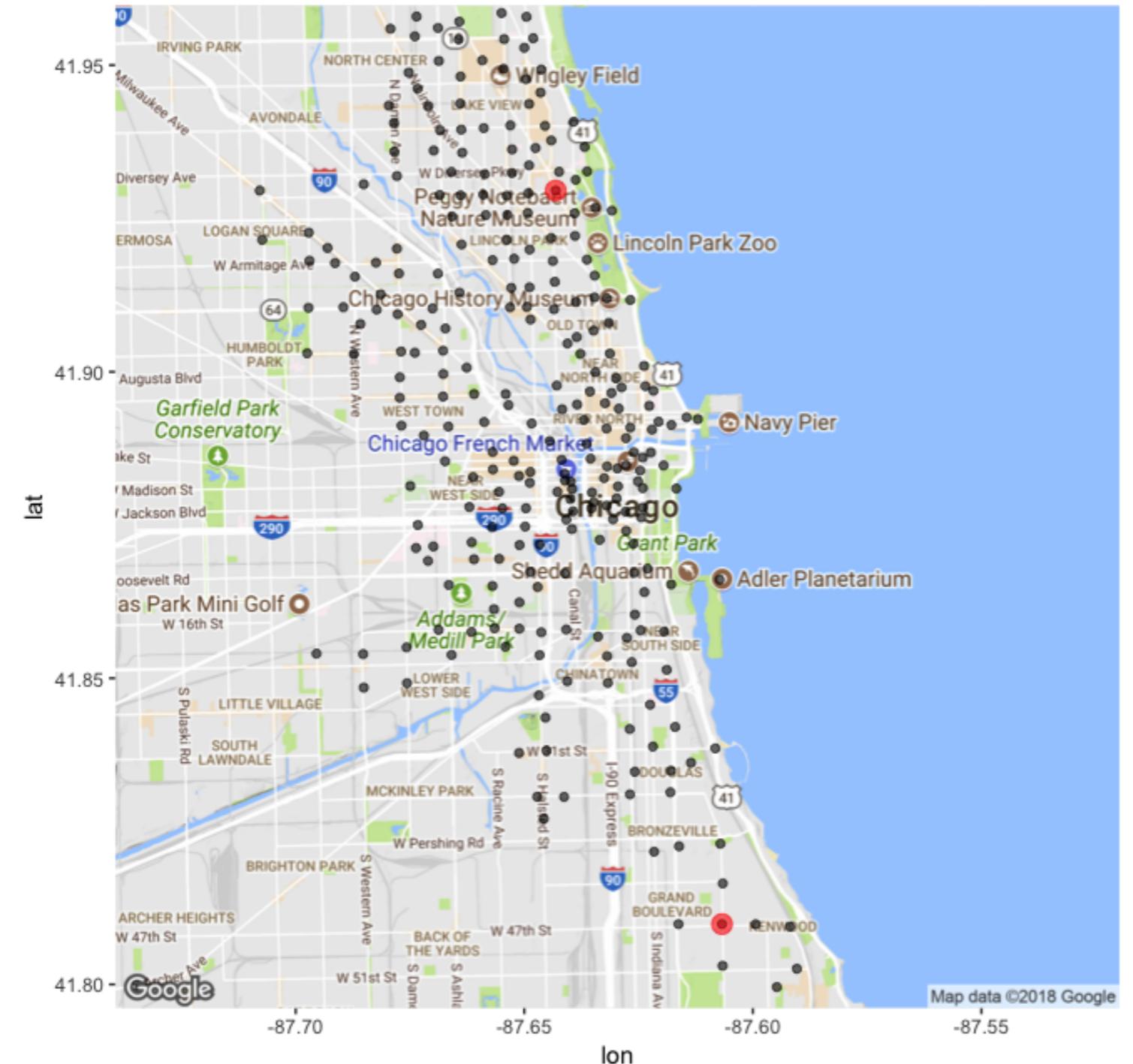
```
farthest_vertices(trip_g_simp)
```

```
$vertices  
+ 2/300 vertices, named, from 20dcfff:  
[1] 336 340
```

```
$distance  
[1] 5
```

```
get_diameter(trip_g_simp)
```

```
+ 4/300 vertices, named, from 20dcfff:  
[1] 336 267 76 340
```



Geographic distance

```
library(geosphere)
# Get the to stations coordinates
st_to <- bike_dat %>%
  filter(from_station_id == 336) %>%
  sample_n(1) %>%
  select(from_longitude, from_latitude)
# Get the from stations coordinates
st_from <- bike_dat %>%
  filter(from_station_id == 340) %>%
  sample_n(1) %>%
  select(from_longitude, from_latitude)
# find the geographic distance
farthest_dist <- distm(st_from, st_to, fun = distHaversine)
farthest_dist
```

```
[1, ] 13660.66
```

Geographic distance

```
bike_dist <- function(station_1, station_2, divy_bike_df){  
  st1 <- divy_bike_df %>%  
    filter(from_station_id == station_1) %>%  
    sample_n(1) %>%  
    select(from_longitude, from_latitude)  
  st2 <- divy_bike_df %>%  
    filter(from_station_id == station_2) %>%  
    sample_n(1) %>%  
    select(from_longitude, from_latitude)  
  
  farthest_dist <- distm(st1, st2, fun = distHaversine)  
  return(farthest_dist)  
}
```

Let's practice!

CASE STUDIES: NETWORK ANALYSIS IN R

Connectivity

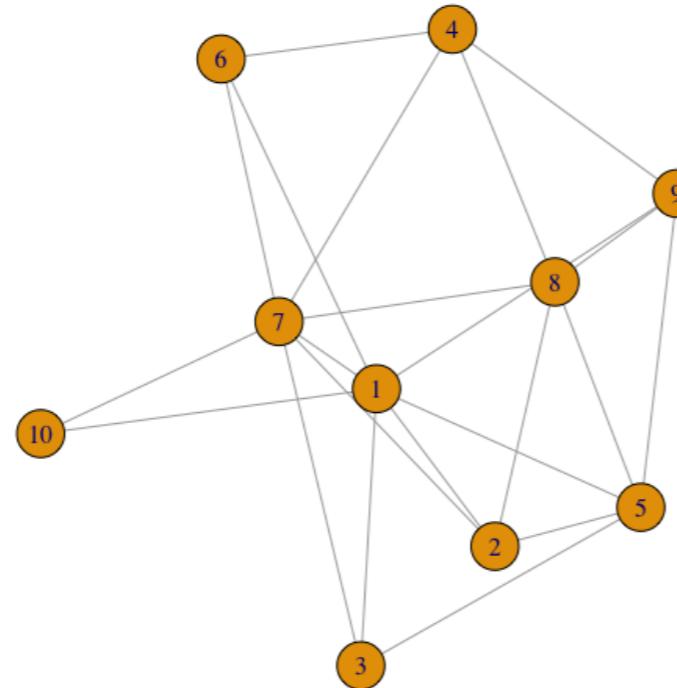
CASE STUDIES: NETWORK ANALYSIS IN R



Edmund Hart
Instructor

Measuring connectivity

```
rand_g <- erdos.renyi.game(10, 0.4, "gnp", directed = FALSE)  
plot(rand_g)
```



Measuring connectivity

```
rand_g <- erdos.renyi.game(10, 0.4, "gnp", directed = FALSE)  
vertex_connectivity(rand_g)
```

```
2
```

```
edge_connectivity(rand_g)
```

```
2
```

Minimum number of cuts

```
min_cut(rand_g, value.only = FALSE)
```

```
$value
[1] 2

$cut
+ 2/18 edges from 17a8fad:
[1] 10--7 10--1

$partition1
+ 1/10 vertex, from 17a8fad:
[1] 10

$partition2
+ 9/10 vertices, from 17a8fad:
[1] 1 2 3 4 5 6 7 8 9
```

