

Jack Reilly

CRASH COURSE

---

# Social Networks: Description and Visualization

---

FEBRUARY 2023

# Agenda

---

**Why do we study social networks?**

---

**What are social networks?**

---

**How do we visualize networks in R?**

---

**Where can I go for more?**

# Learning outcomes

- When we're done, you should be able to:
  - **Describe social networks** using formal language
  - **Construct network data** for analysis and visualization
  - Use the **iGraph** library in R to draw and manipulate basic networks
    - Know where to go for other options

# Part One

---

## Why do we study social networks?

---

# A contemporary story: Polarization

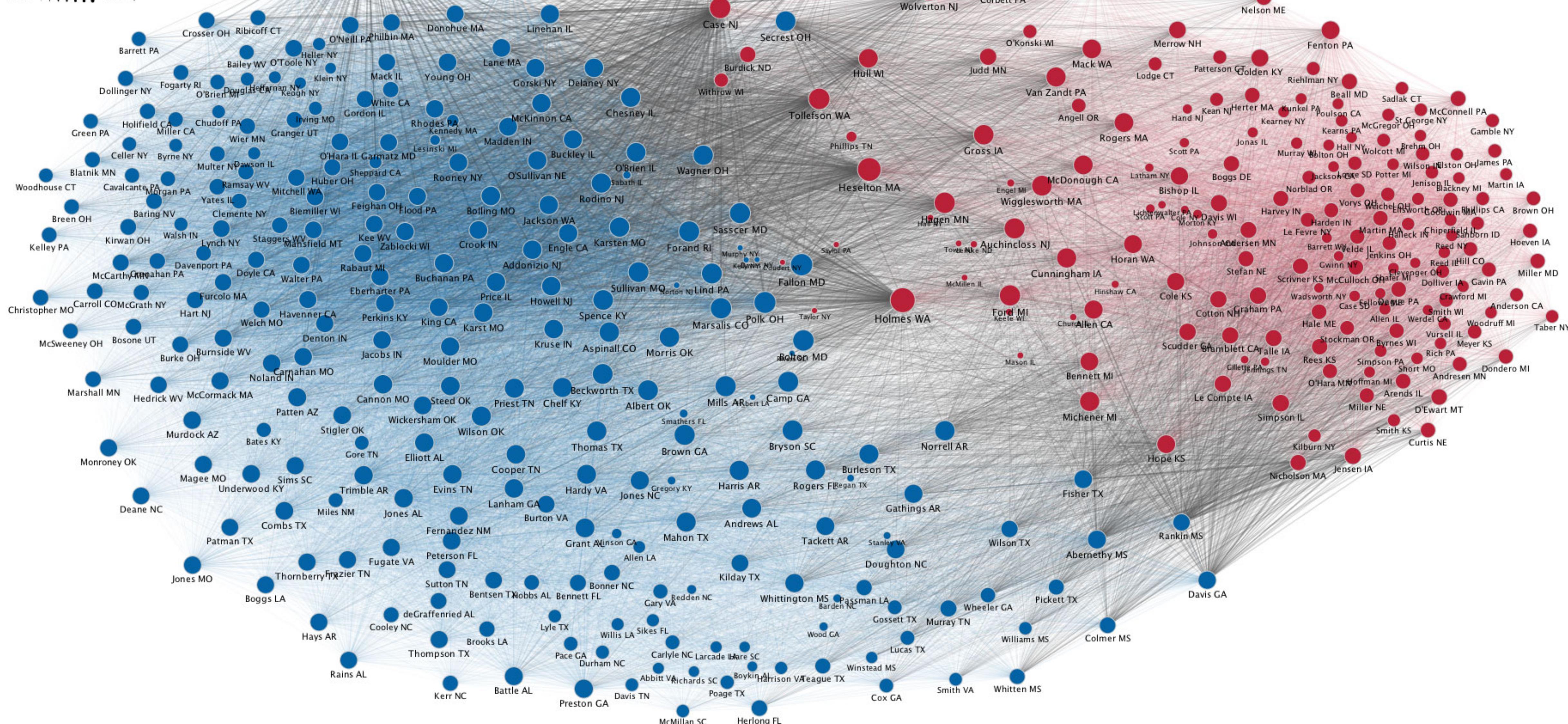
- Specifically, polarization in the United States House of Representatives
  - Based on roll call data, we can see how often different pairs (or “dyads”) of legislators agree
  - For all pairs of legislators:
    - Votes in common (both “aye” or both “nay”) increased the agreement score
    - Votes not in common (one “aye” and one “nay”) decreased the agreement score
  - What does it look like? And does it change?