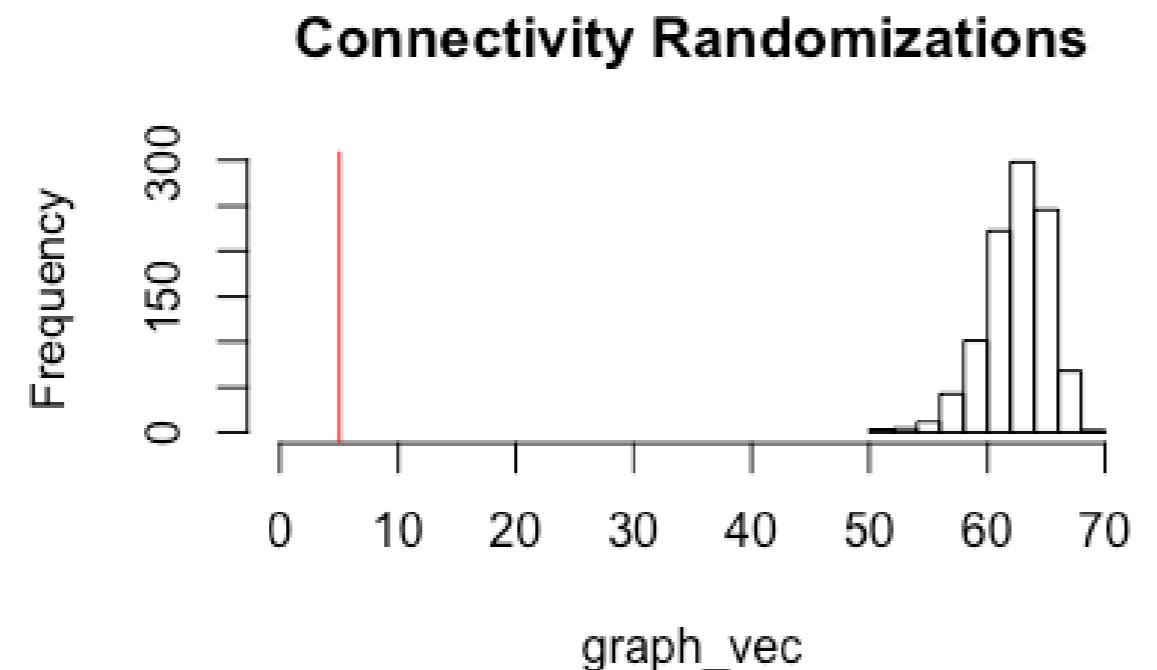
Connectivity randomizations

```
# Get parameters to simulate graph
nv <- gorder(trip_g_ud)
ed <- edge_density(trip_g_ud)

# Empty vector to store output
graph_vec <- rep(NA, 1000)
# Generate 1000 random graphs and find the edge connectivity
for(i in 1:1000) {
  w1 <- erdos.renyi.game(nv, ed, "gnp", directed = TRUE)
  graph_vec[i] <- edge_connectivity(w1)
}
```

Connectivity randomizations

```
# Find actual connectivity  
econn <- edge_connectivity(trip_g_ud)  
hist(graph_vec, xlim = c(0, 140))  
abline(v = edge_connectivity(trip_g_ud))
```



Let's practice!

CASE STUDIES: NETWORK ANALYSIS IN R

Other packages for plotting graphs

CASE STUDIES: NETWORK ANALYSIS IN R



Edmund Hart

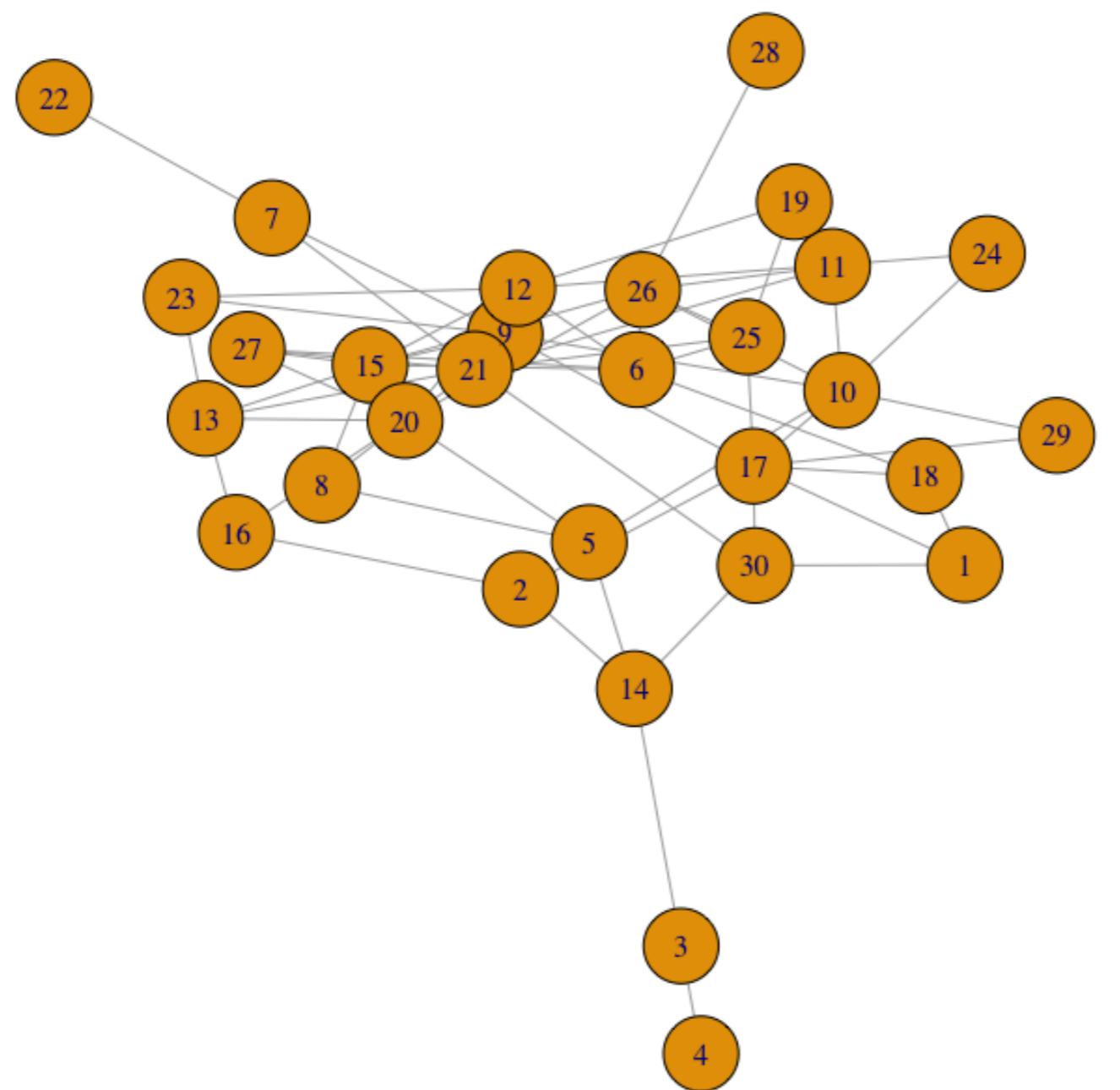
Instructor

Generating data to plot

```
library(ggnetwork)
library(igraph)
library(GGally)
library(intergraph)

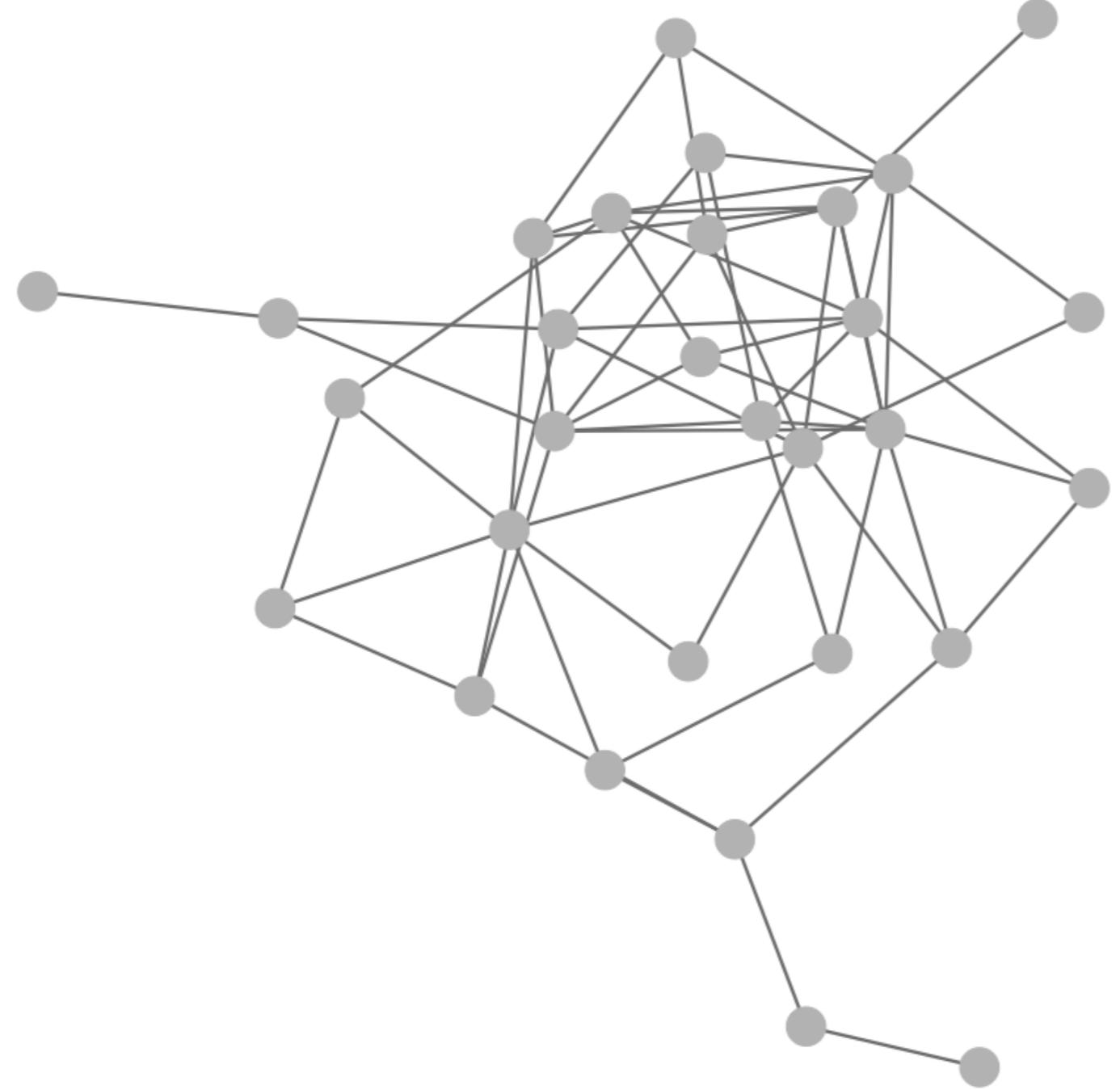
rand_g <- erdos.renyi.game(30, .15, "gnp", directed = F)
rand_g <- simplify(rand_g)

plot(rand_g)
```