# Year: 1949

D-D    D-R    R-R

Degree 1 o 400

Few ||| Many

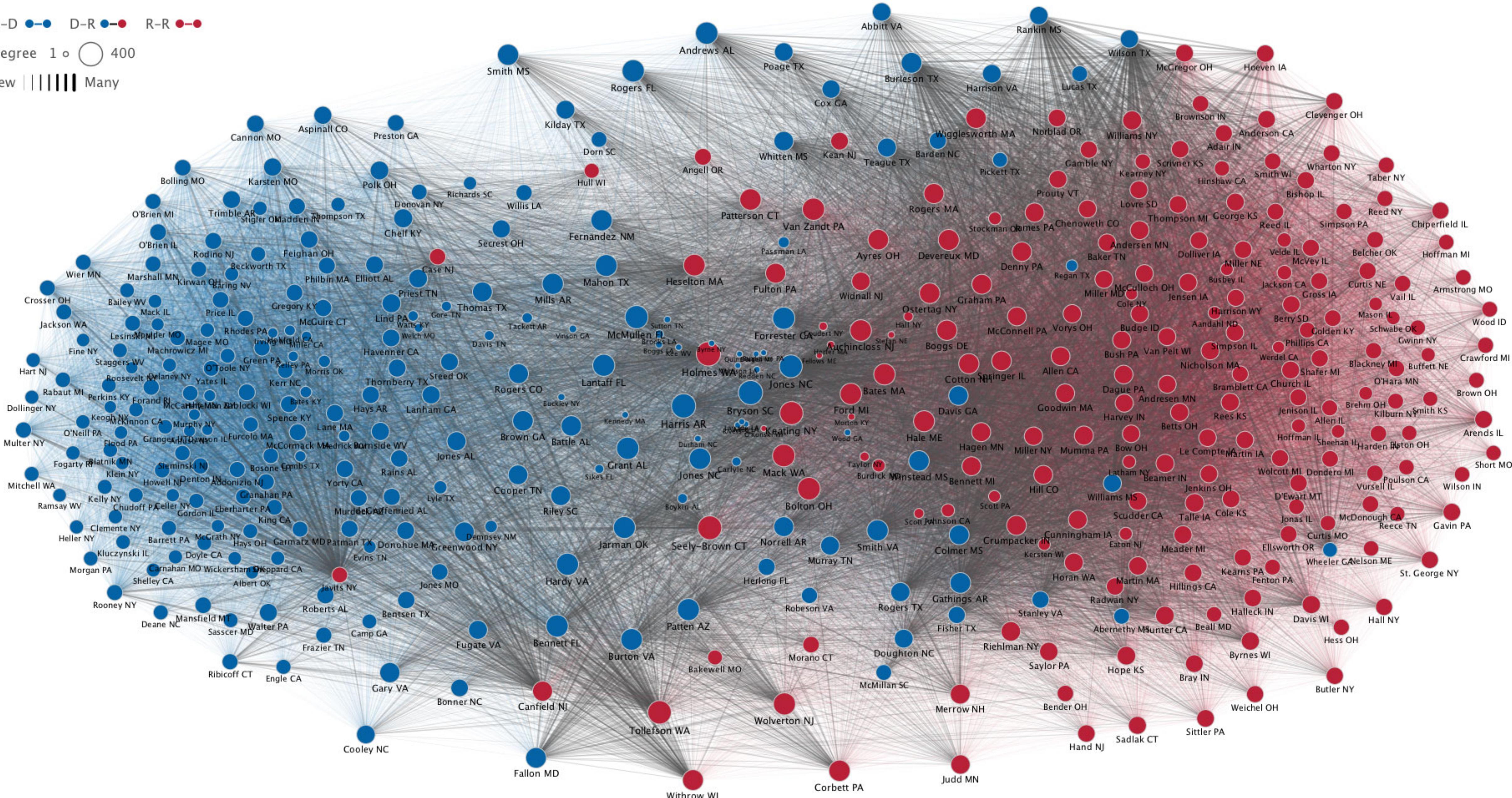


# Year: 1951

D-D    D-R    R-R

Degree 1 o 400

Few ||| Many

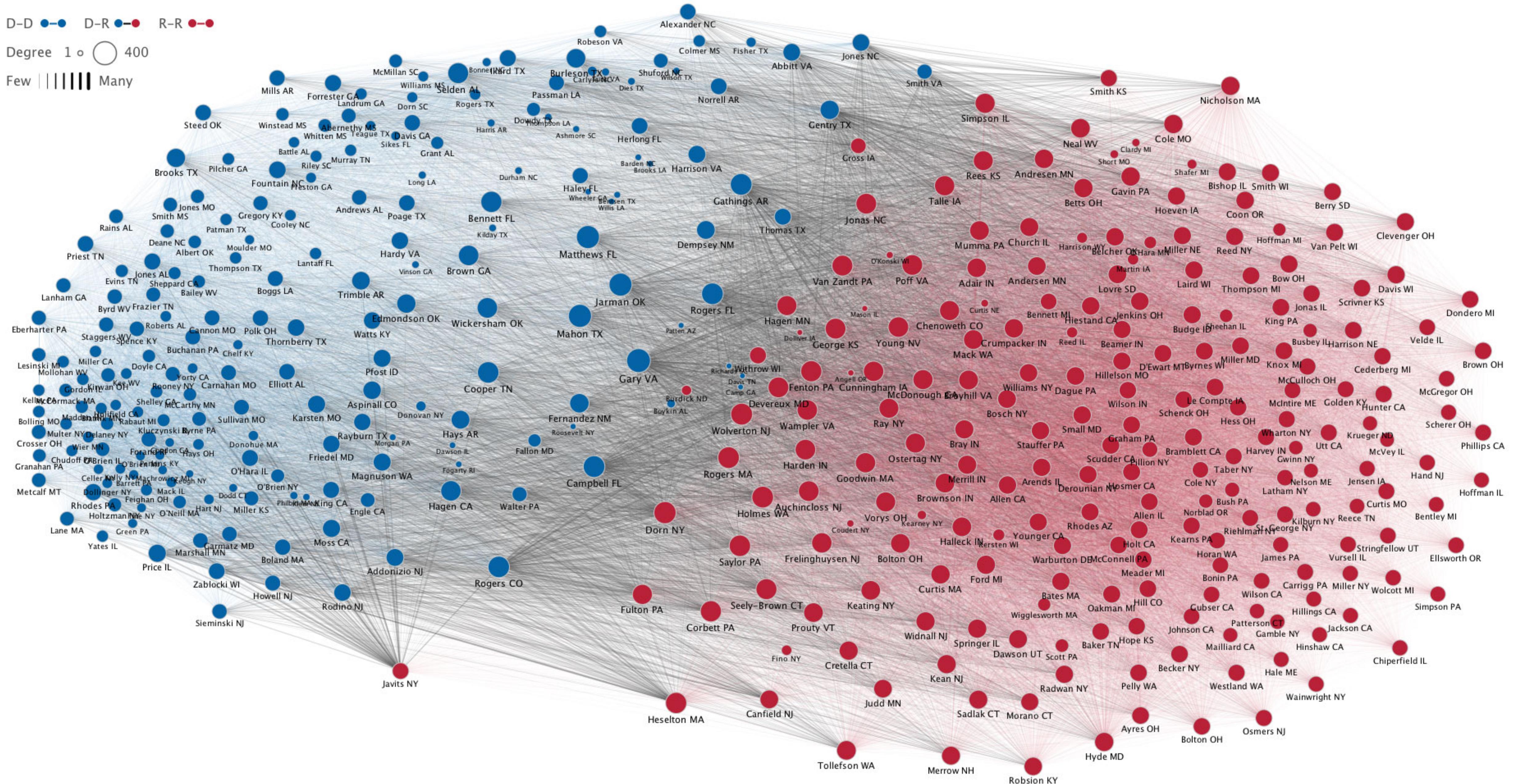


# Year: 1953

D-D •— D-R •— R-R •—

Degree 1 o 400

Few ||| Many

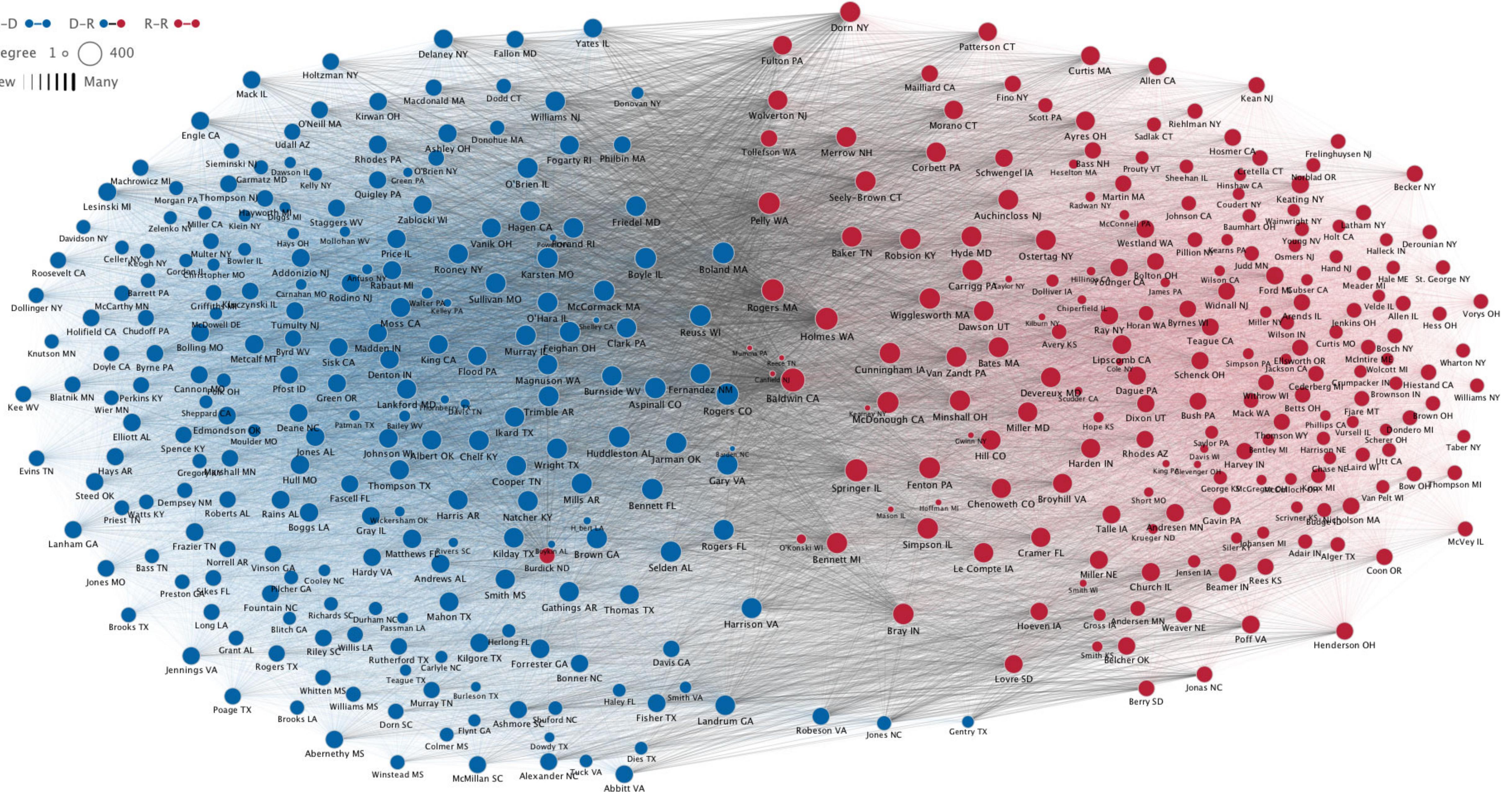


Year: 1955

D-D  D-R  R-R 

Degree  $1^\circ$  400

Few | | | | | Many

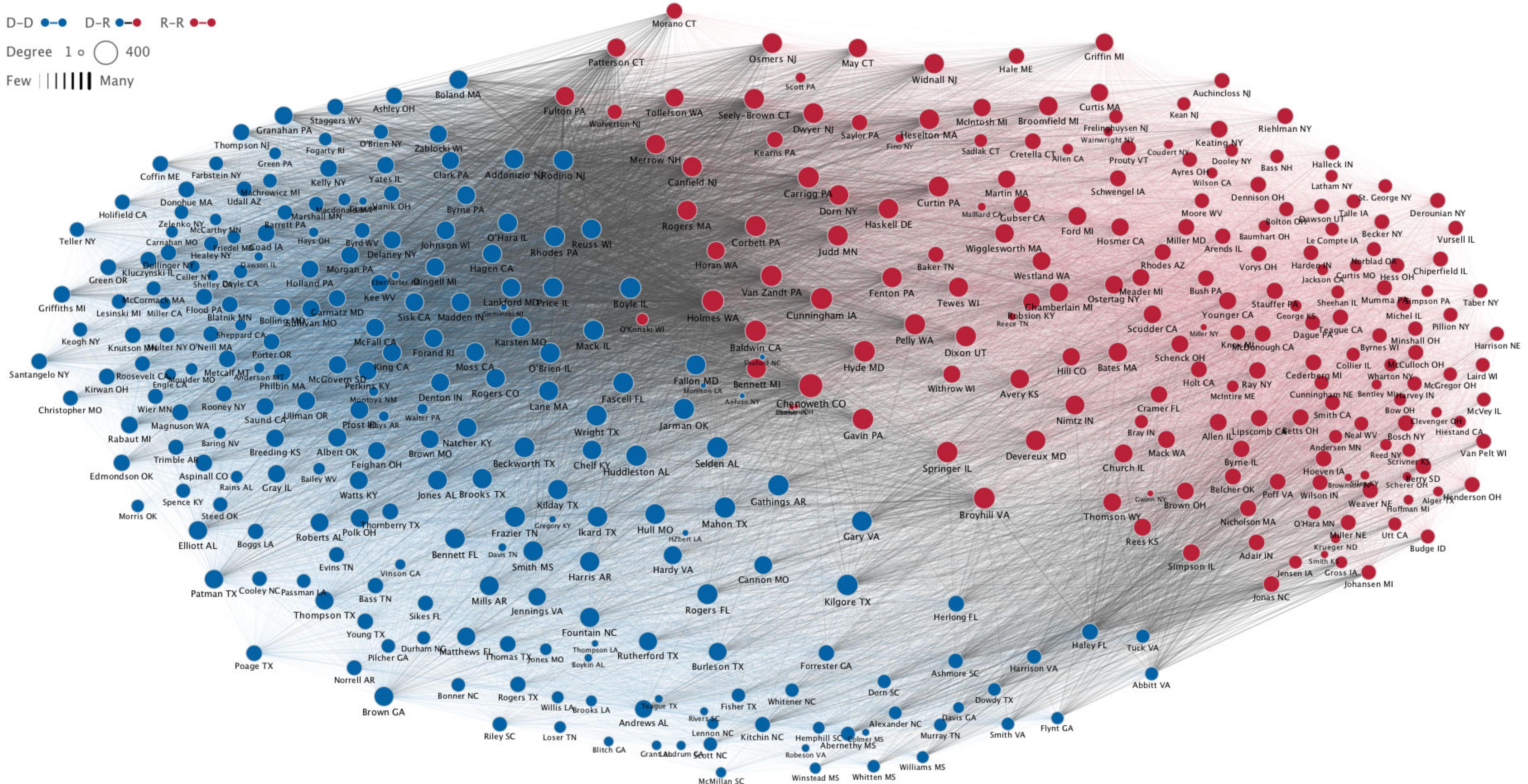


# Year: 1957

D-D    D-R    R-R

Degree 1 o 400

Few ||| Many

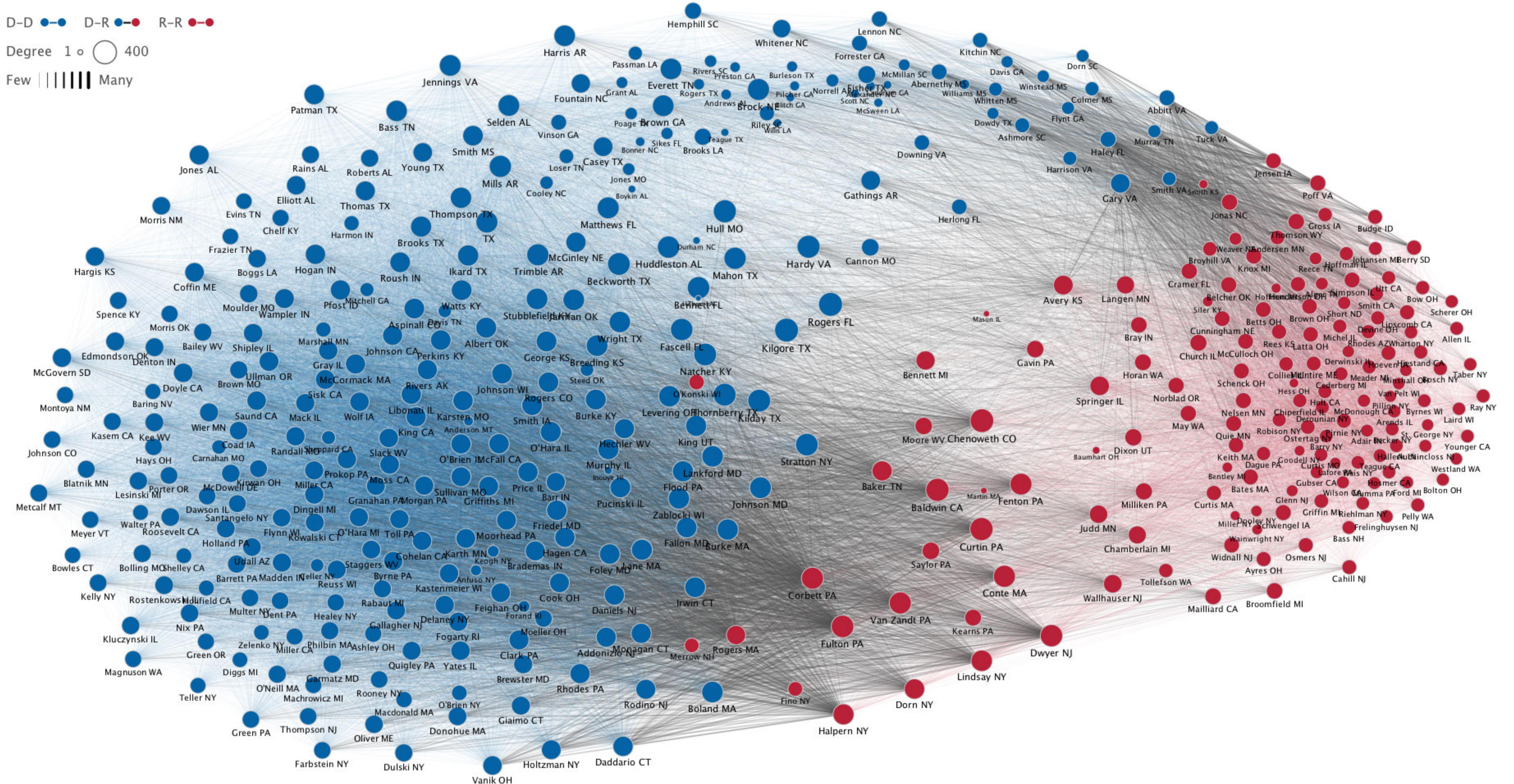


# Year: 1959

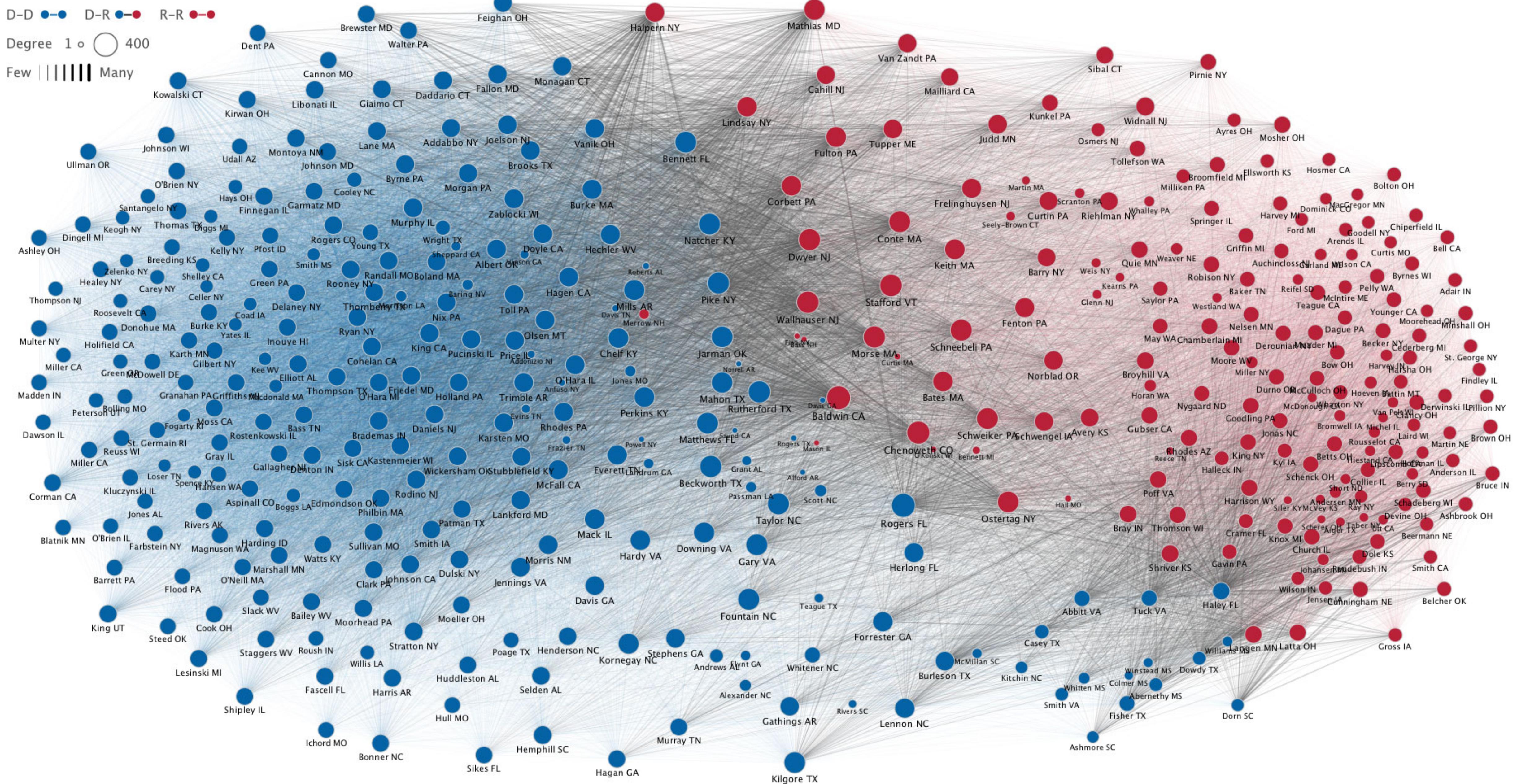
D-D    D-R    R-R