Basic ggnet2

```
net <- asNetwork(rand_g)  
ggnet2(net)
```

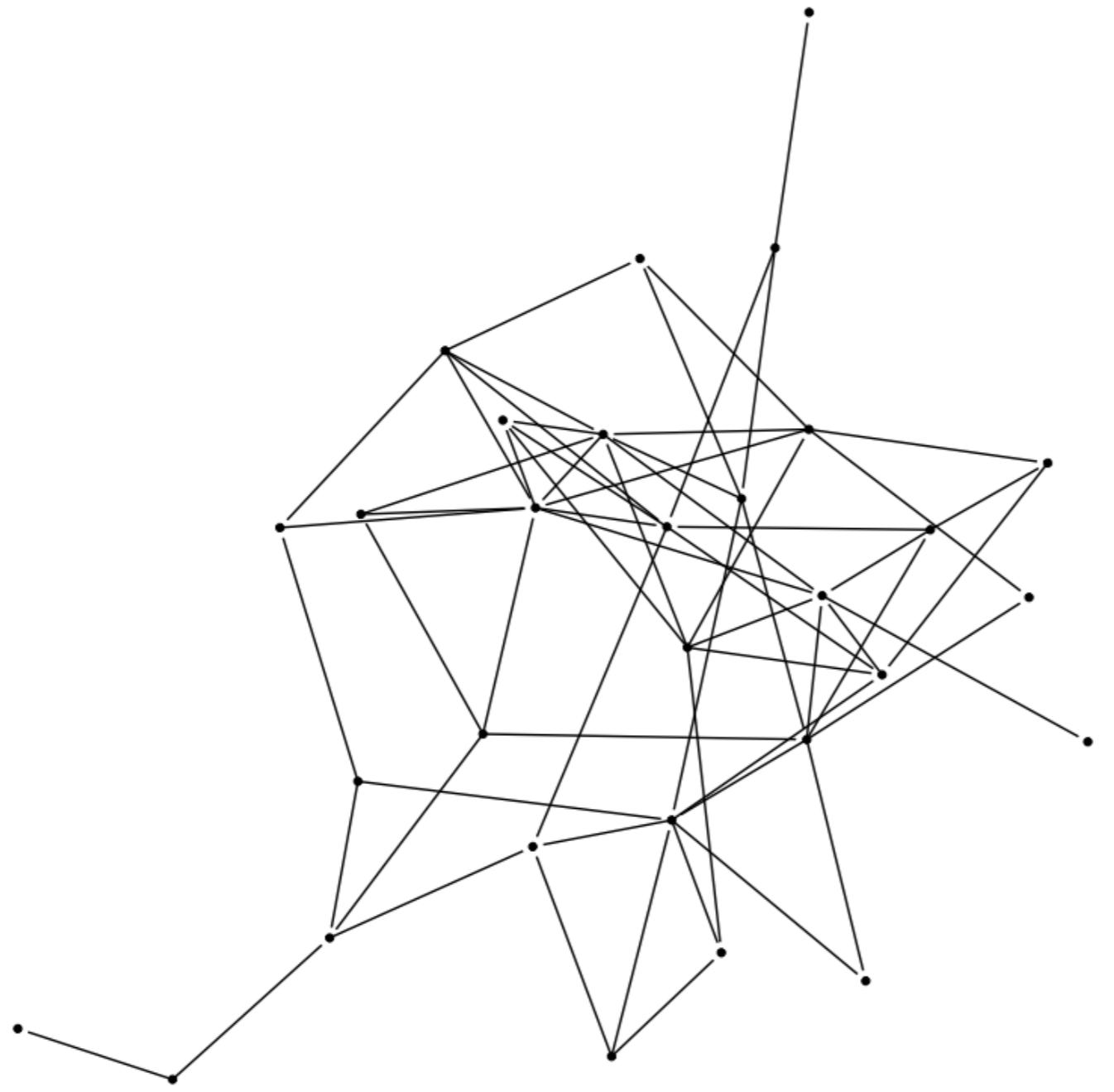


Basic ggnetwork

```
gn <- ggnetwork(rand_g)
g <- ggplot(gn, aes(x = x, y = y, xend = xend, yend = yend)) +
  geom_edges() +
  geom_nodes() +
  theme_blank()
head(gn)
```

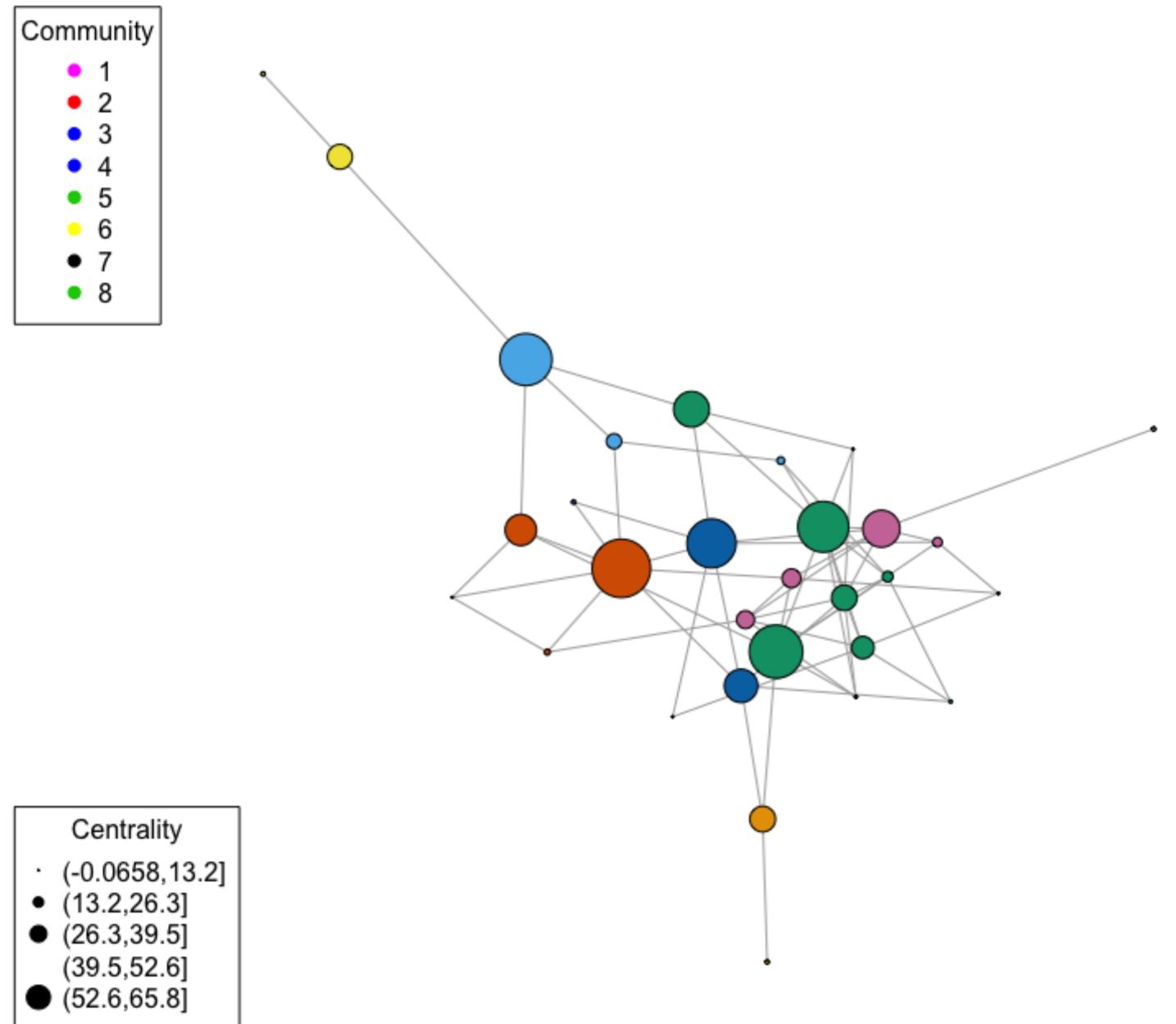
	x	y	na.x	vertex.names	xend	yend	na.y	
1	0.4729841	0.01697675	FALSE		1	0.4729841	0.01697675	NA
2	0.1883442	0.42284666	FALSE		2	0.1883442	0.42284666	NA
3	0.3485247	0.82865654	FALSE		3	0.3485247	0.82865654	NA
4	0.3905894	1.00000000	FALSE		4	0.3905894	1.00000000	NA

```
plot(g)
```



Plotting graphs with attributes

```
# Add attributes
V(rand_g)$cent <- betweenness(rand_g)
V(rand_g)$comm <- membership(cluster_walktrap(rand_g))
# Make plot
plot(rand_g, vertex.label = NA, margin = 0,
      vertex.color = V(rand_g)$comm,
      vertex.size = V(rand_g)$cent / 6)
# Add legend for community membership
legend('topleft', legend= sort(unique( V(rand_g)$comm)),
       col= sort(unique(V(rand_g)$comm)), pch = 19, title = "Community")
# Add cuts and then get quantiles for size legend
cc <- cut(V(rand_g)$cent, 5)
scaled <- quantile(V(rand_g)$cent, seq(0.3, 0.9, length = 5)) / 25
# Add size legend for centrality
legend('bottomleft', legend= levels(cc),
       pt.cex = scaled, pch = 19, title = "Centrality")
```



ggnet2 plot with attributes

```
net <- asNetwork(rand_g)

ggnet2(net,
       node.size = "cent",
       node.color = "comm",
       edge.size = 0.8,
       color.legend = "Community Membership",
       color.palette = "Spectral",
       edge.color = c("color", "gray88"),
       size.cut = TRUE,
       size.legend = "Centrality")
```

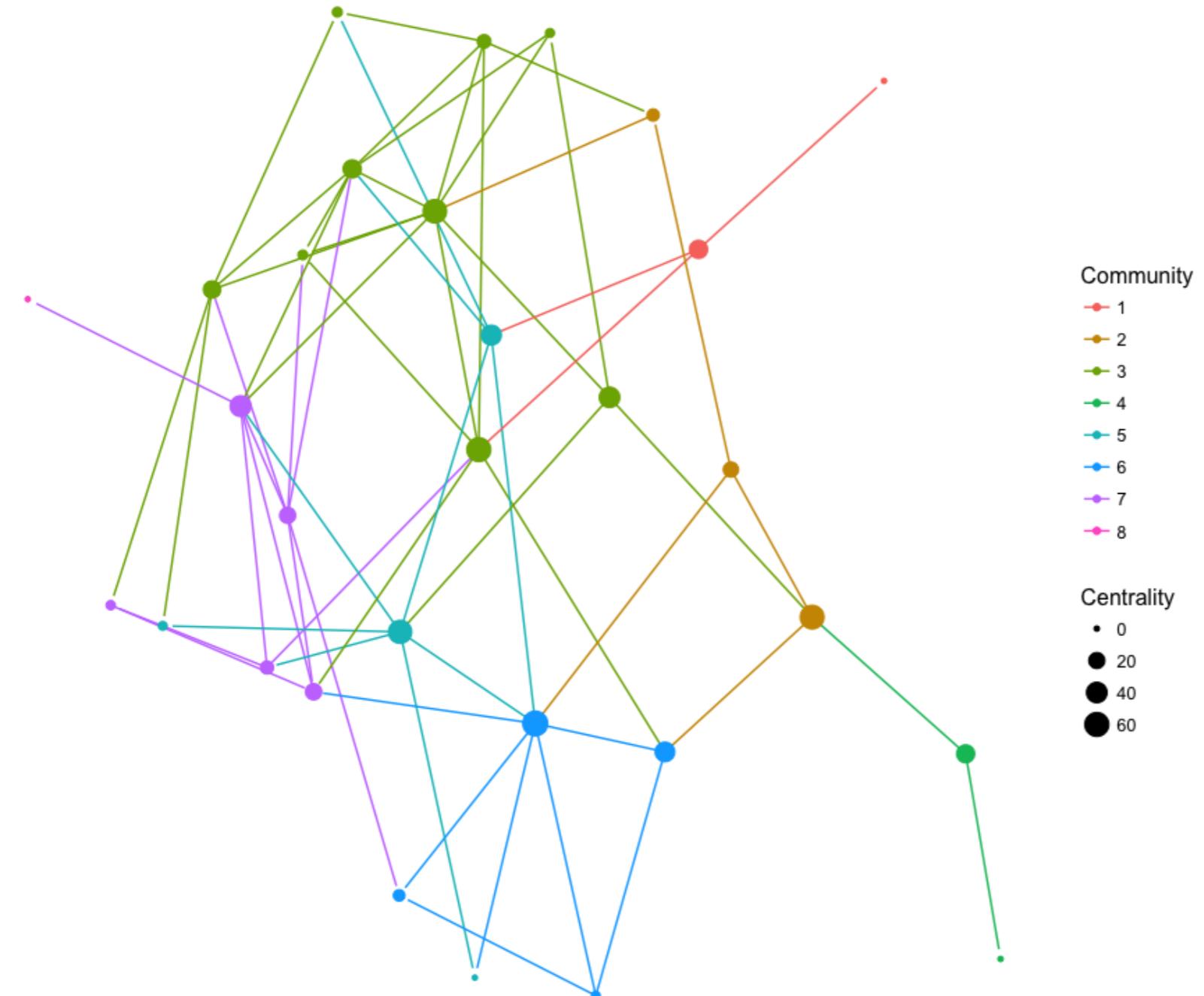


ggnetwork plot with attributes

```
gn <- ggnetwork(rand_g)

g <- ggplot(gn, aes(x = x, y = y, xend = xend, yend = yend)) +
  geom_edges(aes(color = as.factor(comm))) +
  geom_nodes(aes(color = as.factor(comm), size = cent)) +
  theme_blank() +
  guides(
    color = guide_legend(title = "Community"),
    size = guide_legend(title = "Centrality"))

plot(g)
```



Let's practice!