Degree 1 o 400

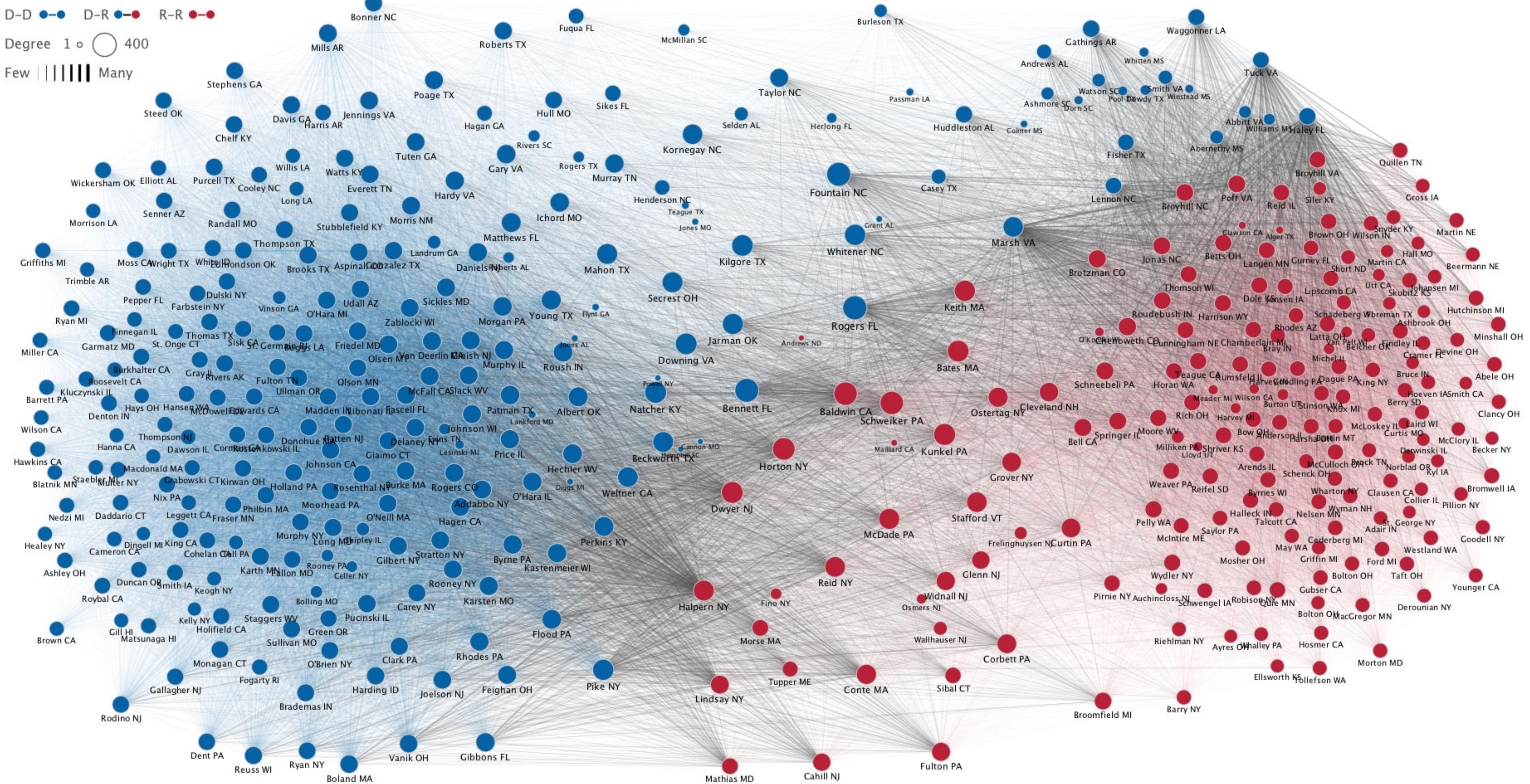
Few ||| Many



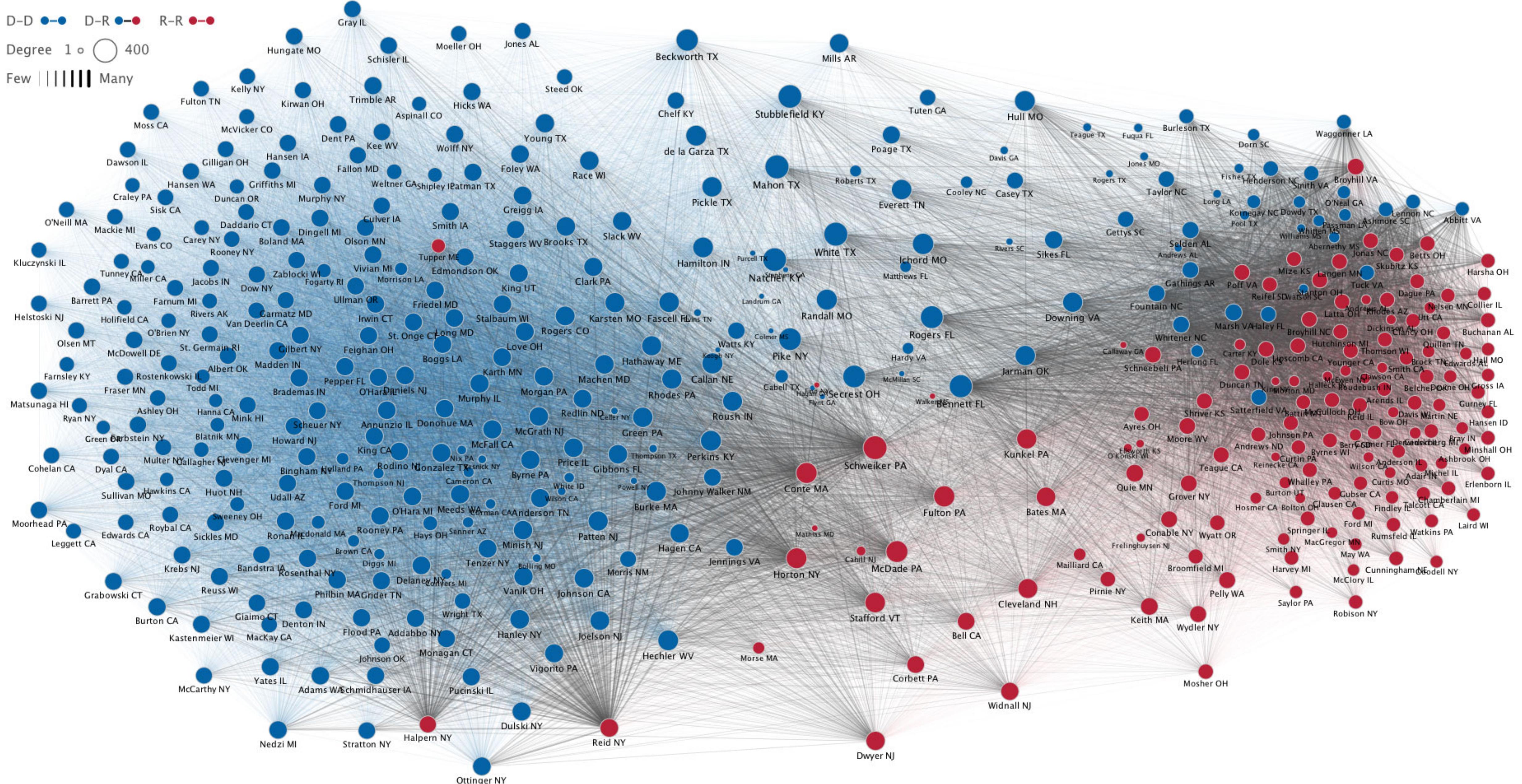
# Year: 1961



# Year: 1963



Year: 1965

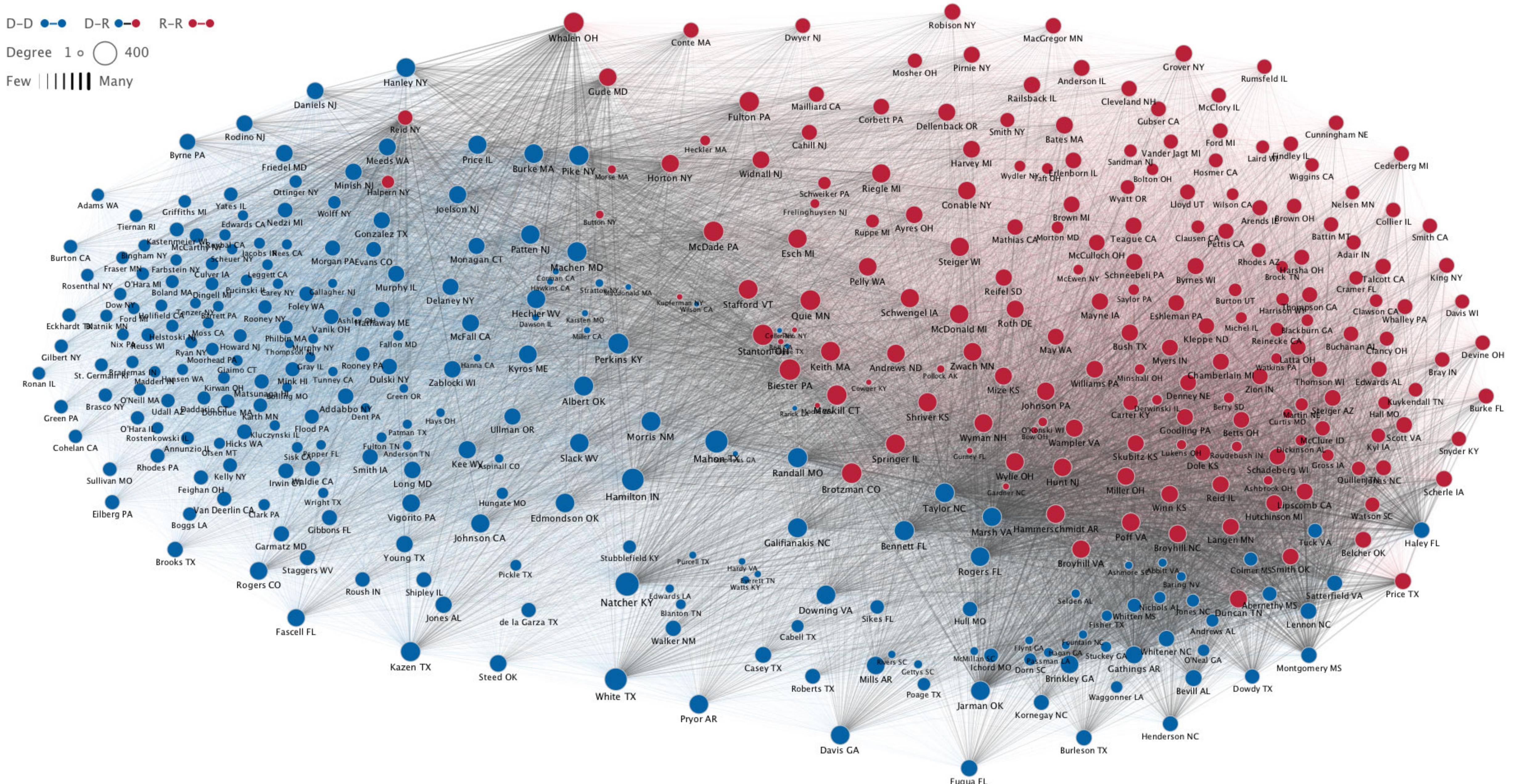


Year: 1967

D-D  D-R  R-R 

Degree  $1^\circ$  400

Few | | | | | Many

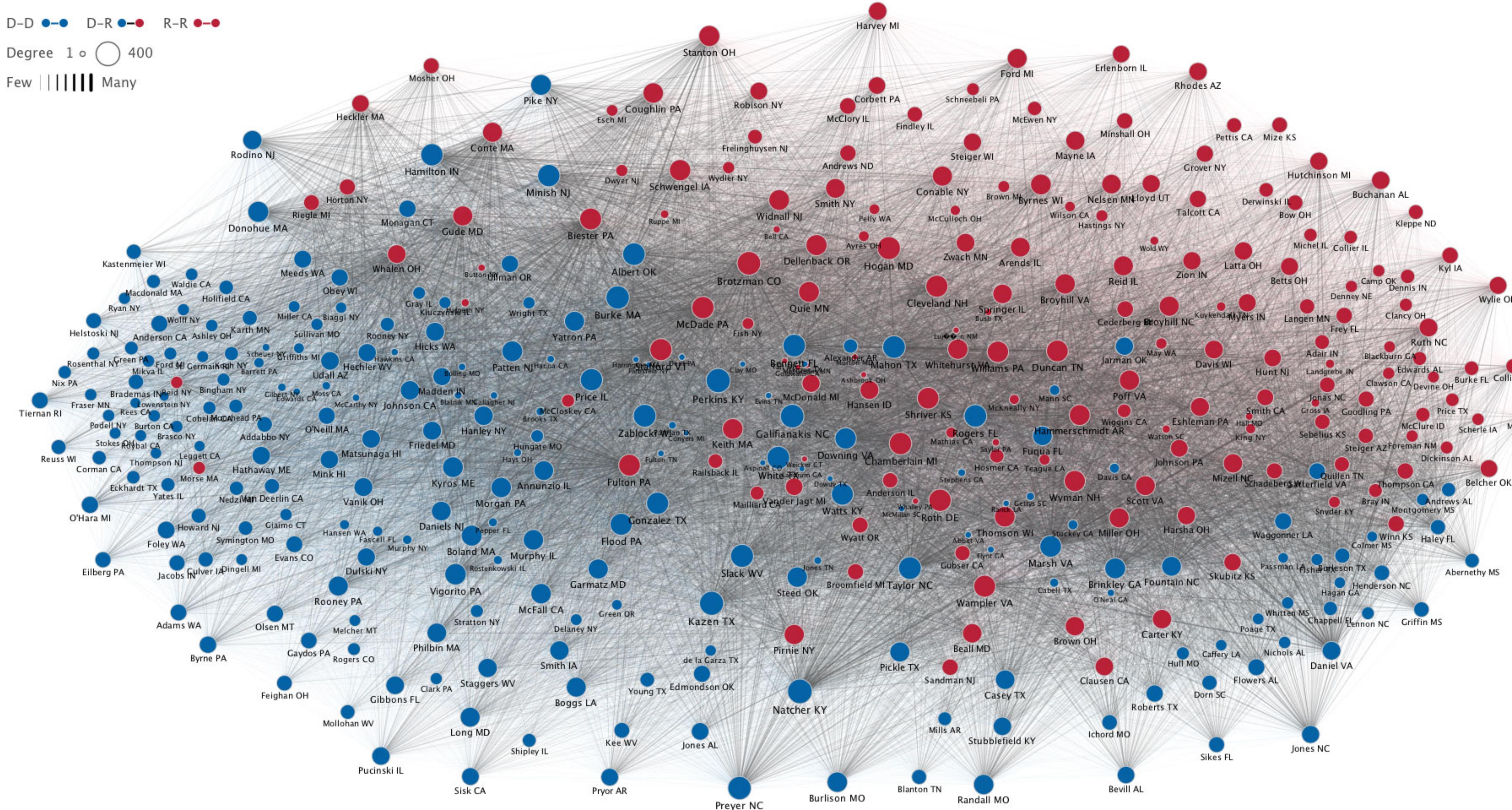


# Year: 1969

D-D    D-R    R-R

Degree 1 o 400

Few ||| Many

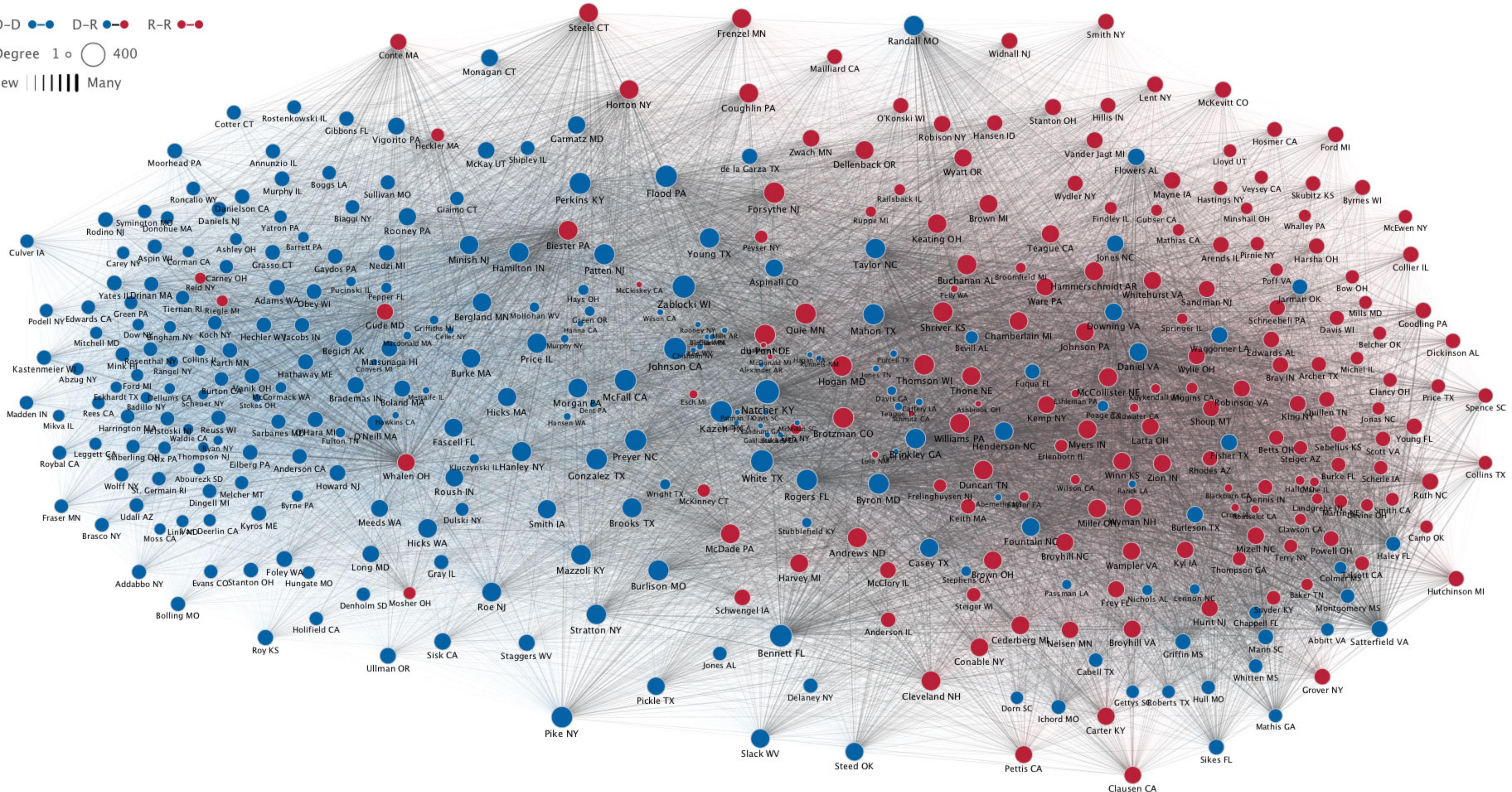


# Year: 1971

D-D    D-R    R-R

Degree 1 o 400

Few ||| Many

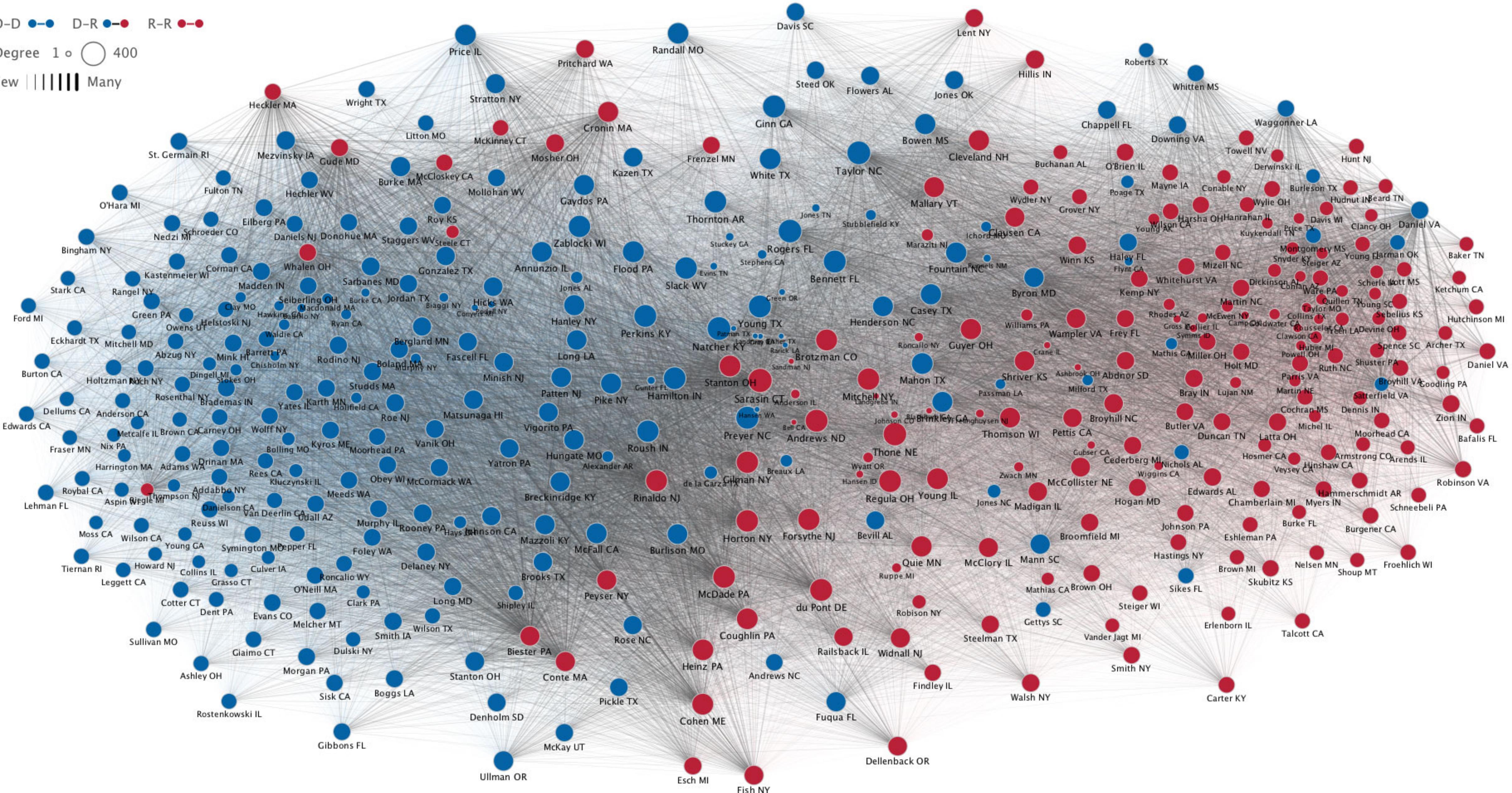


Year: 1973

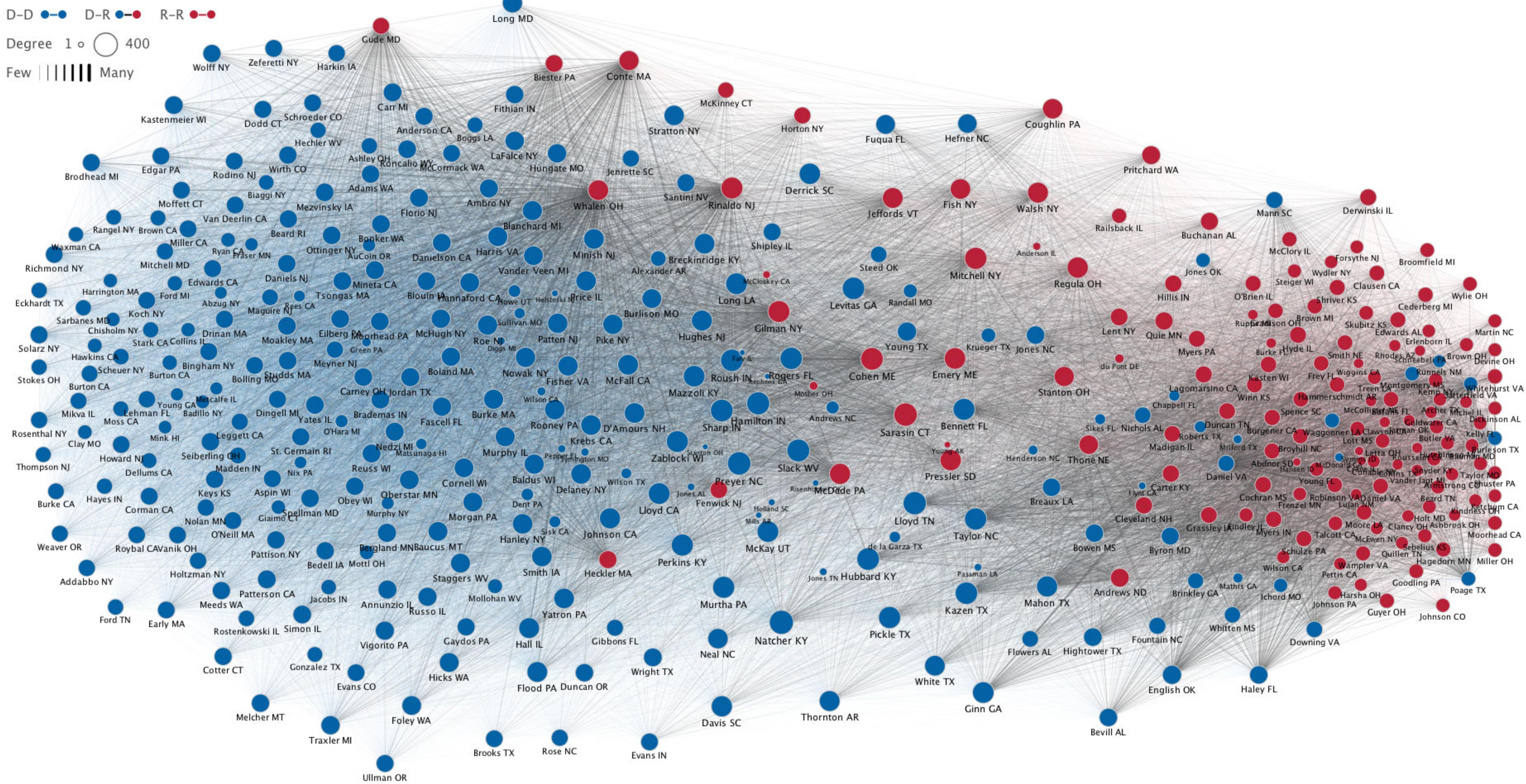
D-D  D-R  R-R 

Degree 1° 400