CASE STUDIES: NETWORK ANALYSIS IN R

Interactive visualizations

CASE STUDIES: NETWORK ANALYSIS IN R



Edmund Hart
Instructor

Generating some data

```
library(igraph)
library(ggnetwork)
library(ggiraph)
library(htmlwidgets)
library(networkD3)

# Create random graph
rand_g <- erdos.renyi.game(30, 0.12, "gnp", directed = FALSE)
rand_g <- simplify(rand_g)

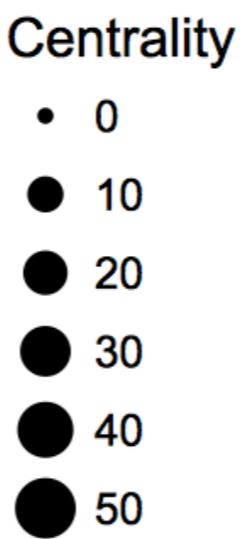
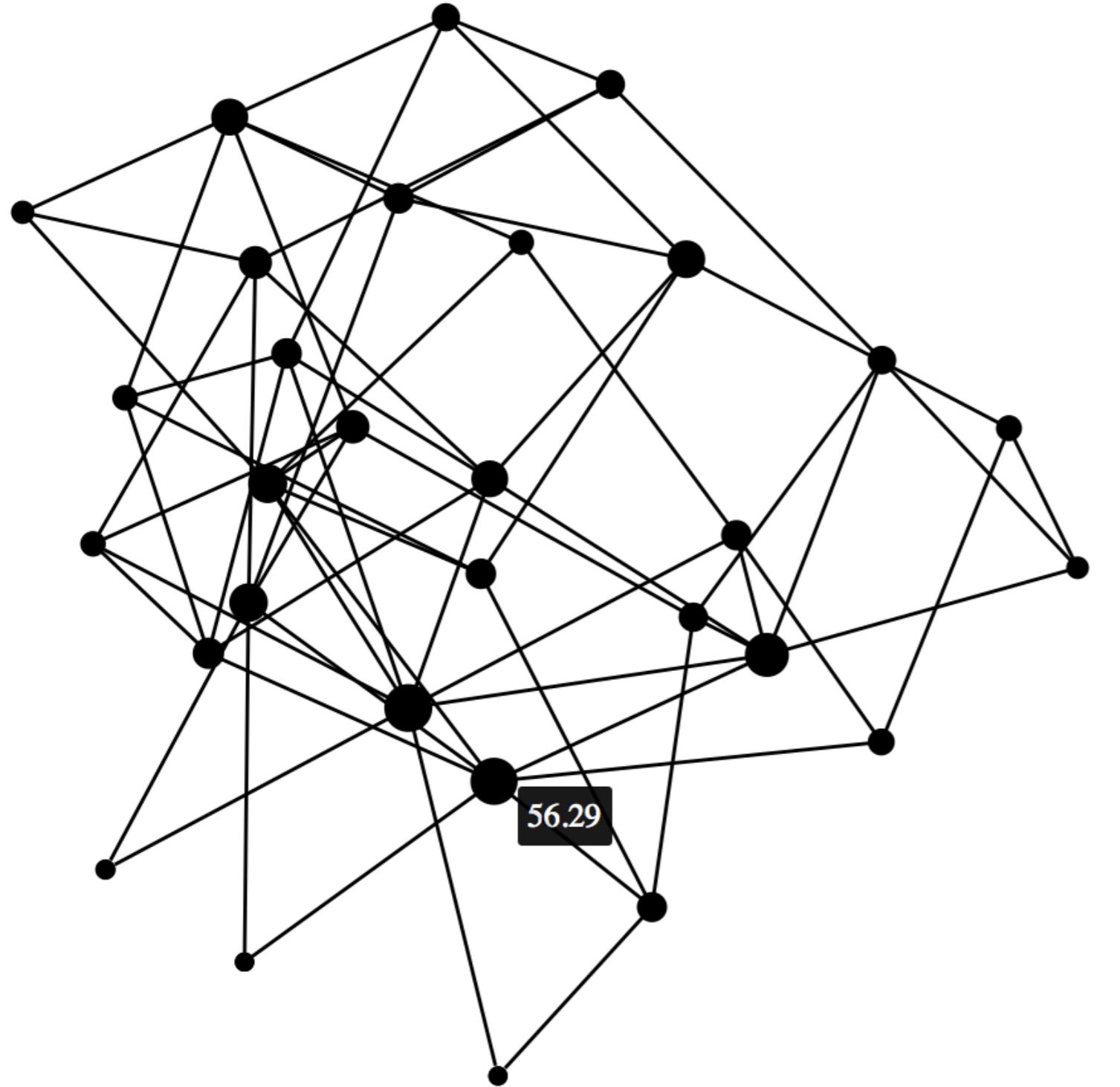
V(rand_g)$cent <- betweenness(rand_g)
```

Interactive plots with ggiraph

```
# Plot graph with ggplot2 and ggnetwork
g <- ggplot(ggnetwork(rand_g),
             aes(x = x, y = y, xend = xend, yend = yend)) +
  geom_edges(color = "black") +
  geom_nodes(aes(size = cent)) + theme_blank() +
  guides(size = guide_legend(title = "Centrality"))

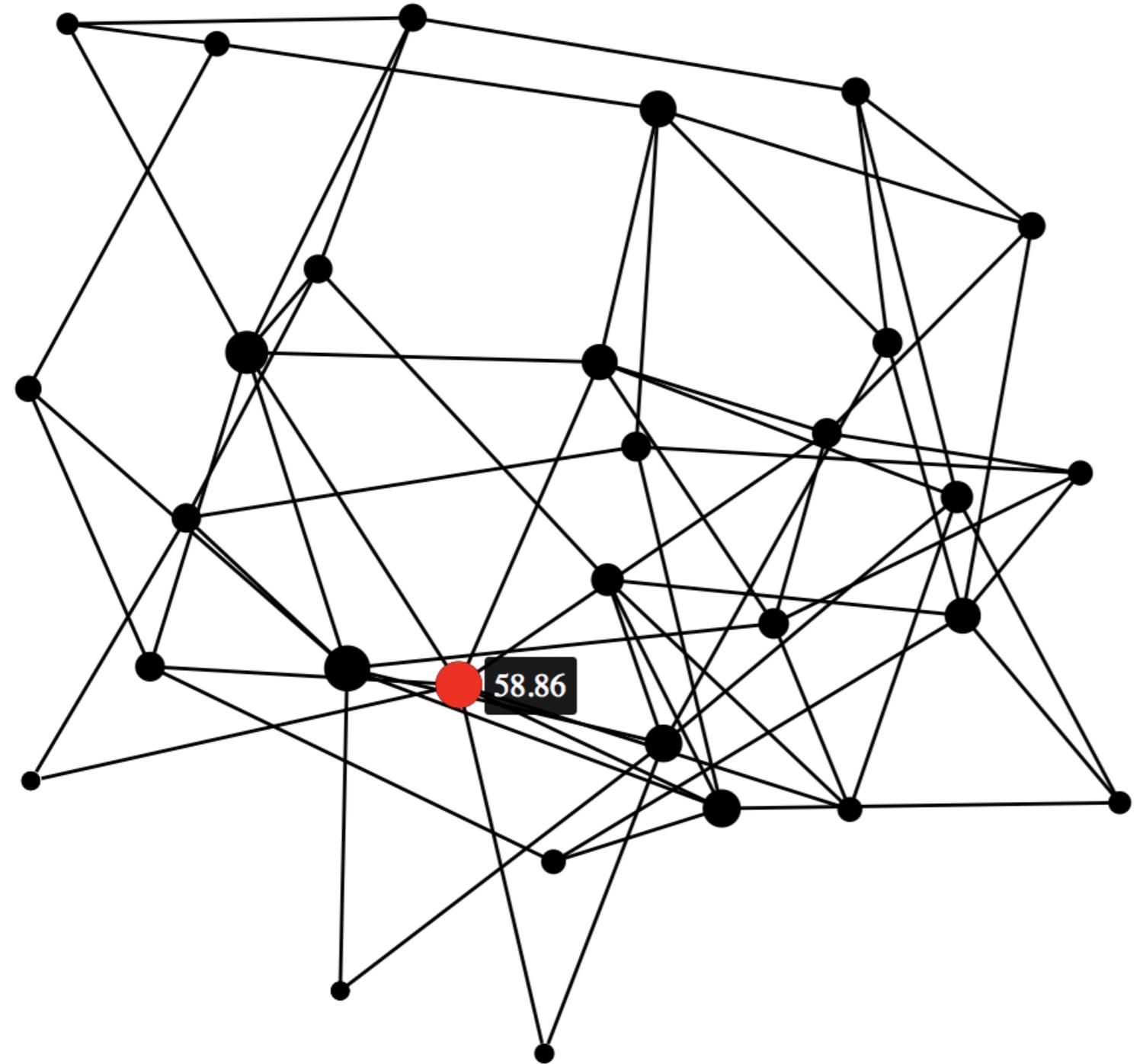
# Create ggiraph object
my_gg <- g + geom_point_interactive(aes(tooltip = round(cent, 2)),
                                      size = 2)

# Display ggiraph object
ggiraph(code = print(my_gg))
```



ggiraph customization

```
my_gg <- g + geom_point_interactive(aes(tooltip = round(cent, 2),  
                                         data_id = round(cent, 2)),  
                                         size = 2)  
  
hover_css = "cursor:pointer;fill:red;stroke:red;r:5pt"  
  
ggiraph(code = print(my_gg),  
        hover_css = hover_css,  
        tooltip_offx = 10,  
        tooltip_offy = -10)
```