Few | | | | | Many



# Year: 1975

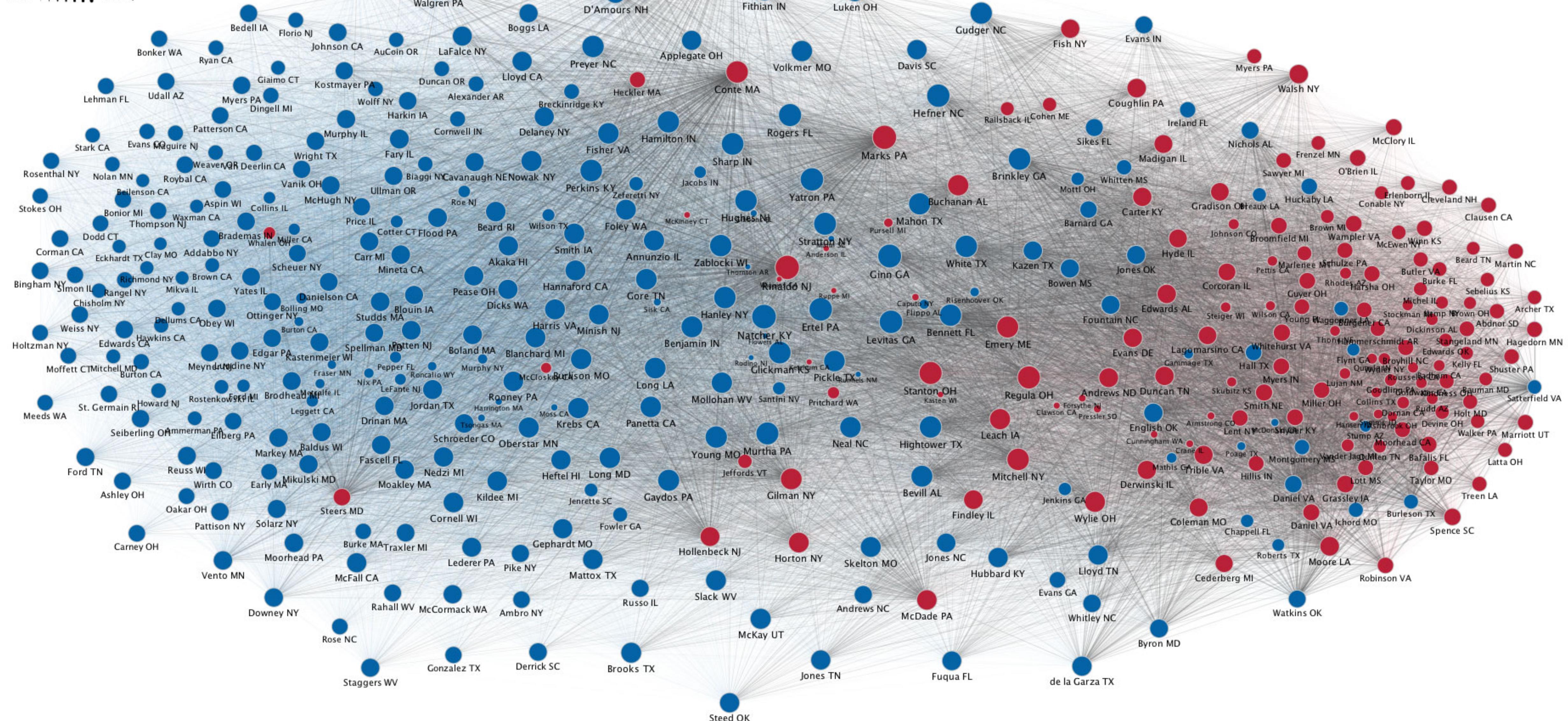


Year: 1977

D-D  D-R  R-R 

Degree 1 °  400

Few | | | | Many

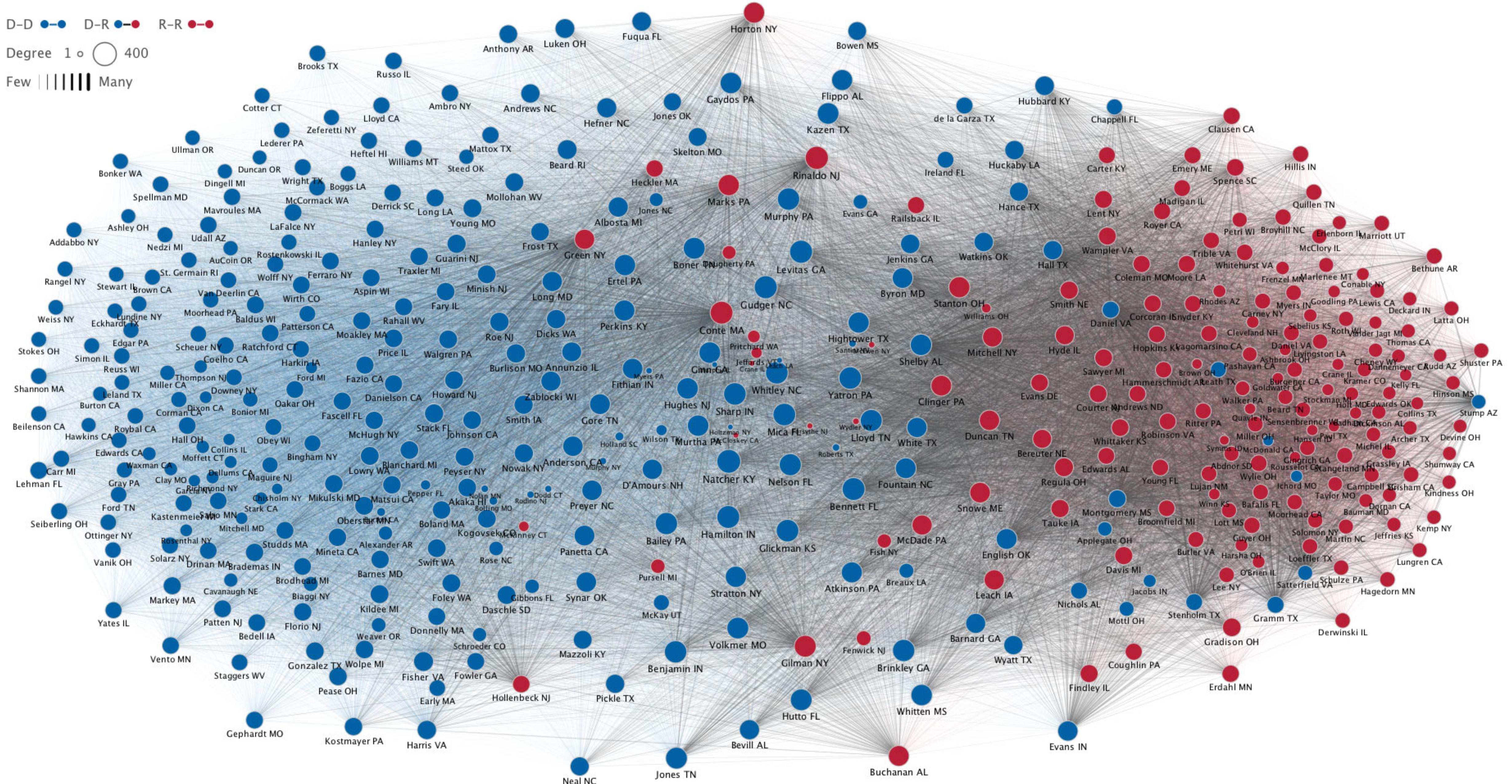


Year: 1979

D-D  D-R  R-R 

Degree 1°  400

Few | | | | | Many

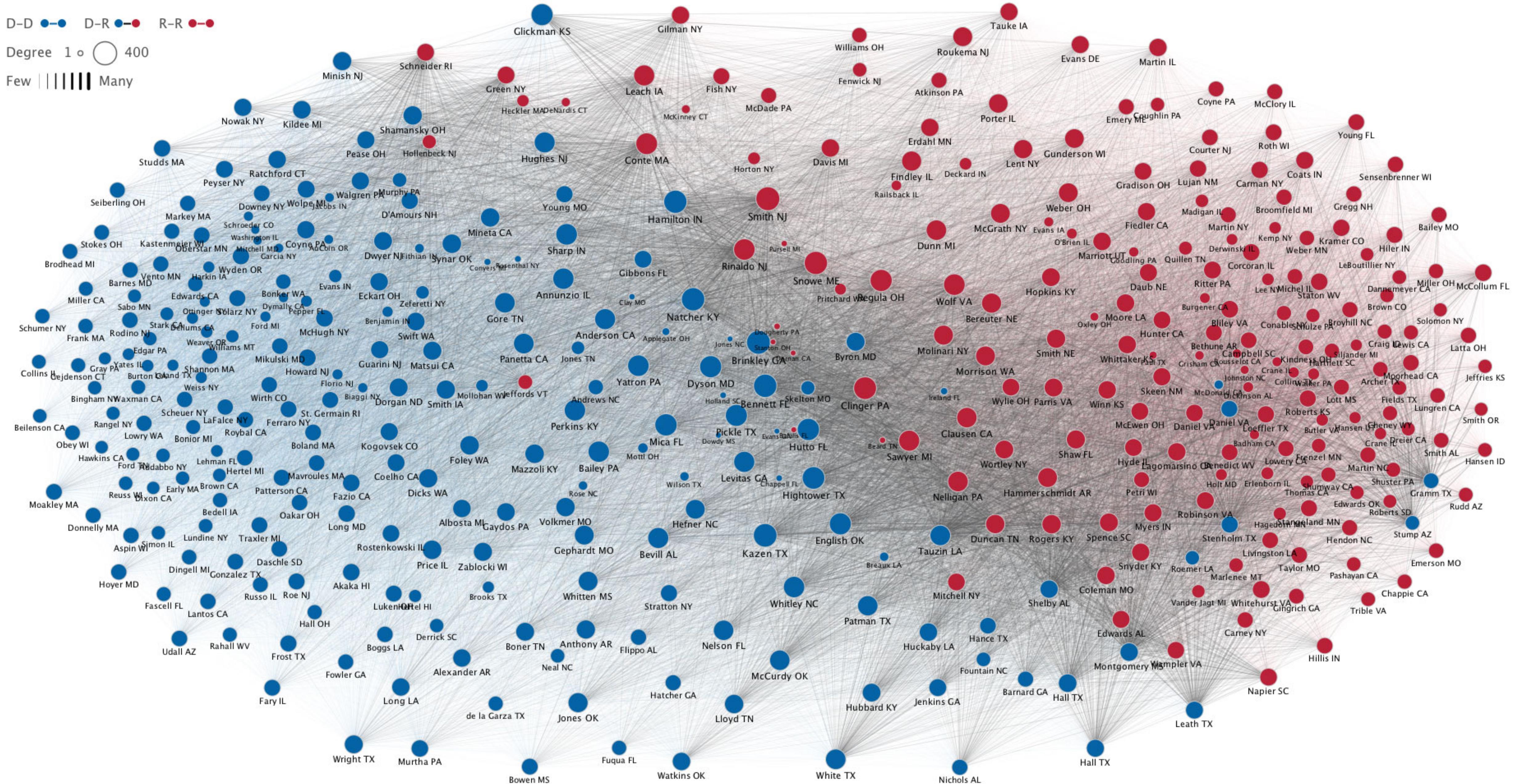


Year: 1981

D-D  D-R  R-R 

Degree  $1^\circ$  400

Few | | | | Many

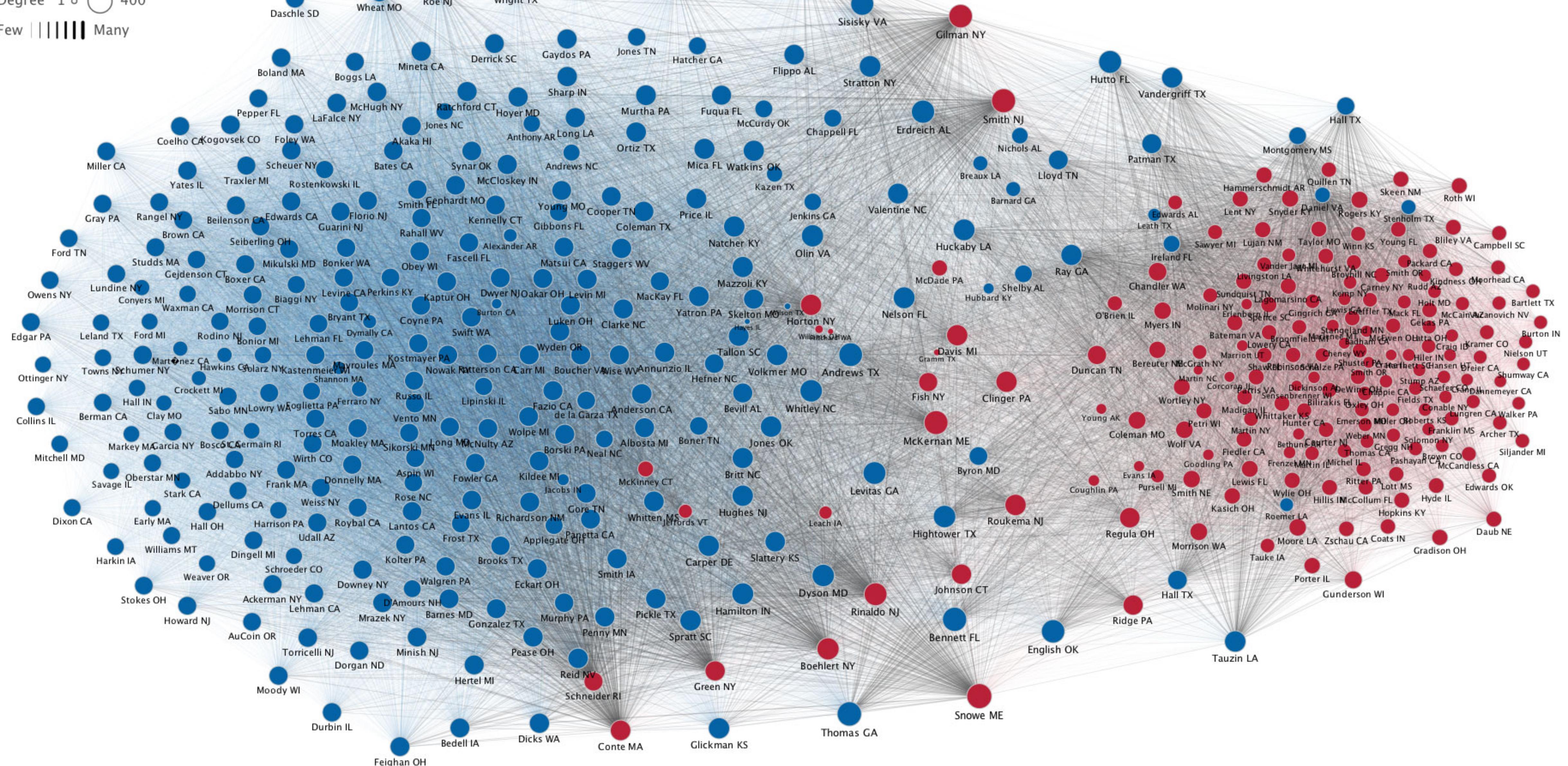


# Year: 1983

D-D •—• D-R •—• R-R •—•

Degree 1 o 400

Few ||| Many

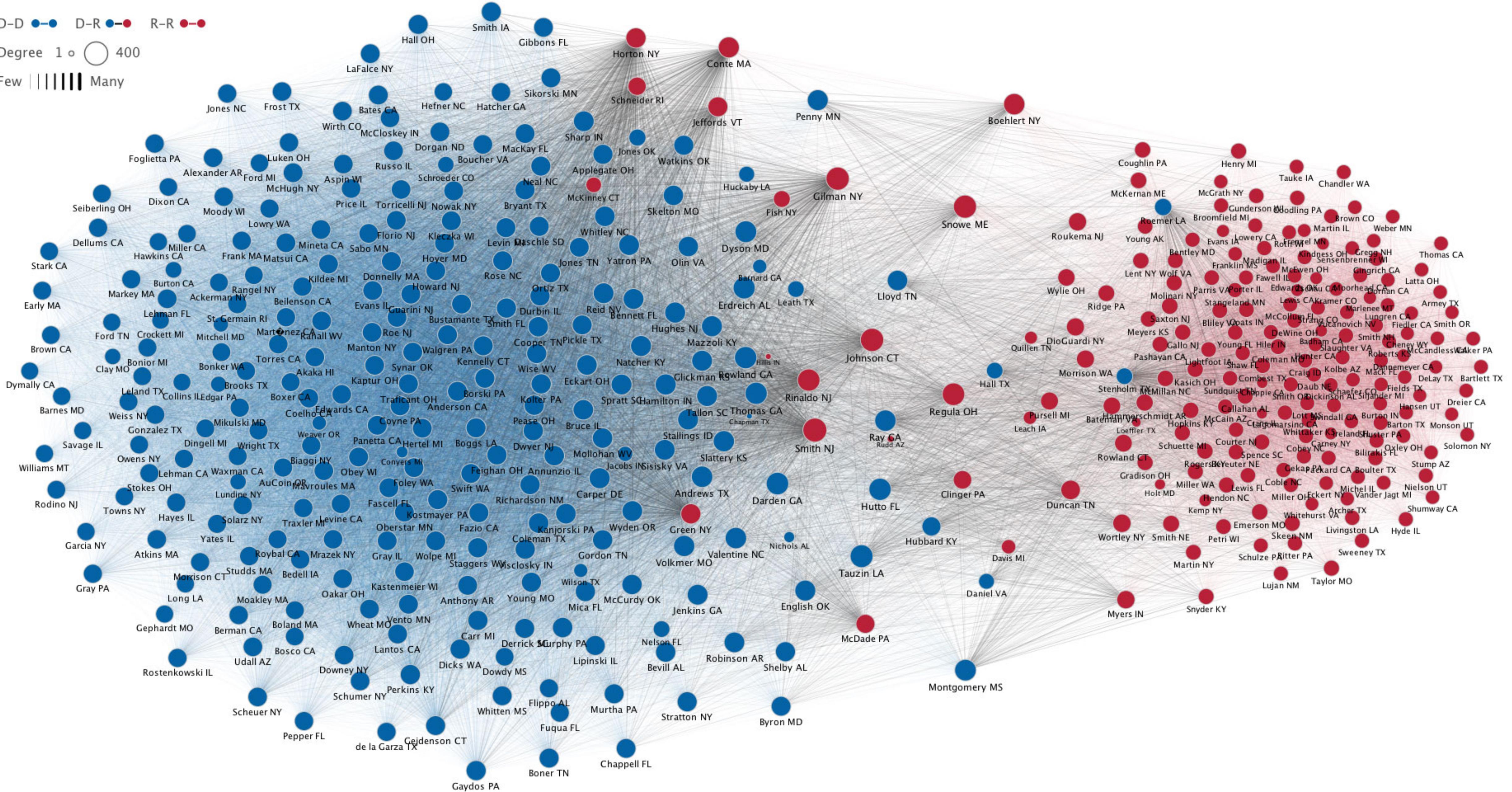


# Year: 1985

D-D    D-R    R-R

Degree 1 o 400

Few ||| Many

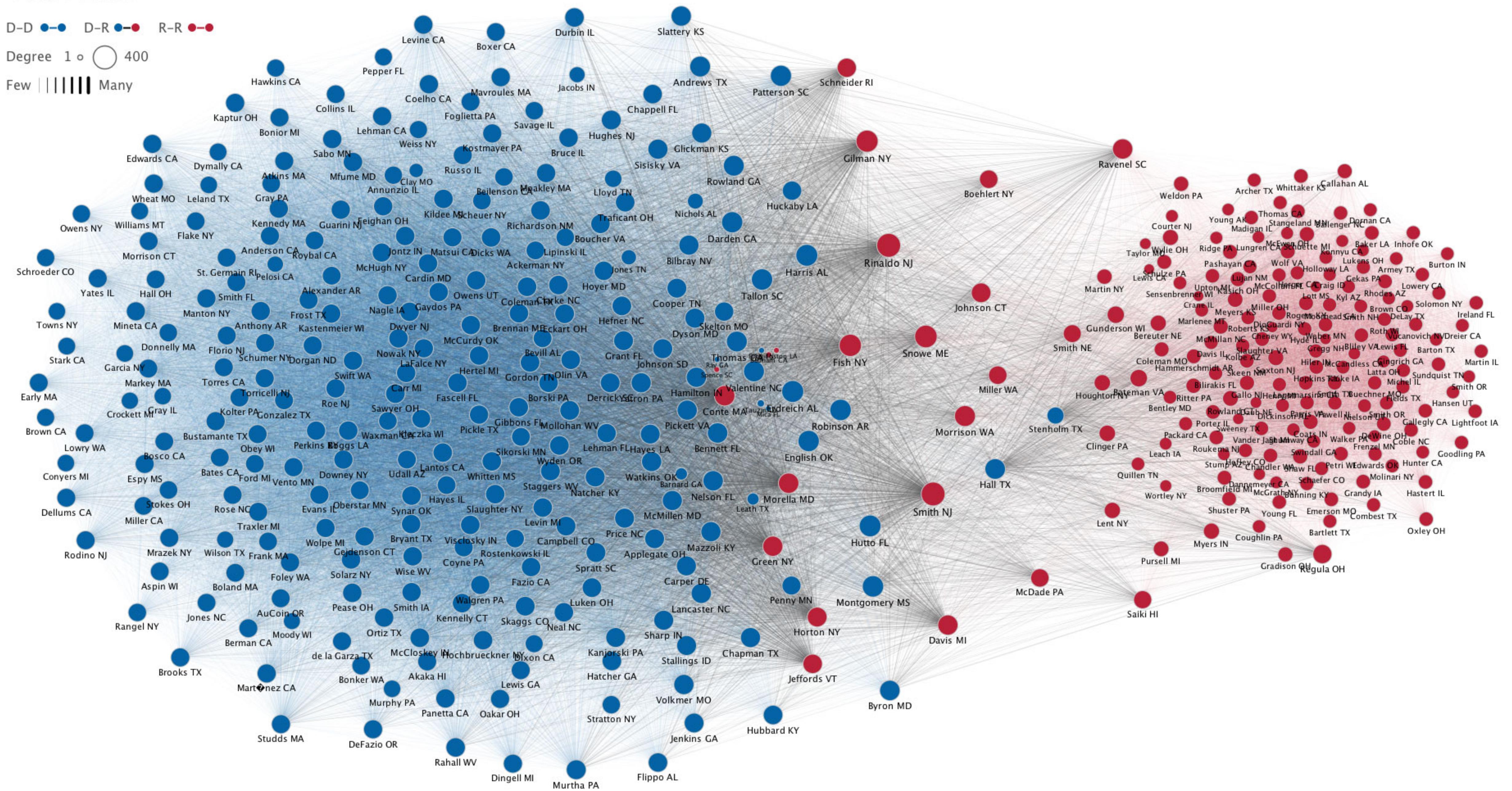


Year: 1987

D-D  D-R  R-R 

Degree 1 °  400

Few | | | | Many

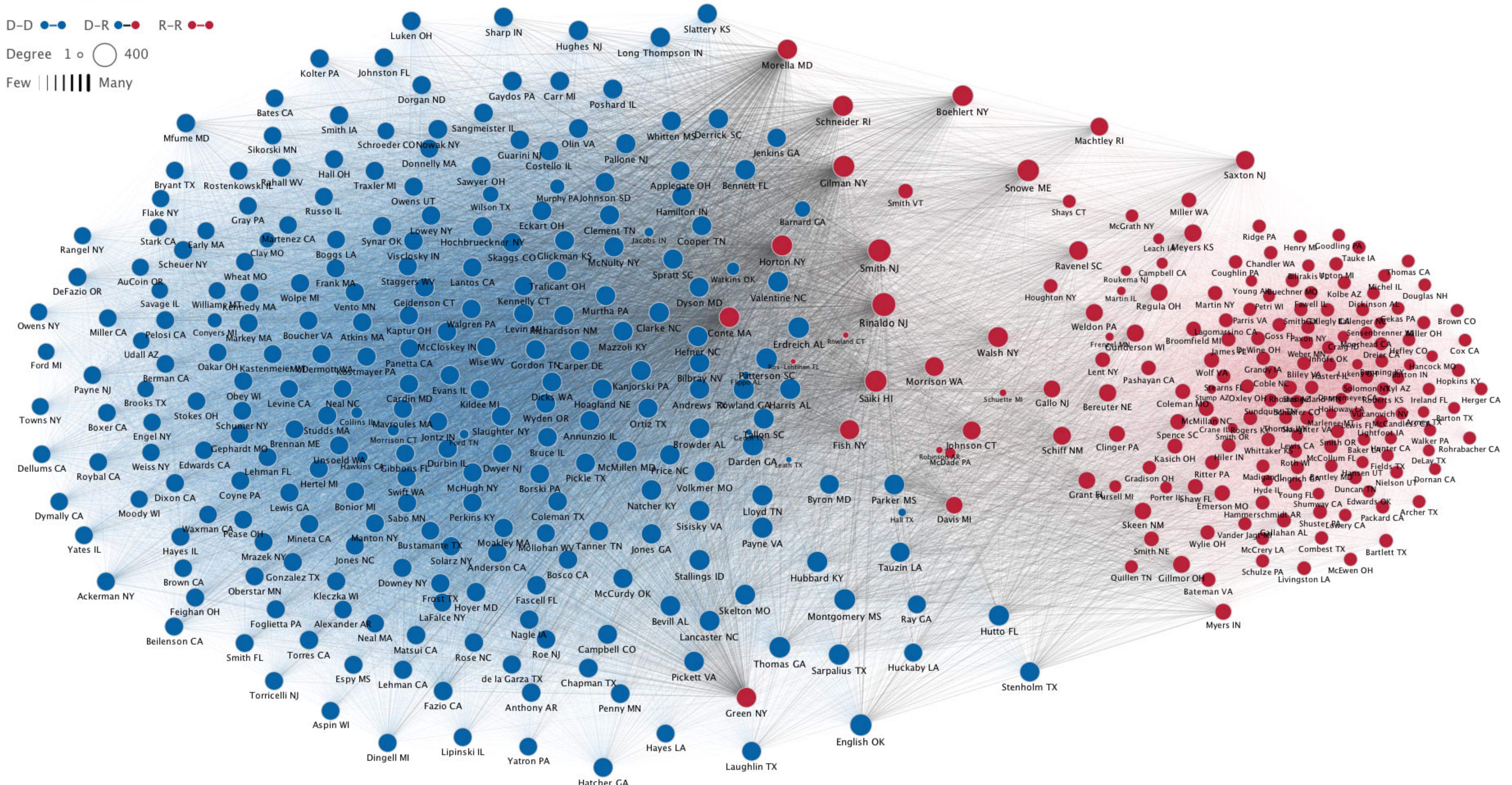


Year: 1989

D-D  D-R  R-R 

Degree 1 °  400

Few | | | | | Many

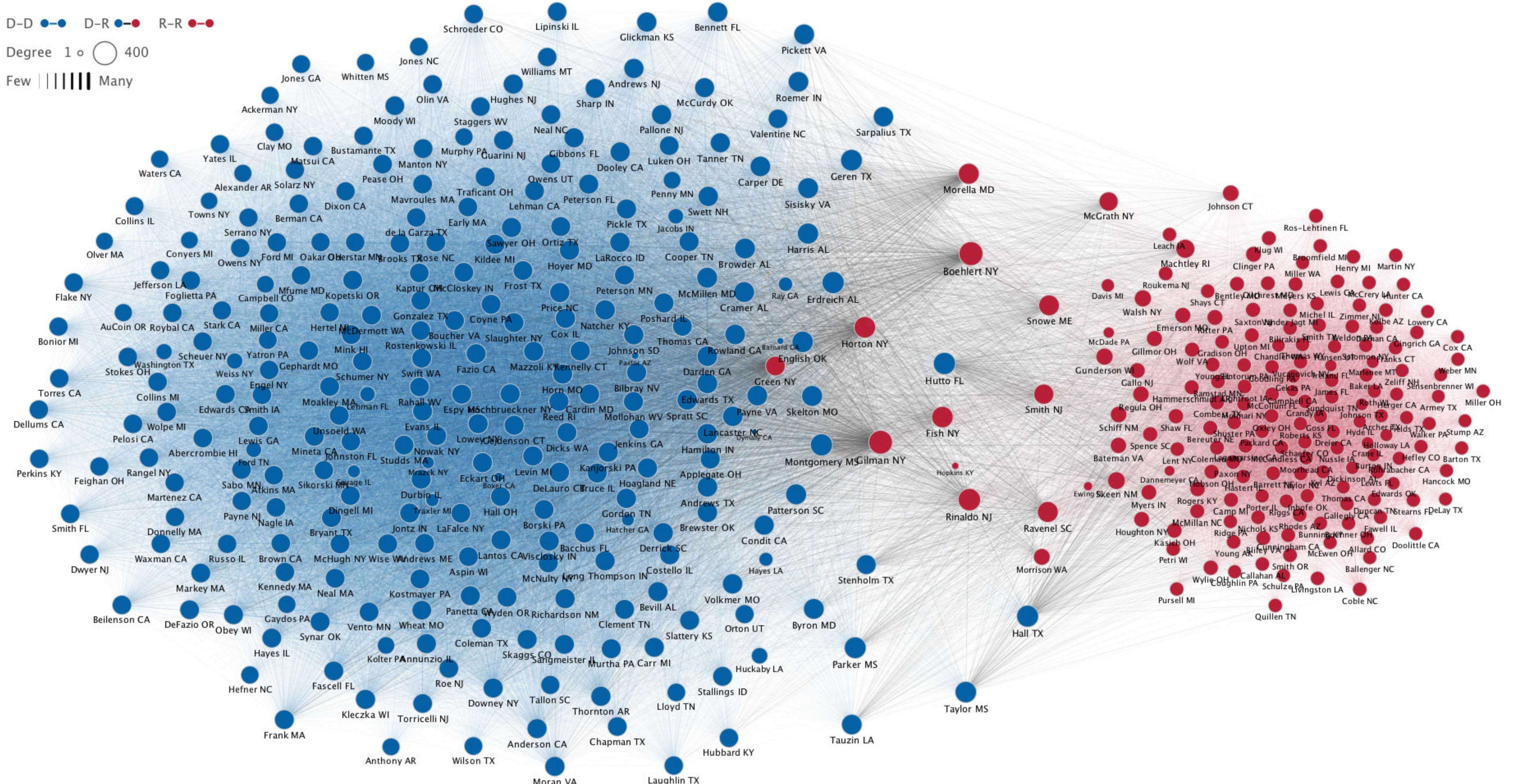


# Year: 1991

D-D    D-R    R-R

Degree 1 o 400

Few ||| Many

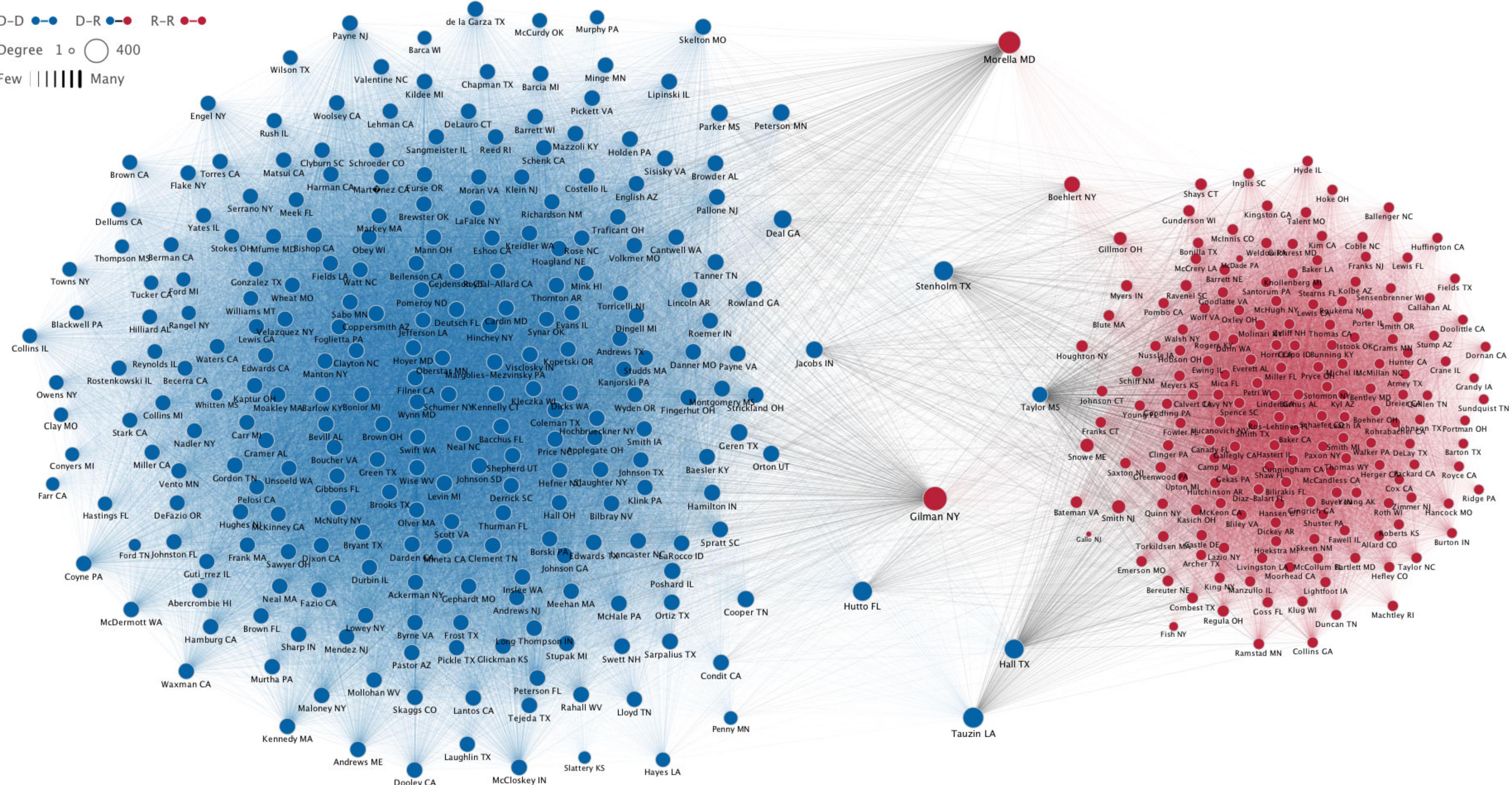


Year: 1993

D-D  D-R  R-R 

Degree 1 °  400

Few | | | | | Many

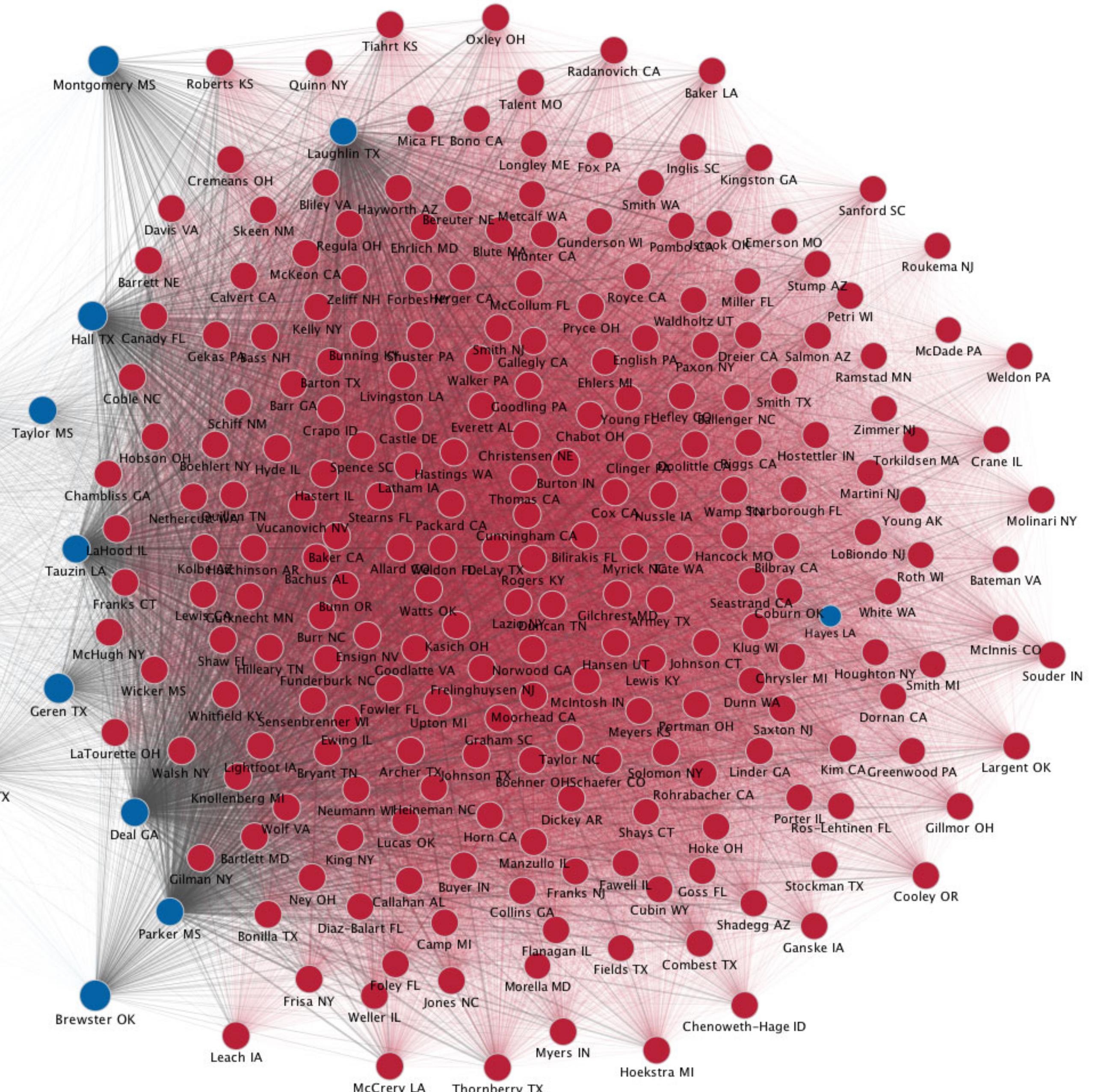
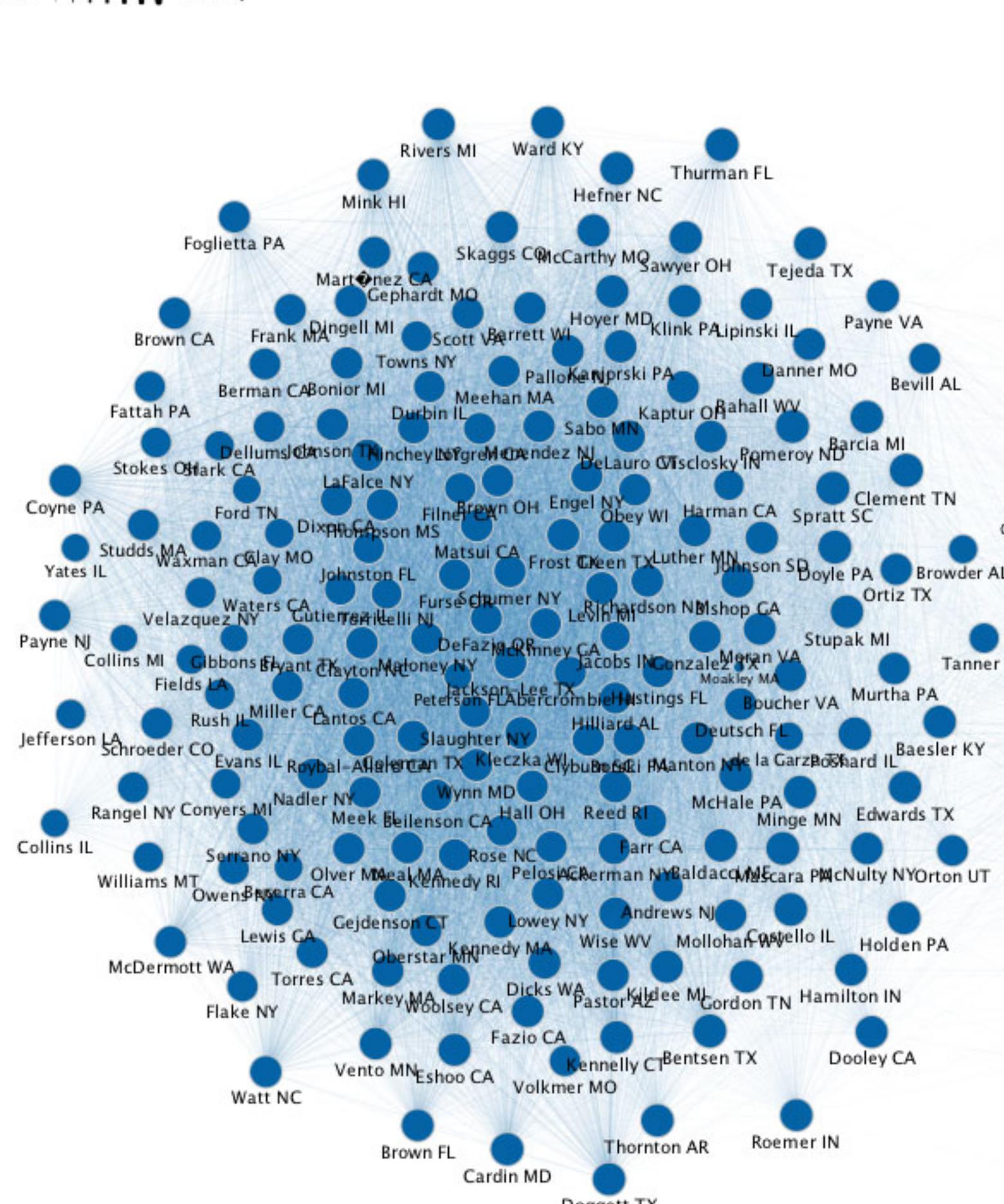


# Year: 1995

D-D    D-R    R-R

Degree 1 o 400

Few ||| Many

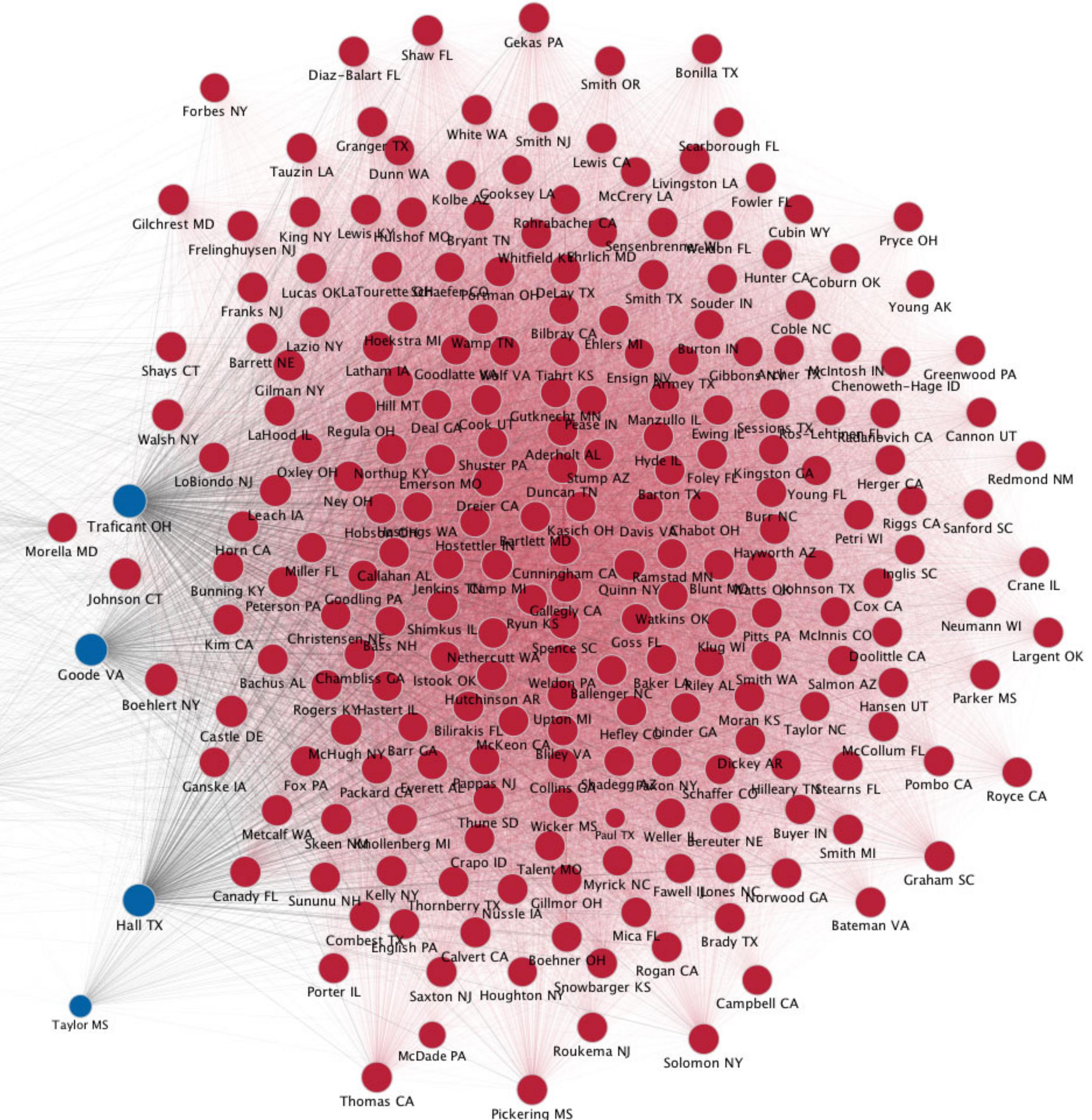
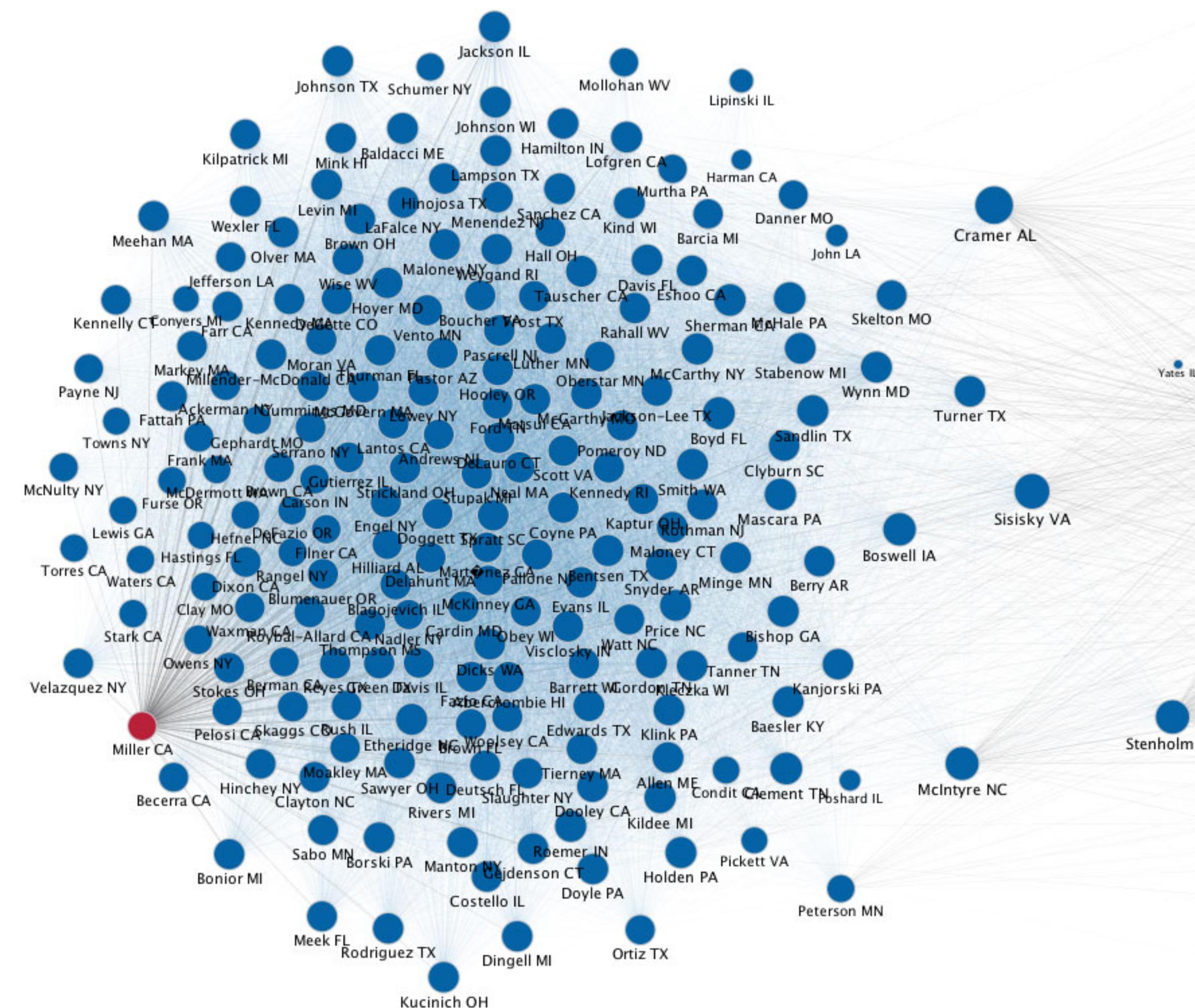