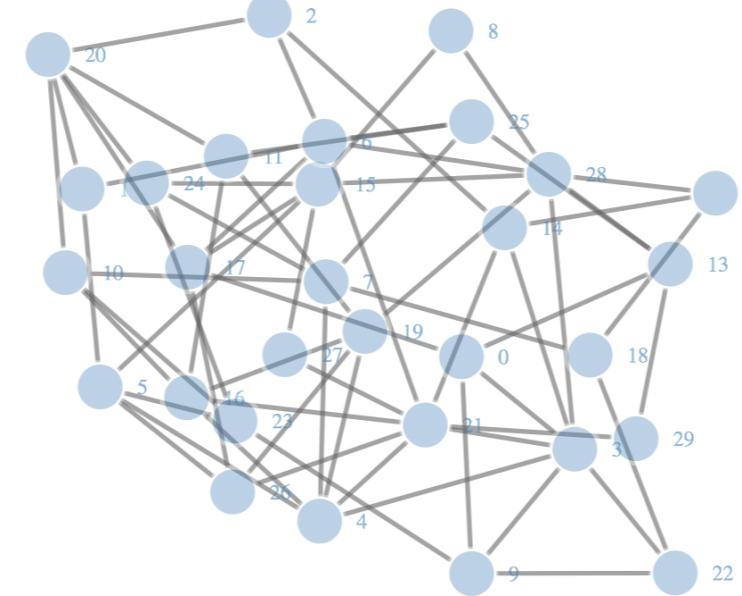


cent

- 0
- 10
- 20
- 30
- 40
- 50

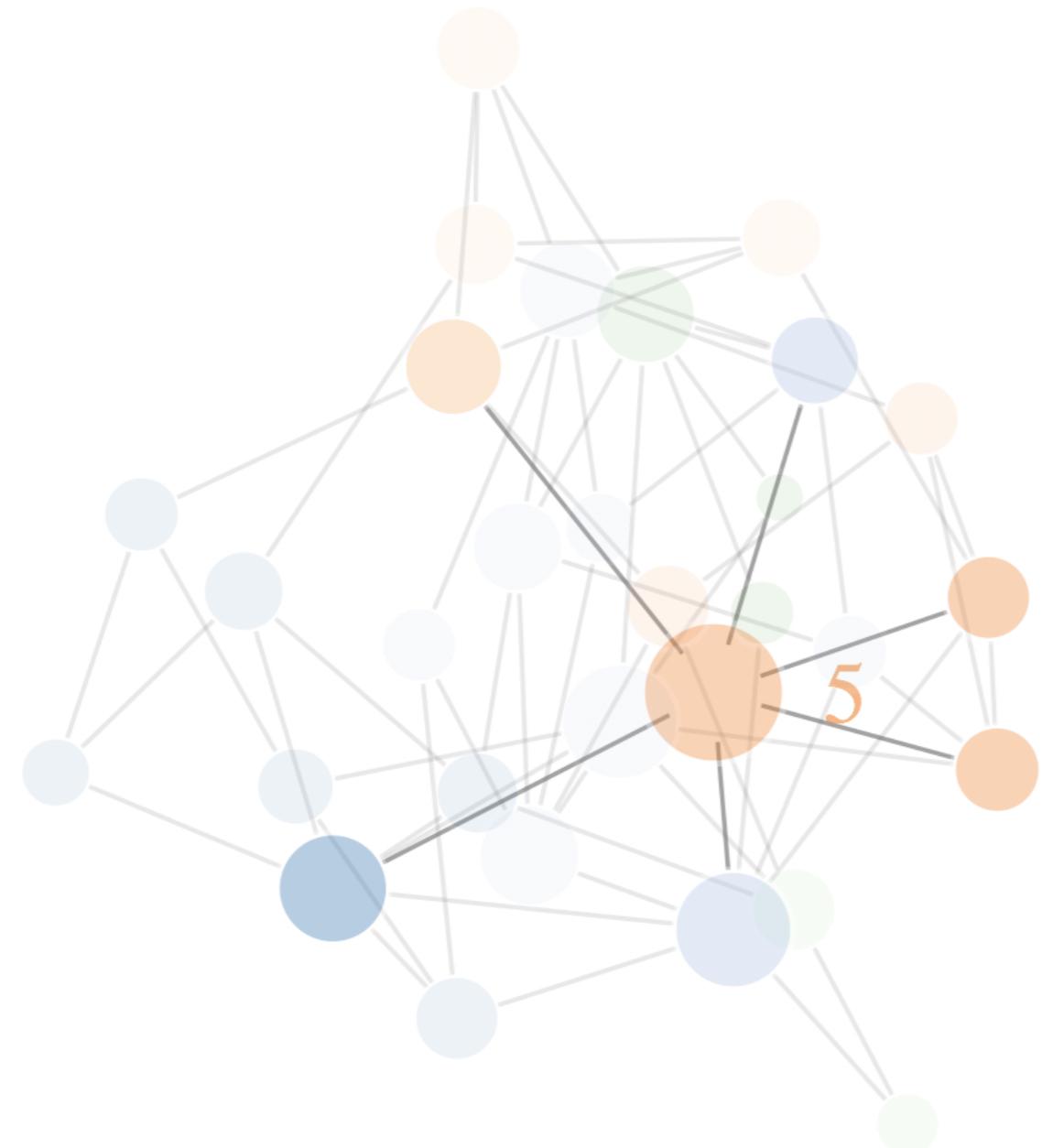
Plotting with networkD3

```
# Convert the igraph object  
nd3 <- igraph_to_networkD3(rand_g)  
  
# Create a simple network  
simpleNetwork(nd3$links)
```



More complex networkD3

```
# Add attributes, group is community, and cent is centrality.  
nd3$nodes$group = V(rand_g)$comm  
nd3$nodes$cent = V(rand_g)$cent  
  
# Plot the graph  
forceNetwork(Links = nd3$links,  
             Nodes = nd3$nodes,  
             Source = 'source',  
             Target = 'target',  
             NodeID = 'name',  
             Group = 'group',  
             Nodesize = 'cent',  
             legend = T,  
             fontSize = 20)
```



Let's practice!

CASE STUDIES: NETWORK ANALYSIS IN R

Alternative visualizations

CASE STUDIES: NETWORK ANALYSIS IN R



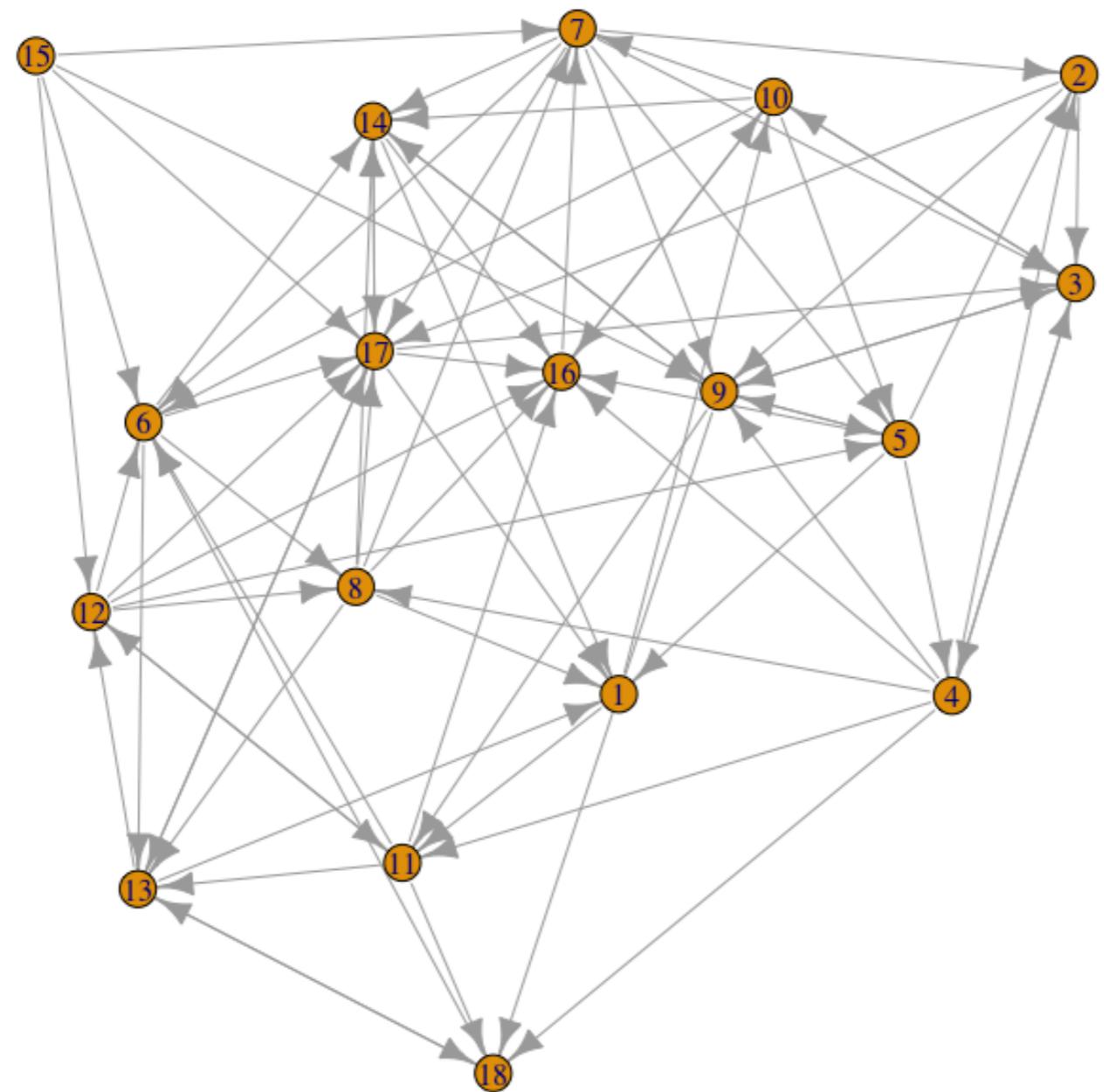
Edmund Hart
Instructor

Introduction to hive plots

```
library(HiveR)
library(igraph)

# Create random graph
rand_g <- erdos.renyi.game(18, 0.3, "gnp", directed = TRUE)

# Plot random graph
plot(rand_g, vertex.size = 7)
```