Year: 1997

D-D  D-R  R-R 

Degree 1 °  400

Few | | | | Many

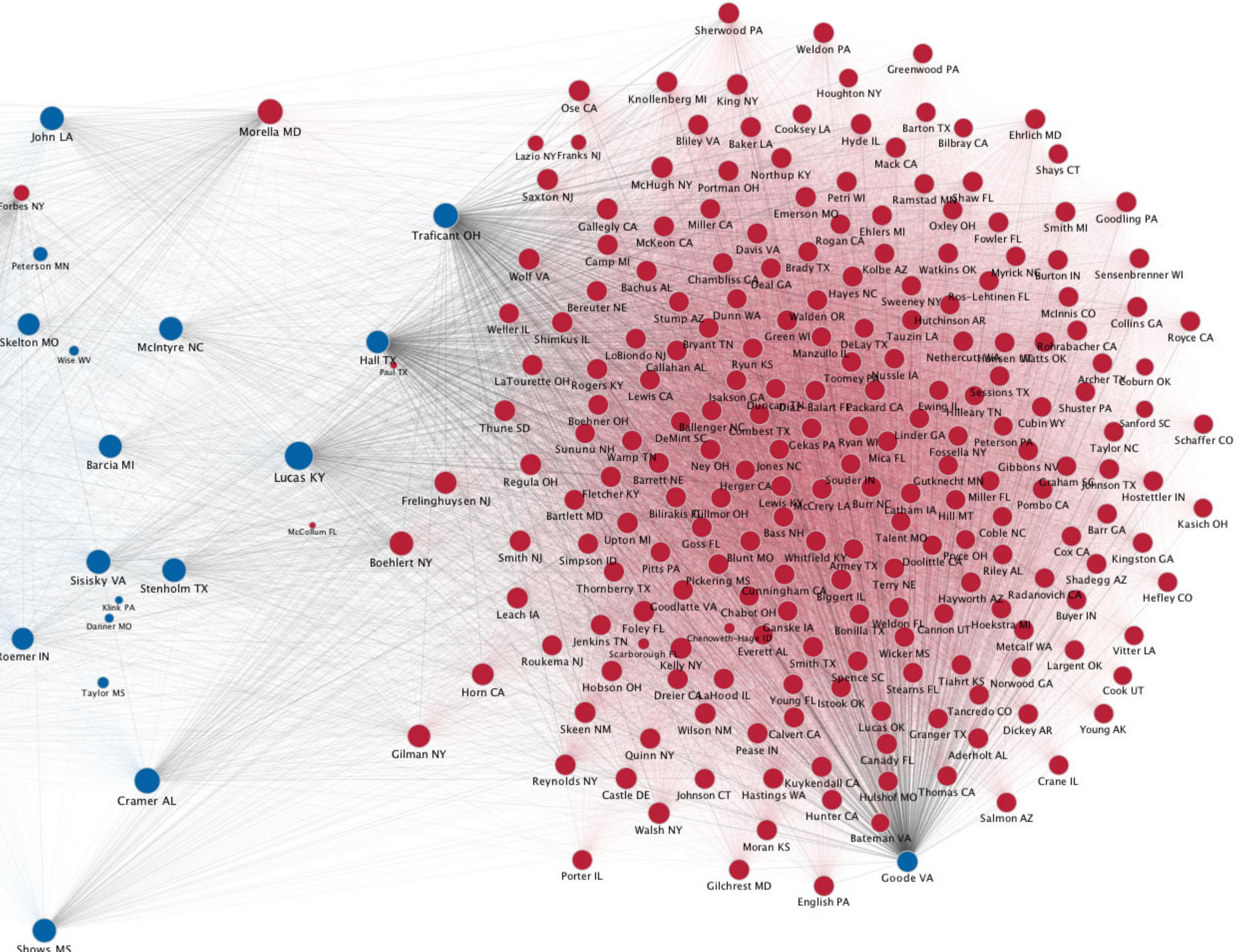
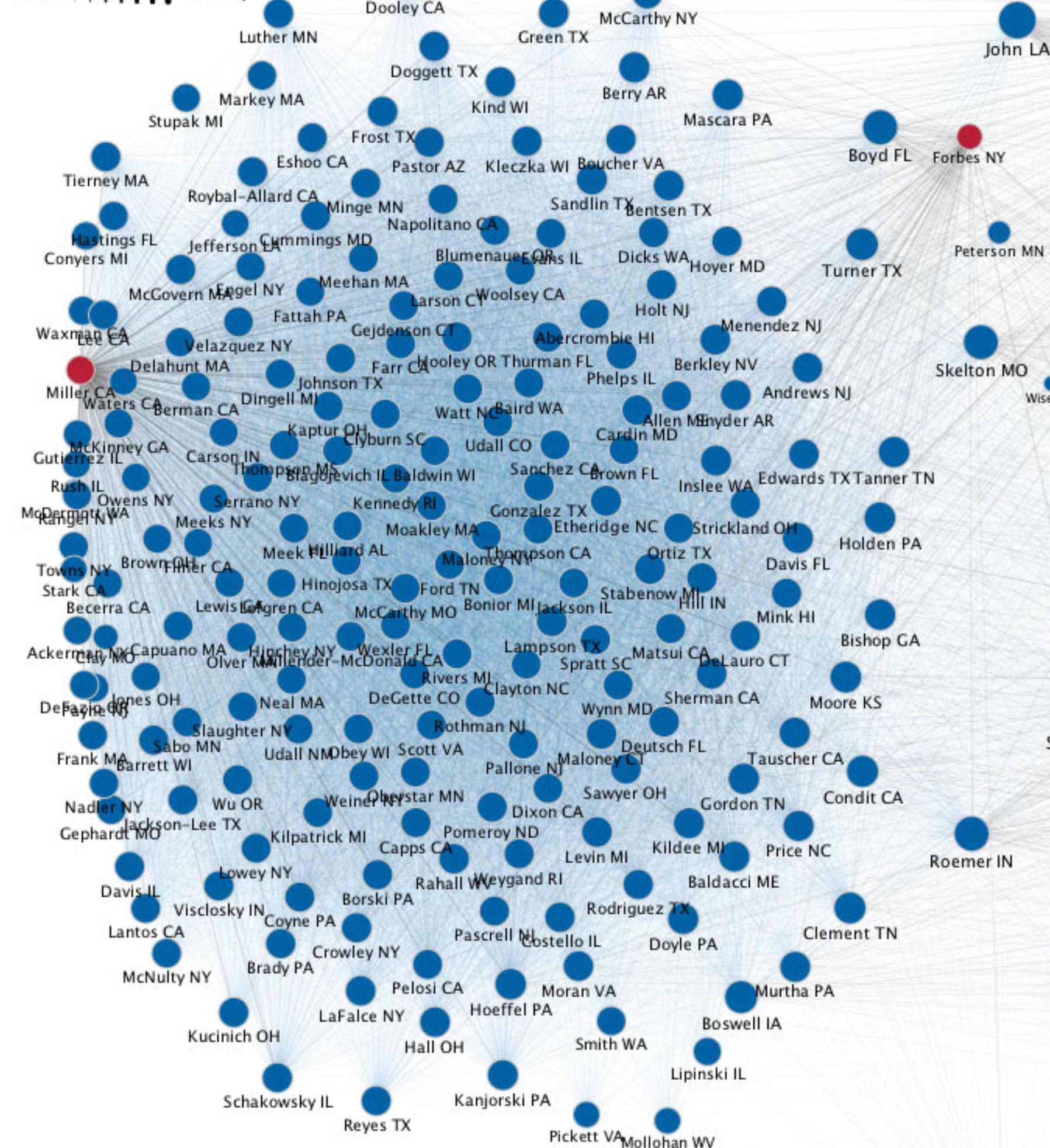


# Year: 1999

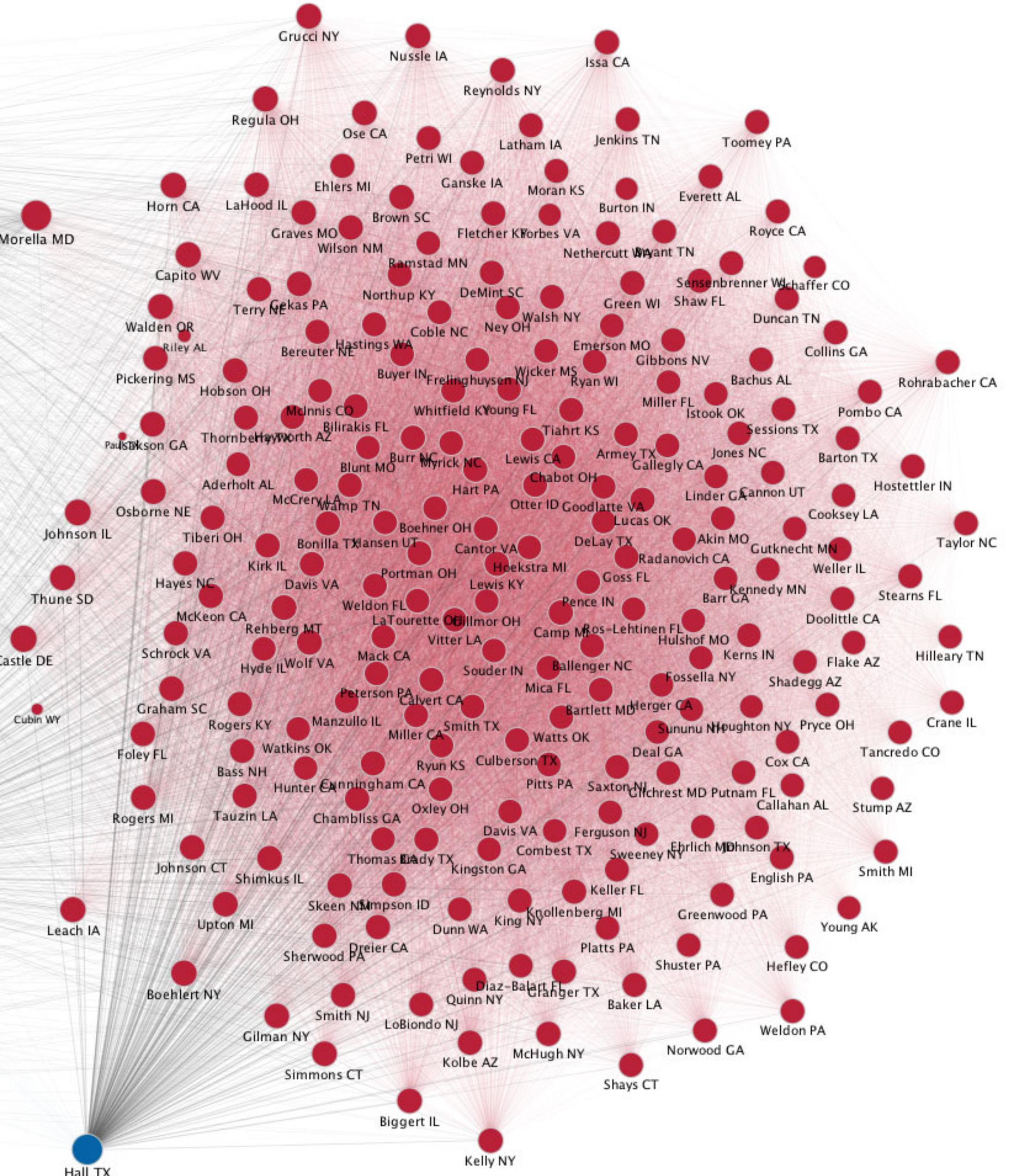
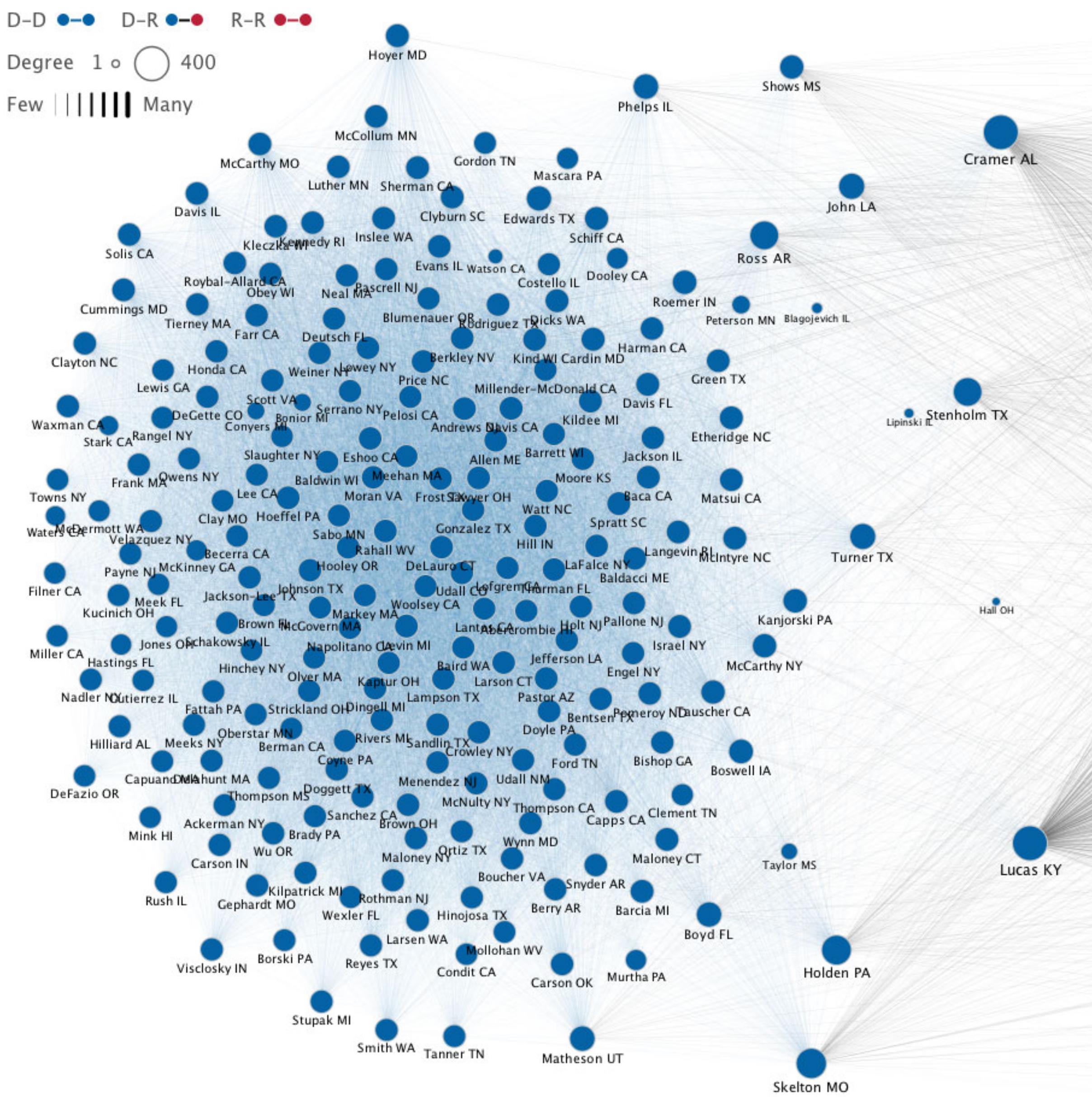
D-D    D-R    R-R

Degree 1 o 400

Few ||| Many



# Year: 2001

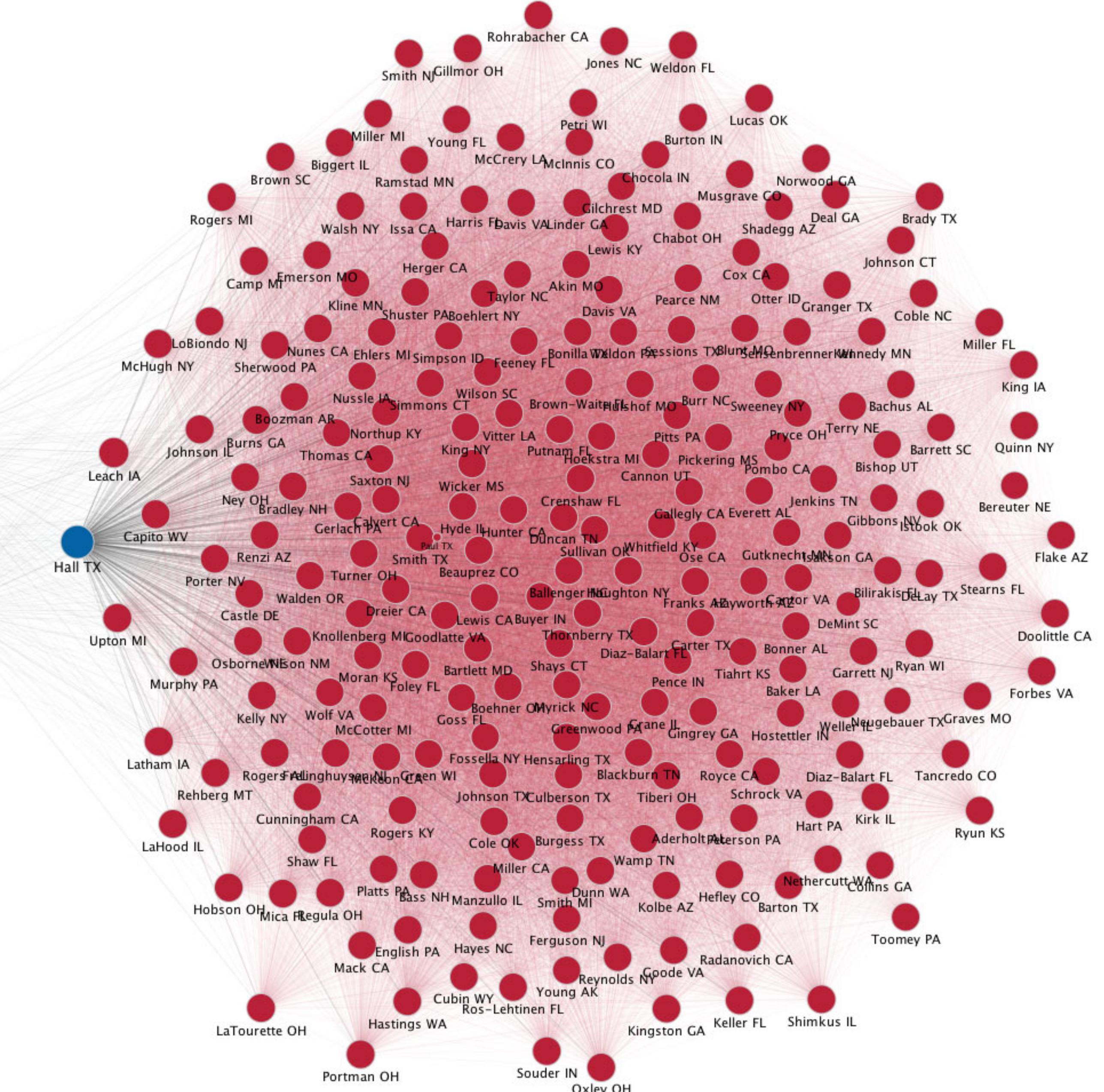
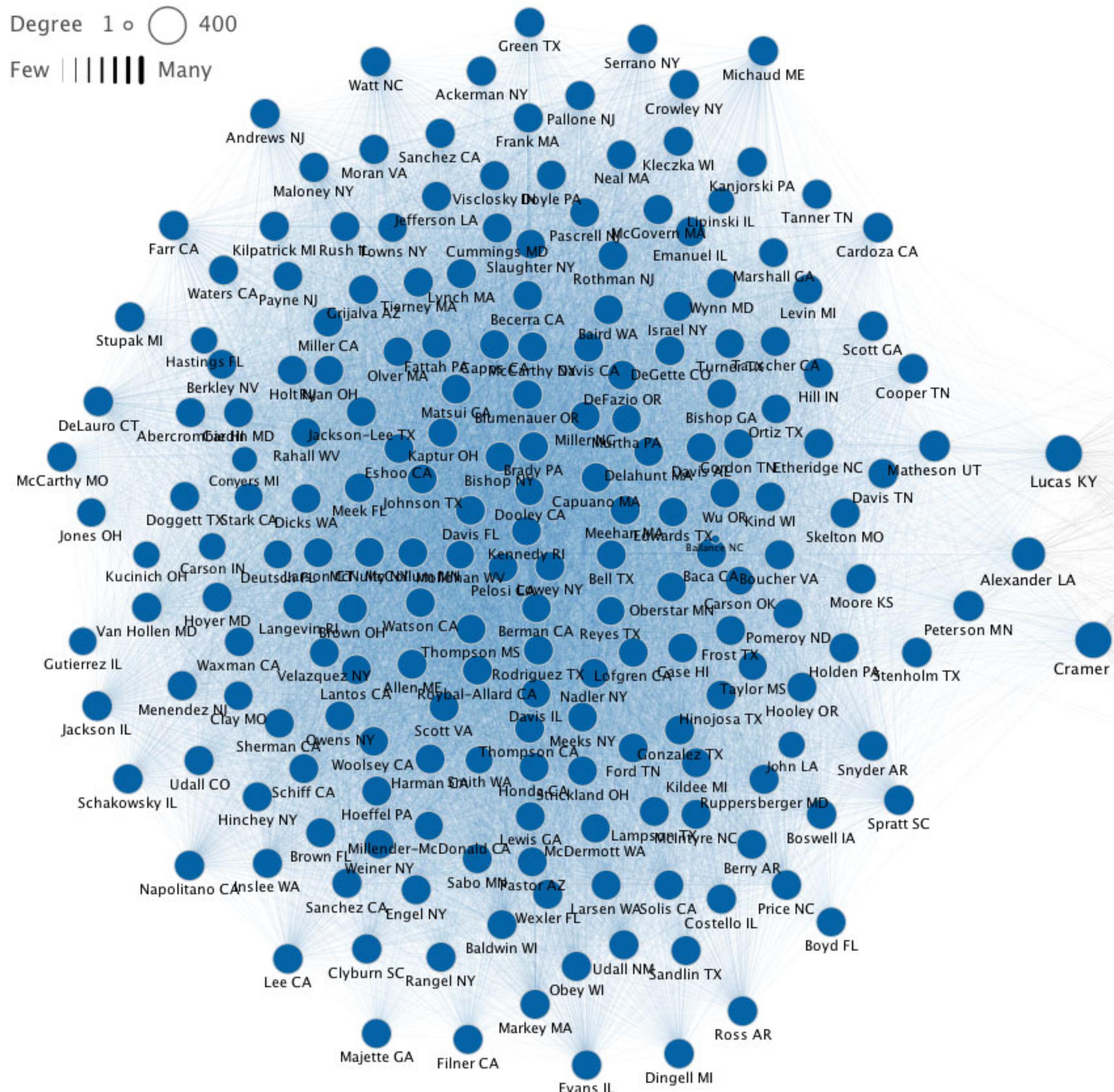


# Year: 2003

D-D    D-R    R-R

Degree 1 o 400

Few |||| Many

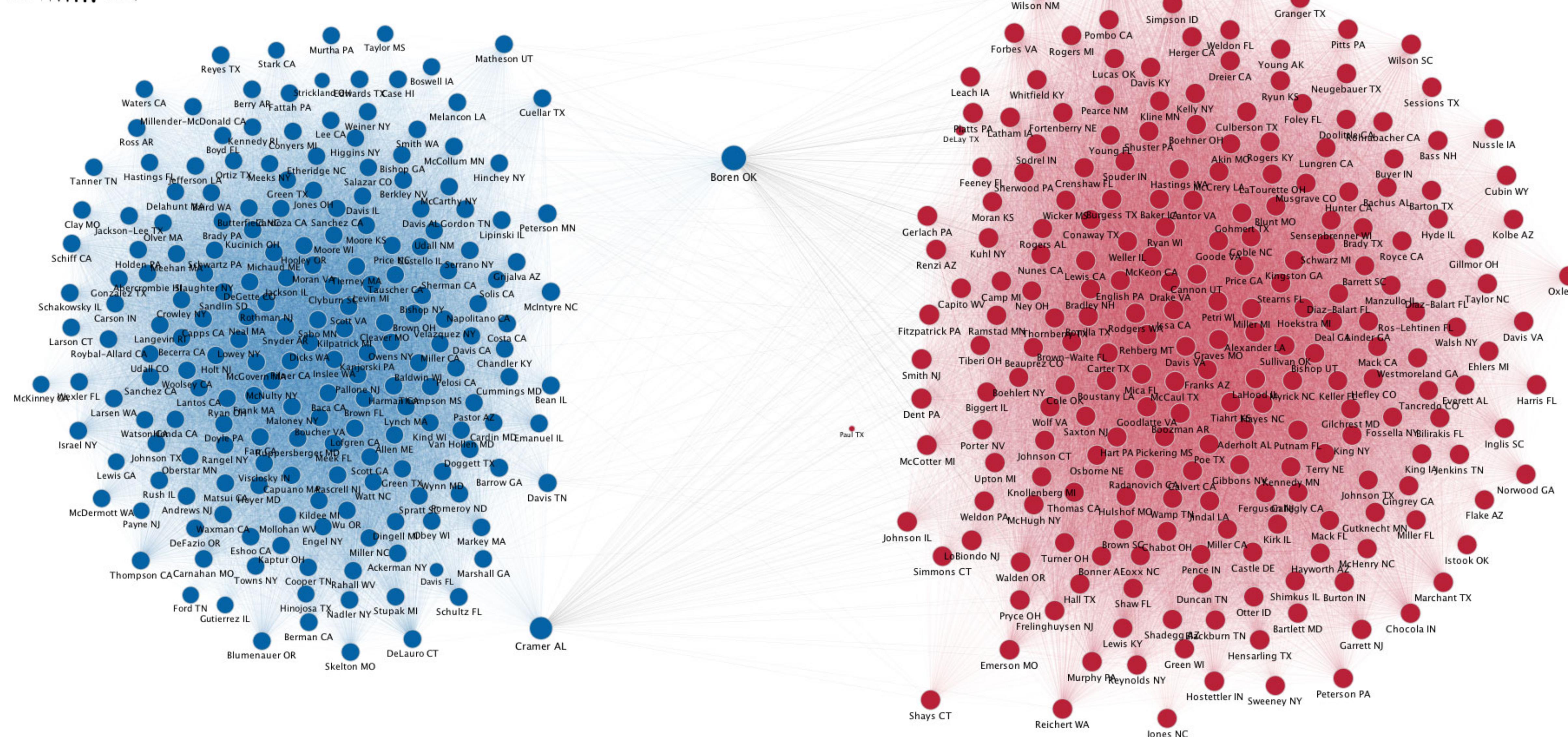


# Year: 2005

D-D •—• D-R •—• R-R •—•

Degree 1 o 400

Few ||| Many

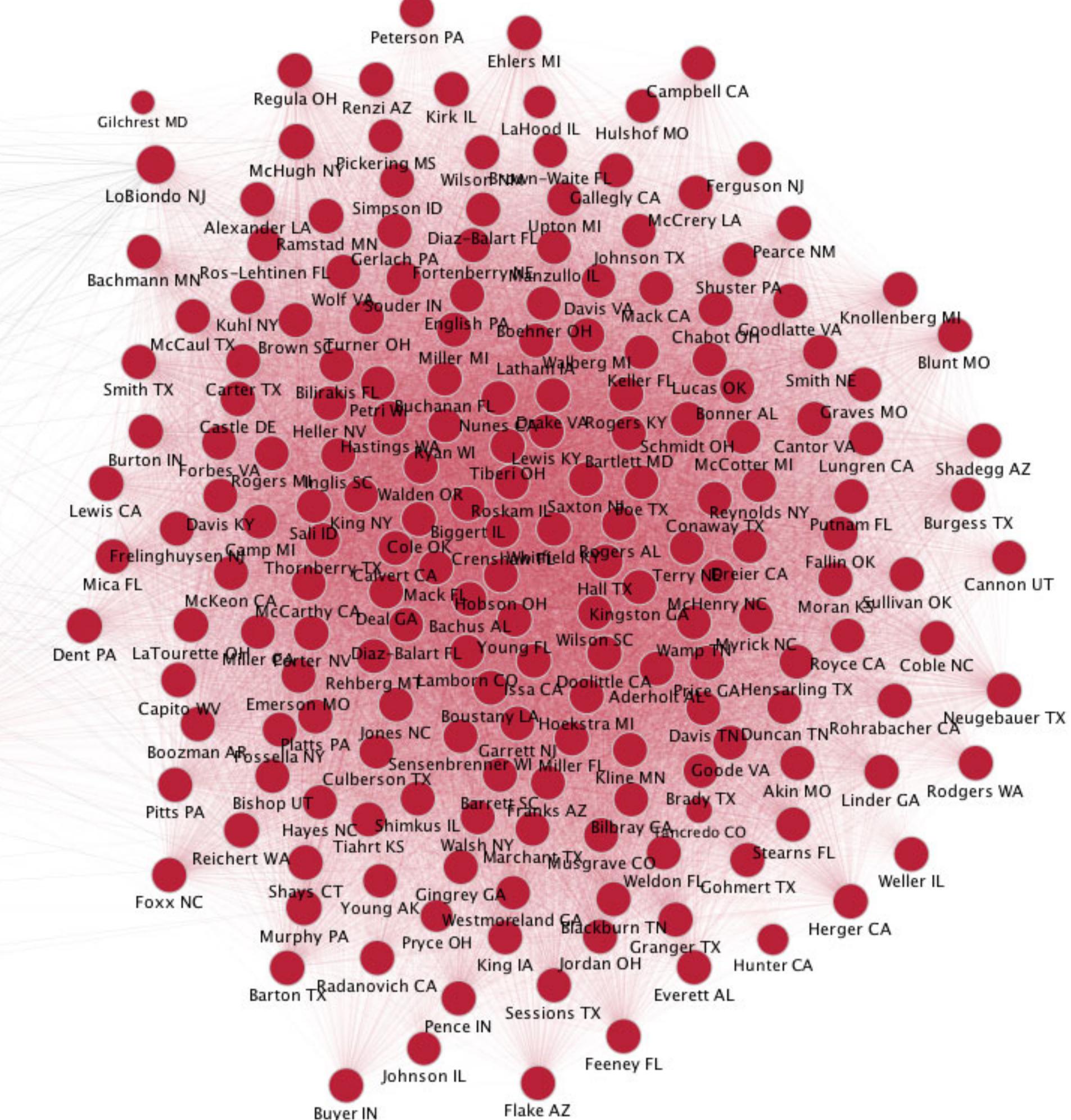
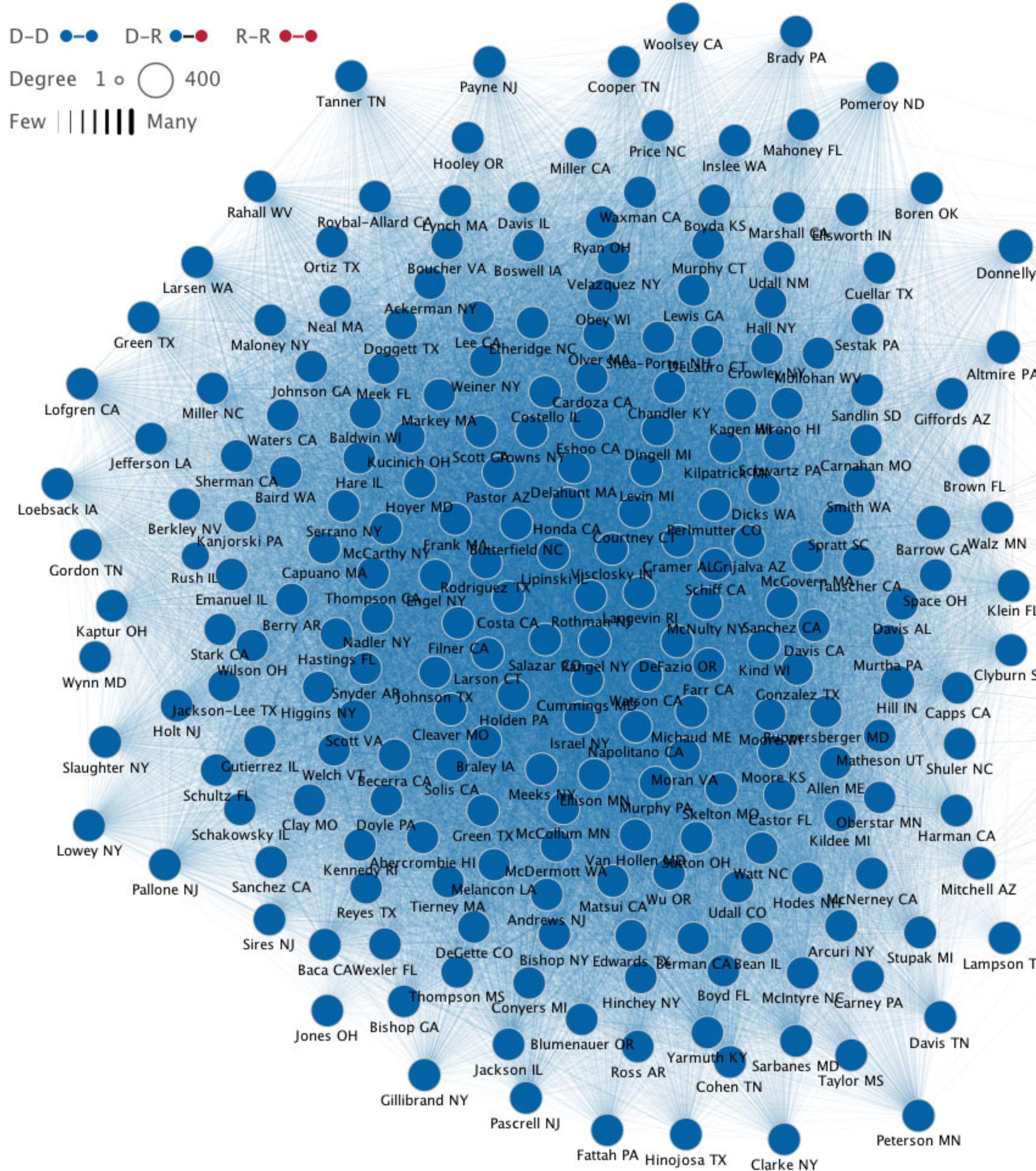


# Year: 2007

D-D    D-R    R-R

Degree 1 o 400

Few ||| Many

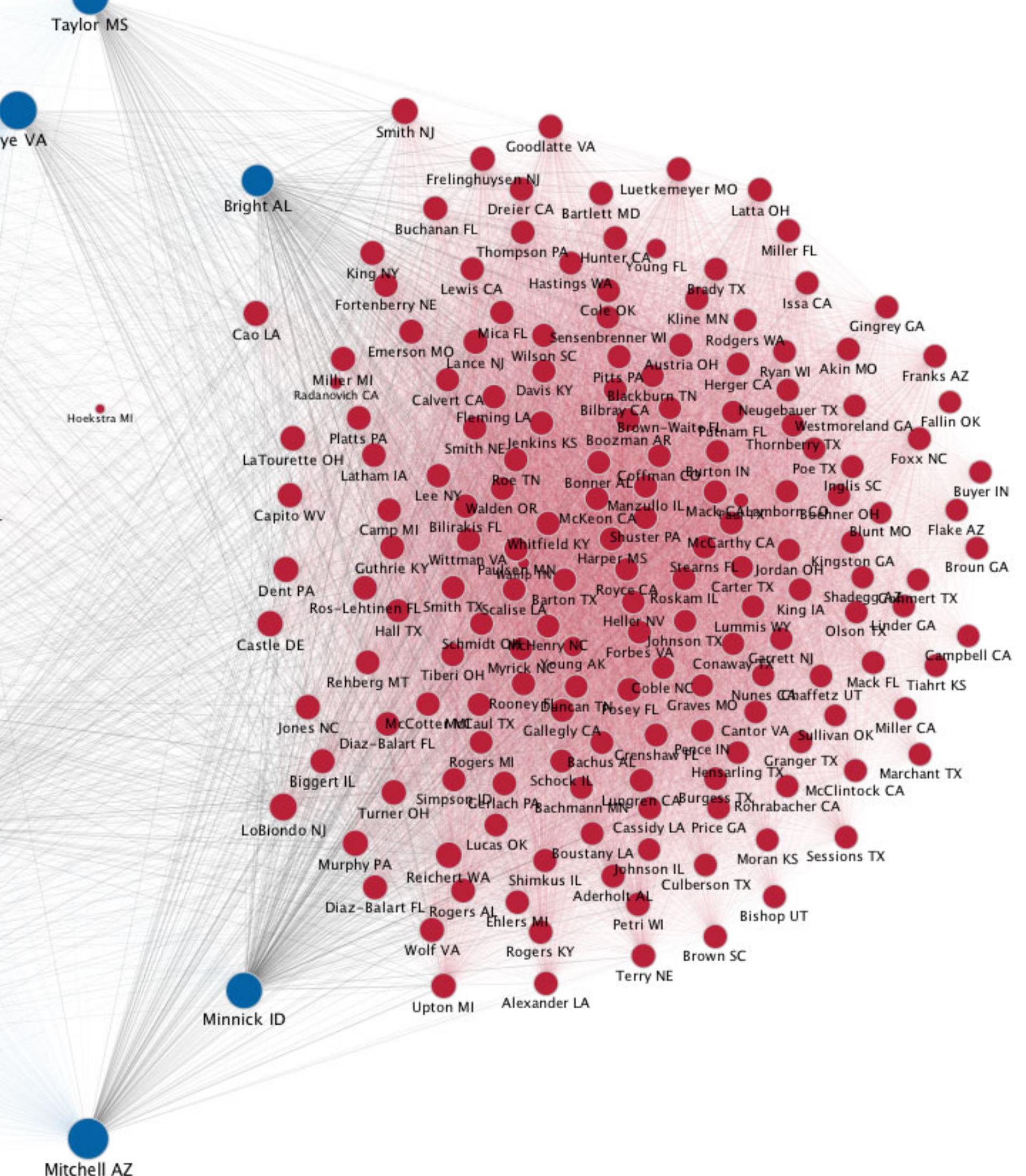
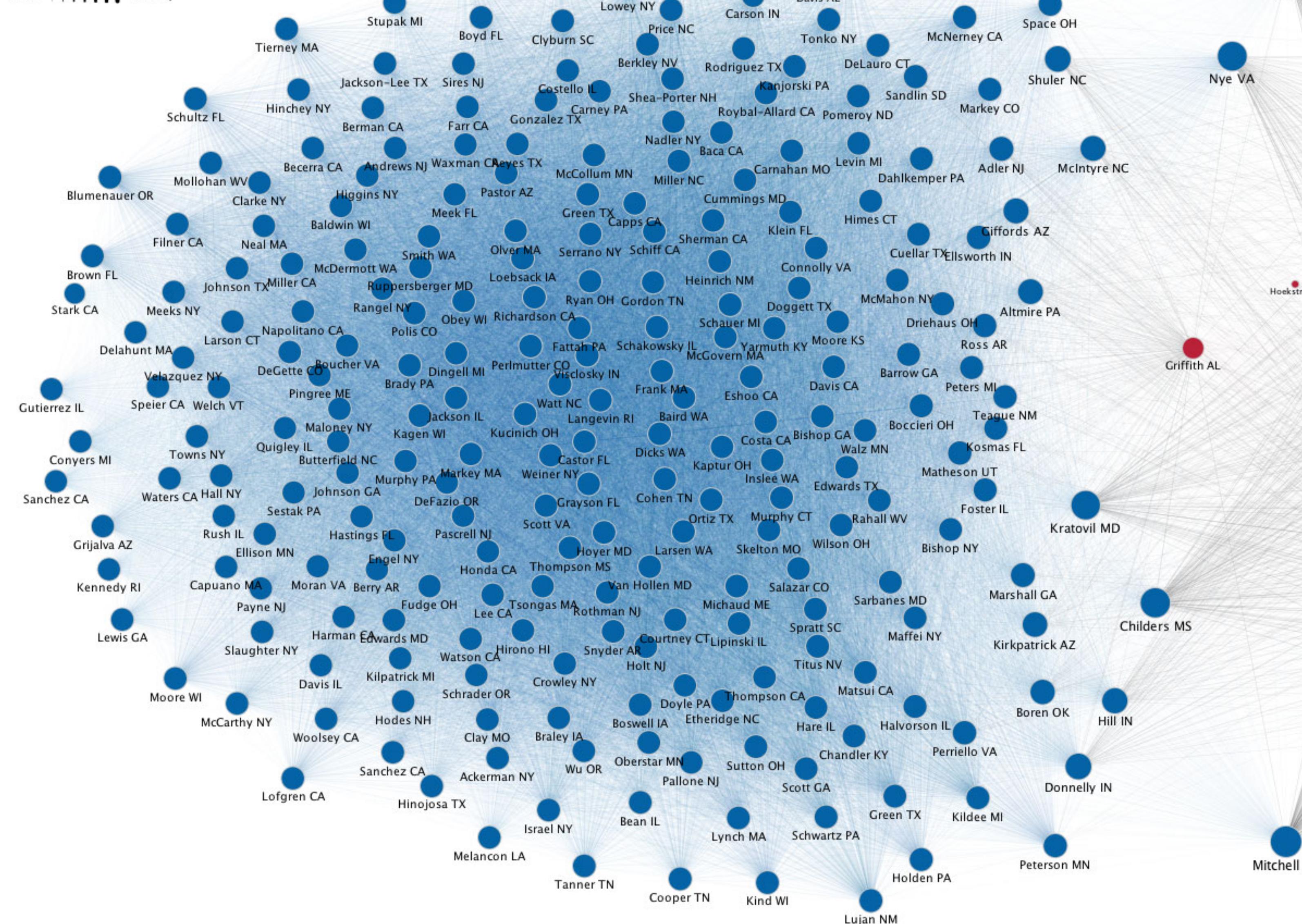