Introduction to hive plots

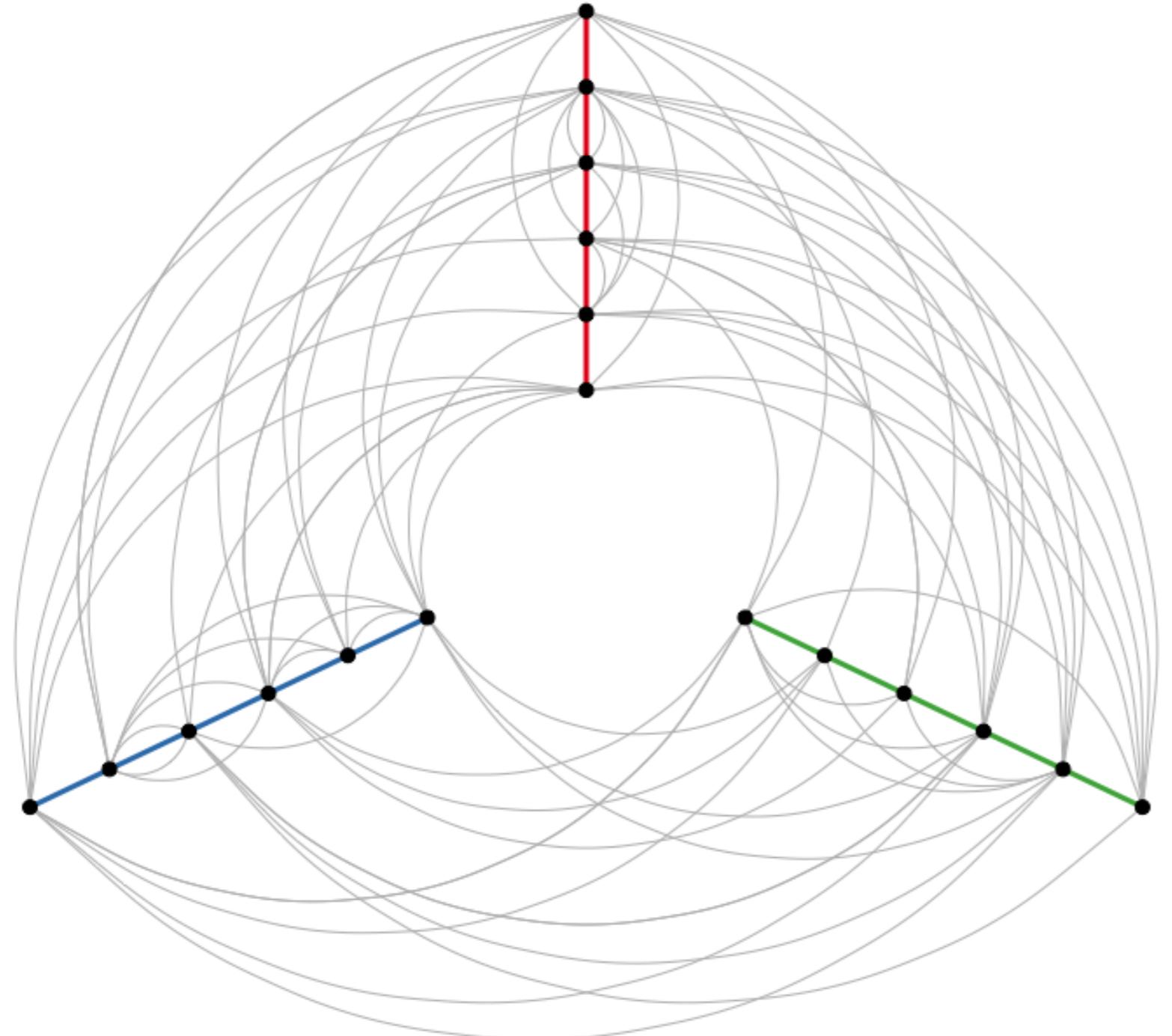
```
# Convert to dataframe for hive plots and add weights
rand_g_df <- as.data.frame(get.edgelist(rand_g))
rand_g_df$weight <- 1
# Convert to hive object
rand_hive <- edge2HPD(edge_df = rand_g_df)
# Set the axis and the radius of each node
rand_hive$nodes$axis <- sort(rep(1:3, 6))
rand_hive$nodes$radius <- as.double(rep(1:6, 3))
```

Introduction to hive plots

```
# See how nodes are modified  
rand_hive$nodes
```

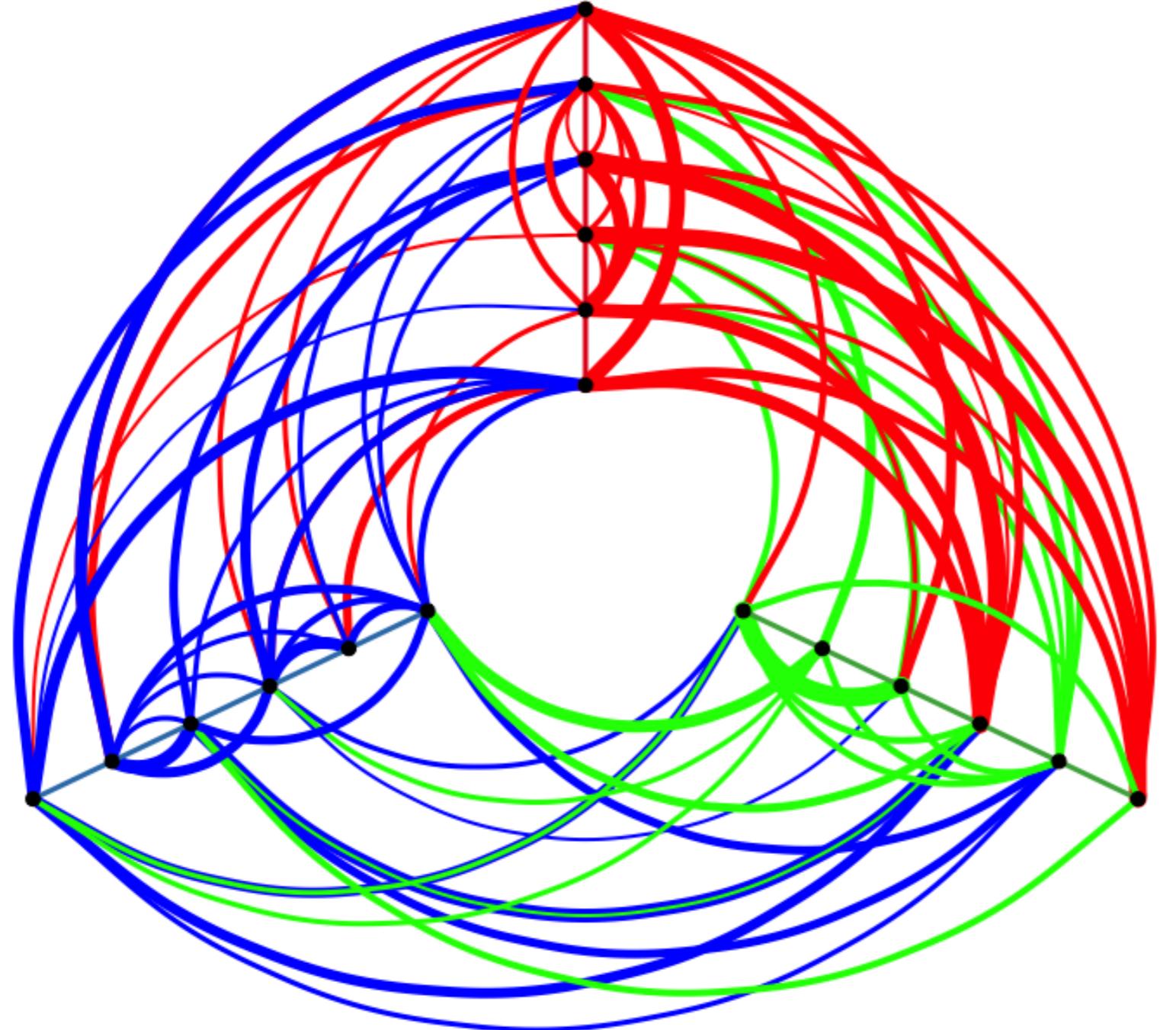
```
id lab axis radius size color  
1 2 1 1 1 black  
2 8 1 2 1 black  
3 9 1 3 1 black  
4 3 1 4 1 black  
5 4 1 5 1 black  
6 7 1 6 1 black  
7 11 2 1 1 black  
8 14 2 2 1 black  
9 18 2 3 1 black
```

```
# See hive plot  
plotHive(rand_hive, method = "abs", bkgnd = "white")
```



Modifying hive plots

```
# Setting location of each node
rand_hive$nodes$axis <- sort(rep(1:3, 6))
rand_hive$nodes$radius <- as.double(rep(1:6, 3))
# Add weights to each edge
rand_hive$edges$weight <- as.double(
  rpois(length(rand_hive$edges$weight), 5)
)
# Add color based on edge origination
rand_hive$edges$color[rand_hive$edges$id1 %in% 1:6] <- 'red'
rand_hive$edges$color[rand_hive$edges$id1 %in% 7:12] <- 'blue'
rand_hive$edges$color[rand_hive$edges$id1 %in% 13:18] <- 'green'
# Plot
plotHive(rand_hive, method = "abs", bkgnd = "white")
```

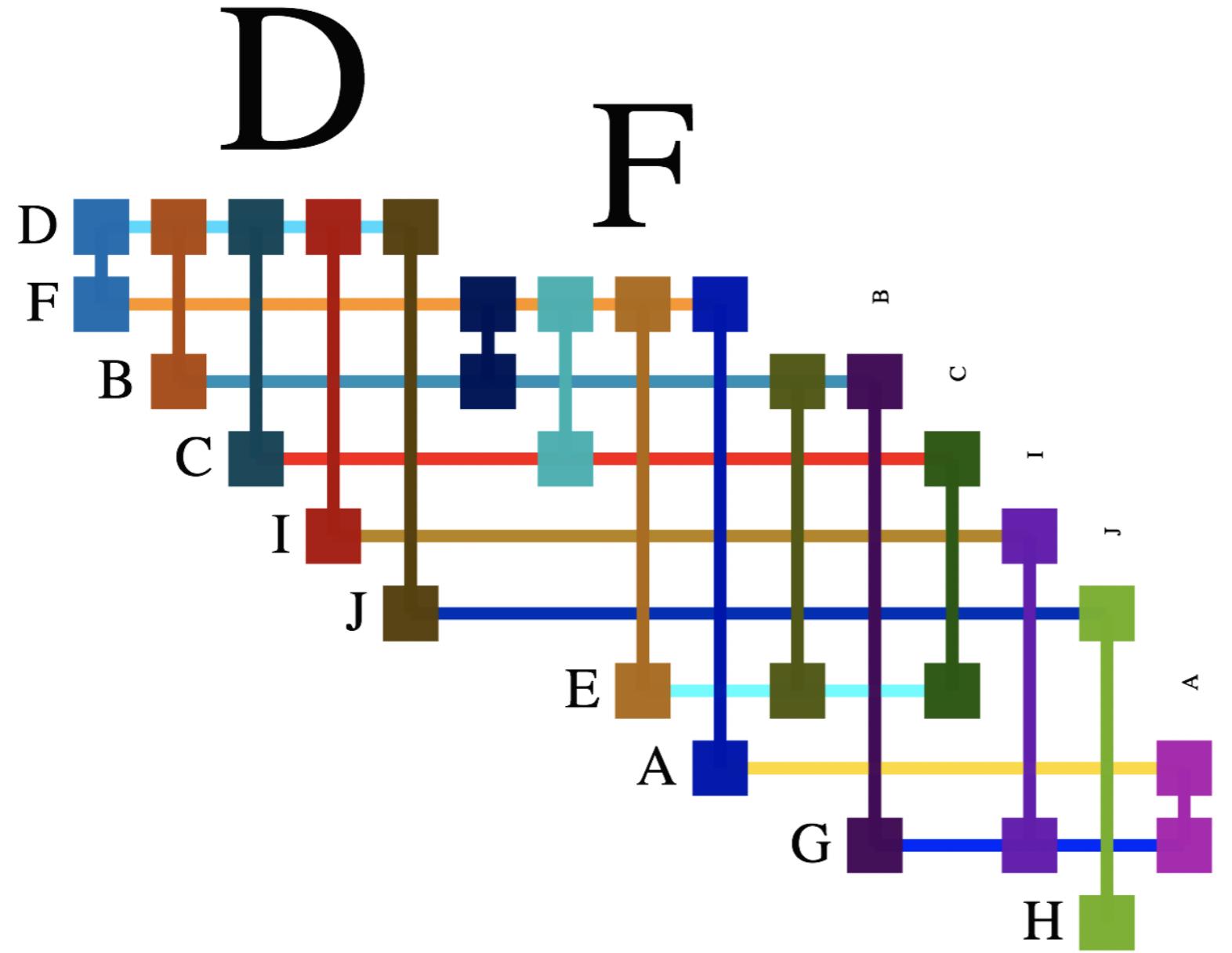


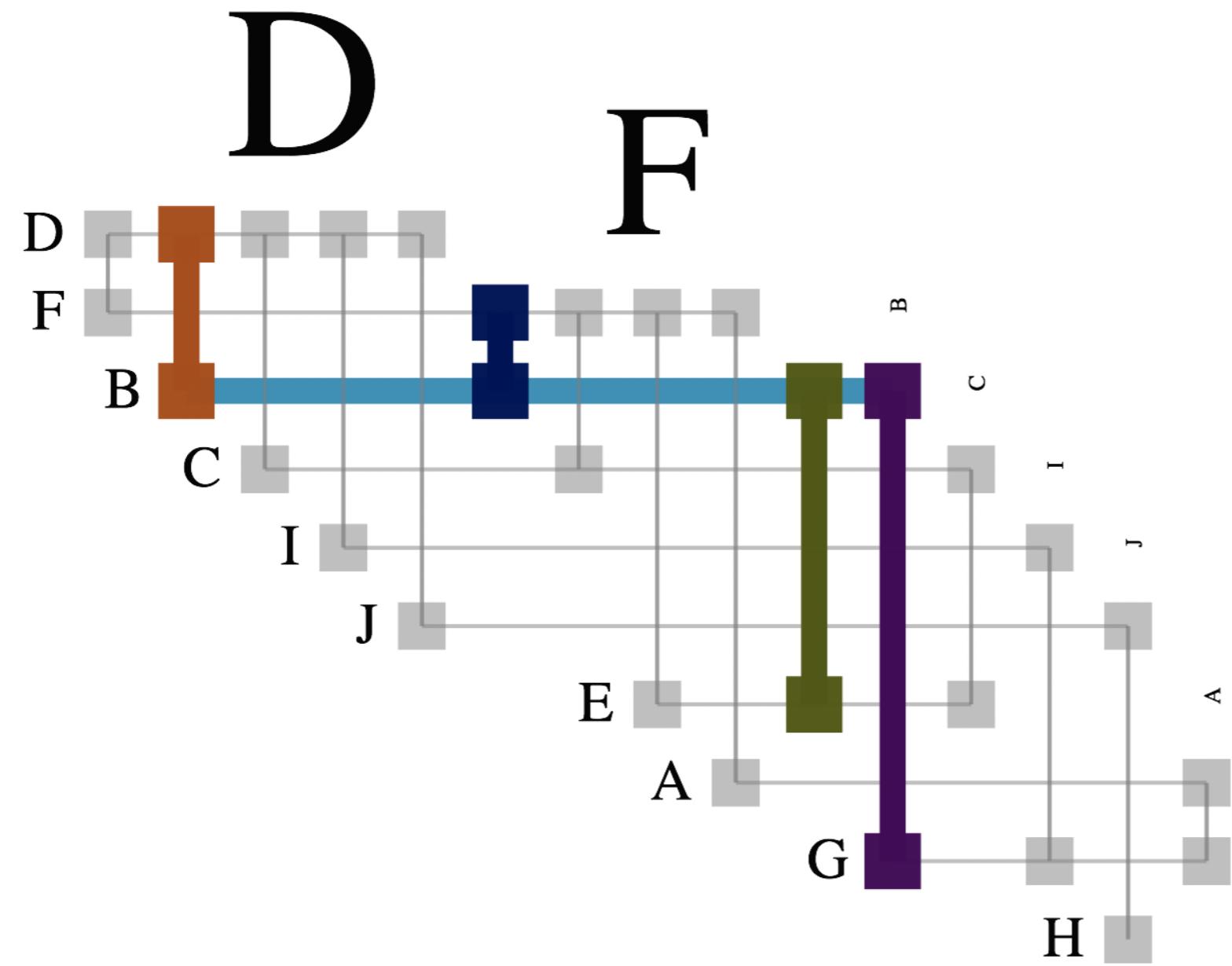
Biofabric plots

```
# Create random graph
rand_g <- erdos.renyi.game(10, 0.3, "gnp", directed = FALSE)
rand_g <- simplify(rand_g)

# Add names to vertices
V(rand_g)$name <- LETTERS[1:length(V(rand_g))]

# Create biofabric plot
biofbc <- bioFabric(rand_g)
bioFabric_htmlwidget(biofbc)
```





Let's practice!

CASE STUDIES: NETWORK ANALYSIS IN R