# Year: 2009

D-D    D-R    R-R

Degree 1 o 400

Few ||| Many

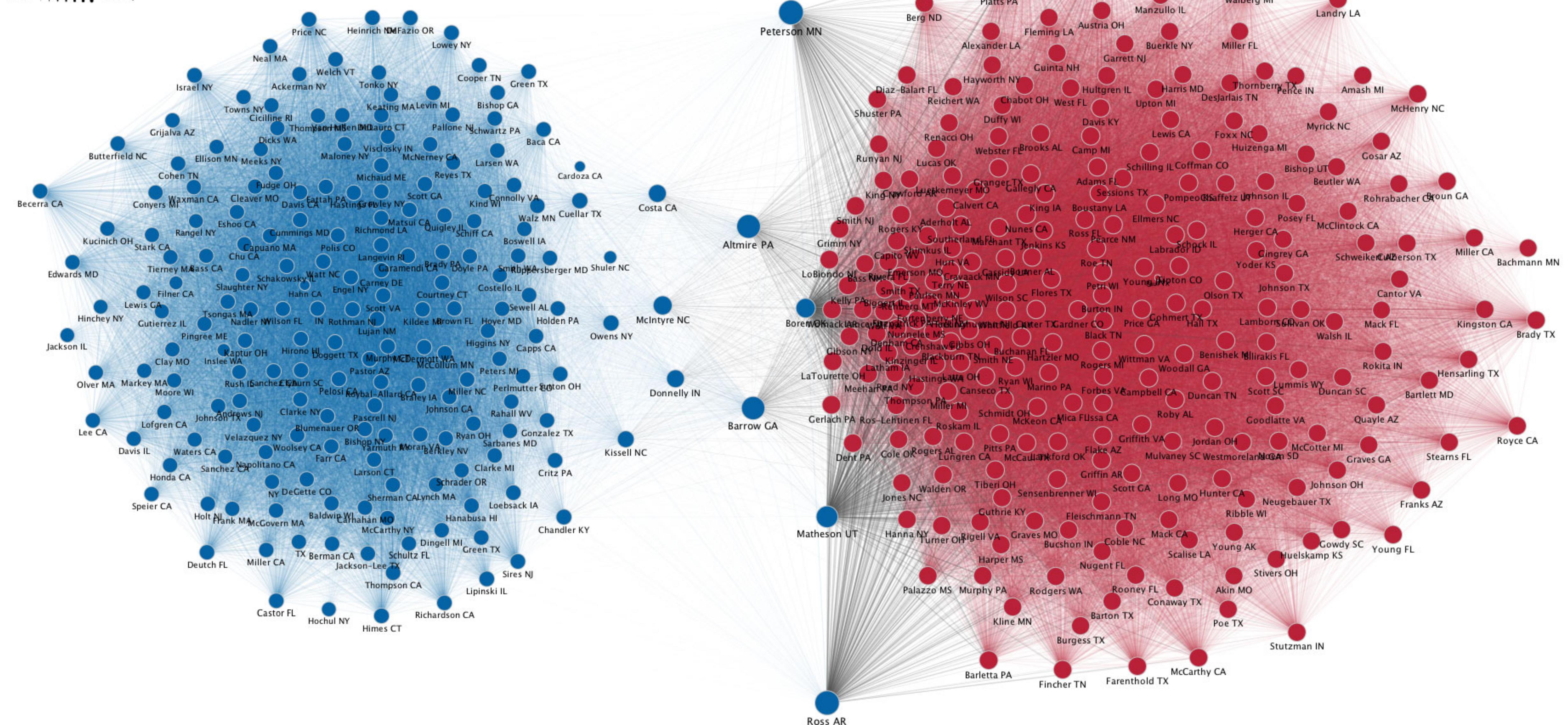


# Year: 2011

D-D    D-R    R-R

Degree 1 o 400

Few ||| Many



## Part Two

---

### What are social networks?

---

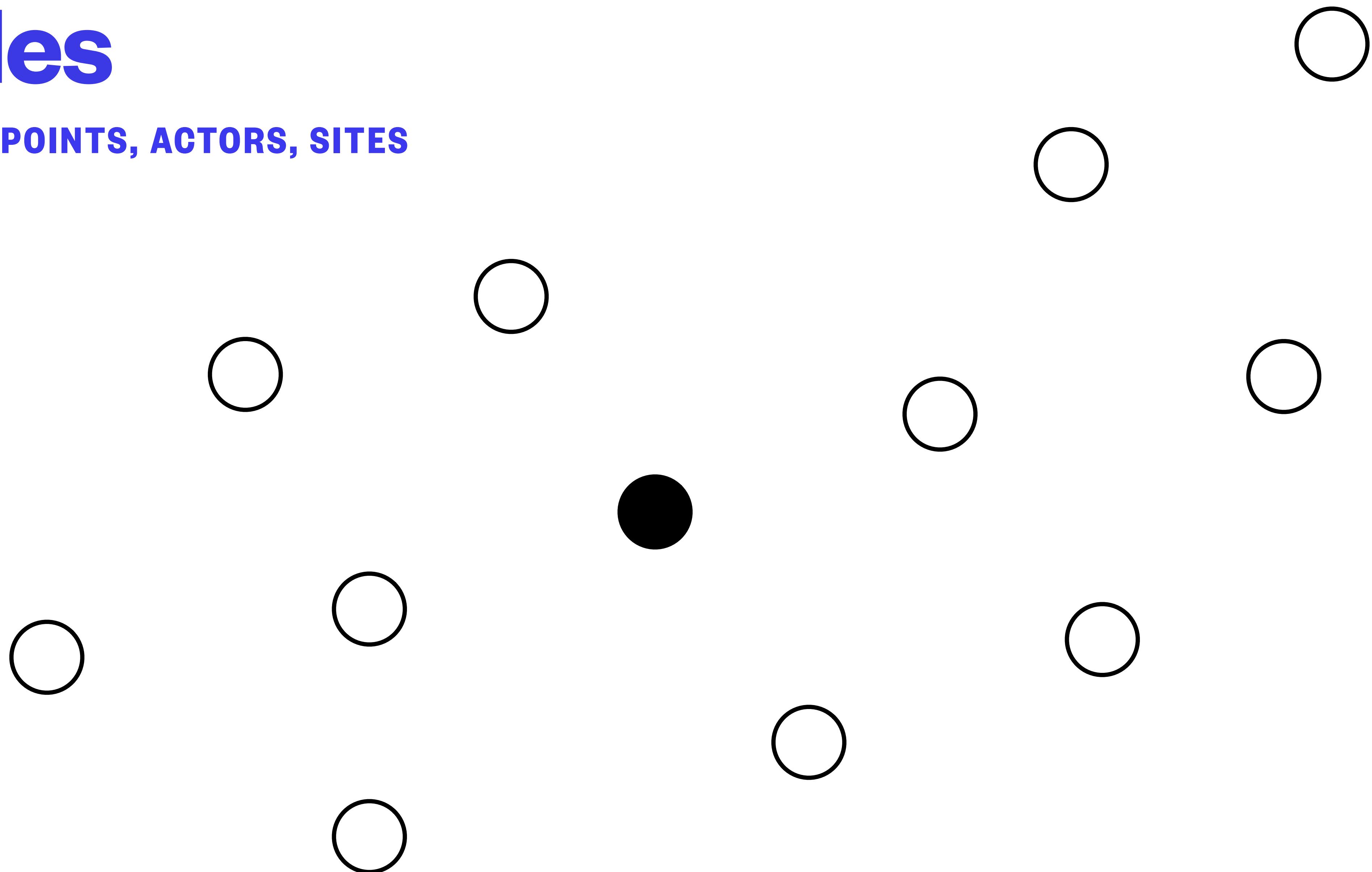
# Networks have two basic elements:

## Nodes & Edges

- Let's take an imaginary ego-centric friendship network to illustrate
  - Ego-centric: A network focused on a particular individual

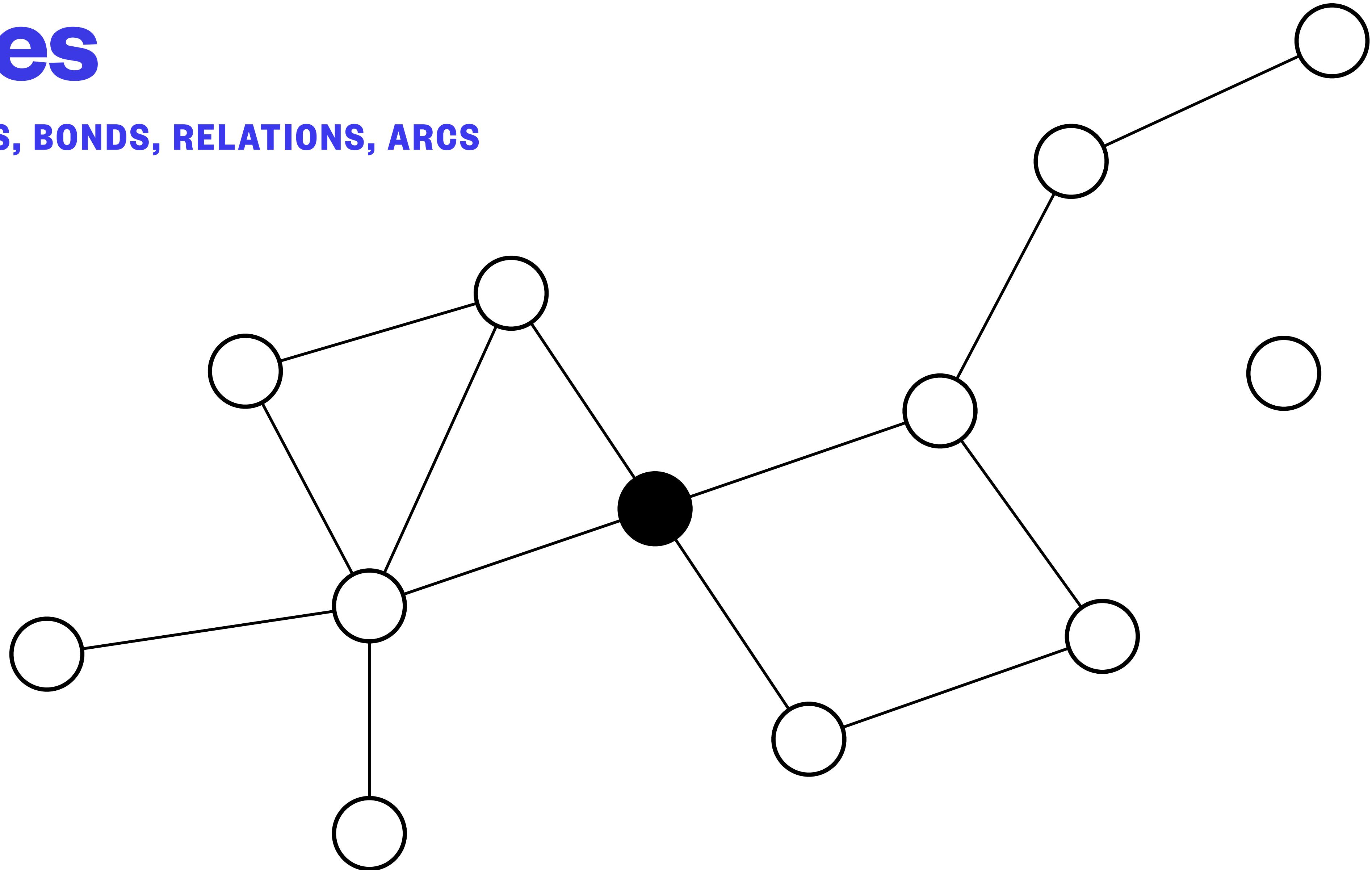
# Nodes

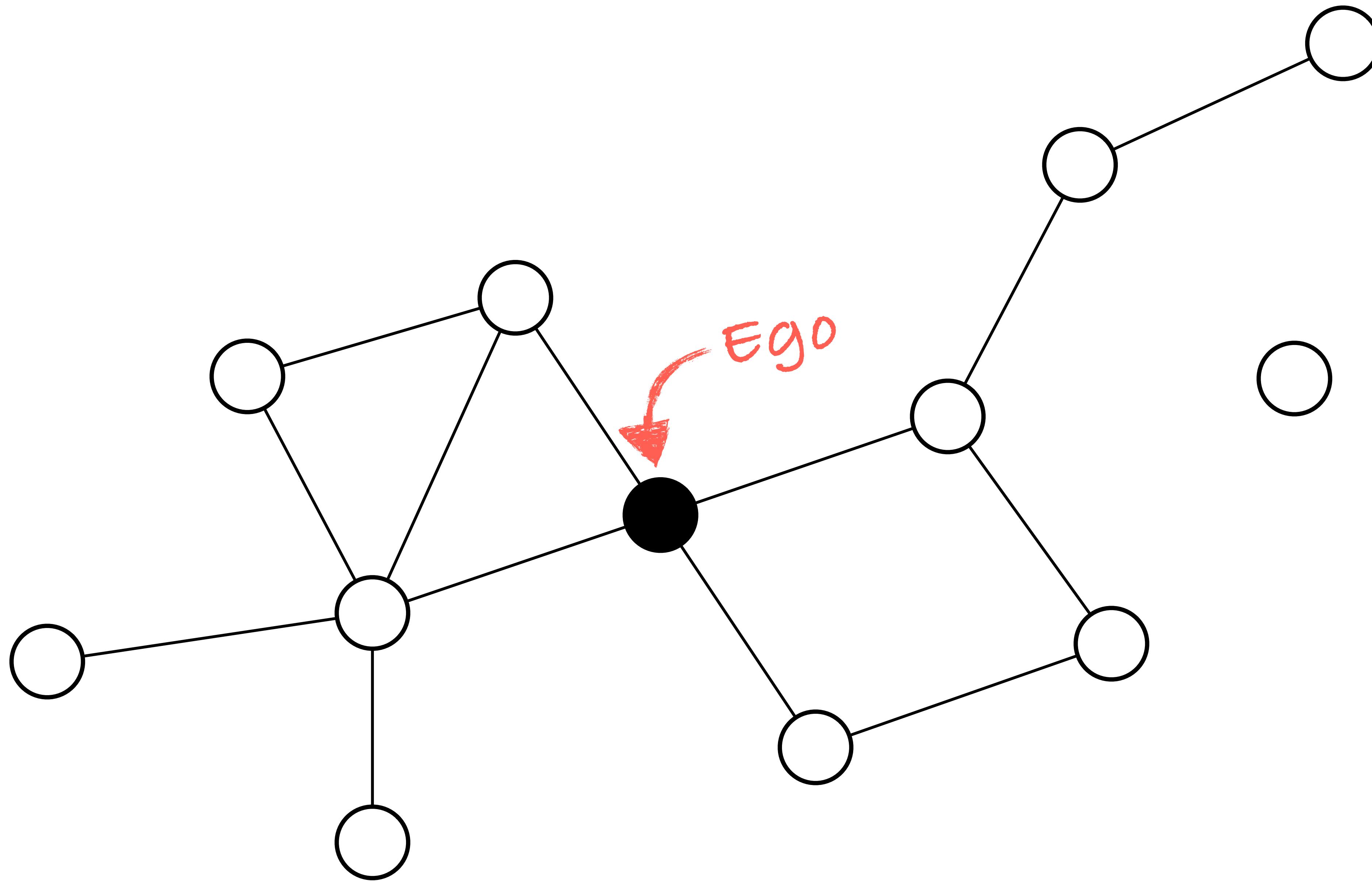
**VERTICES, POINTS, ACTORS, SITES**

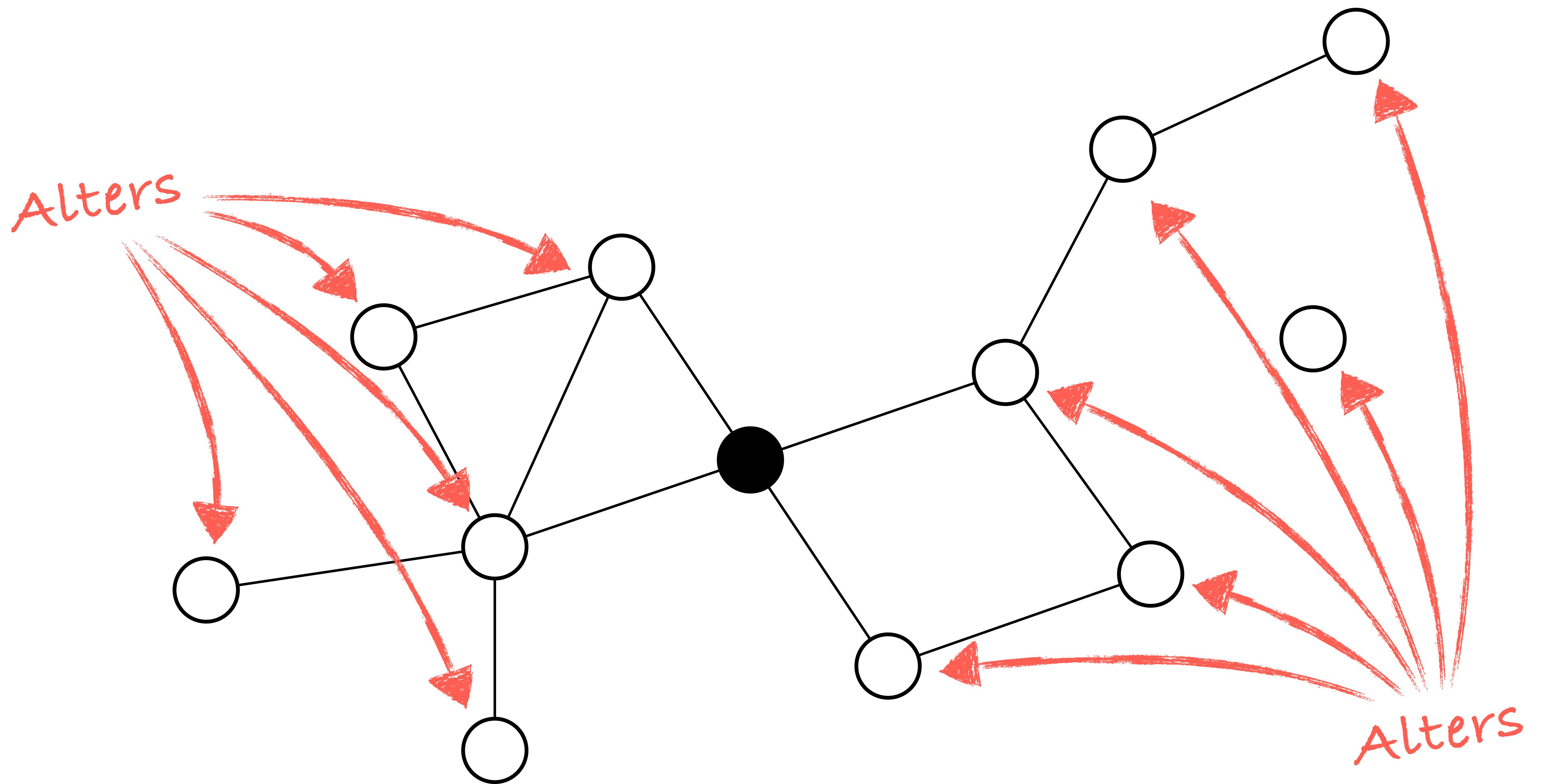


# Edges

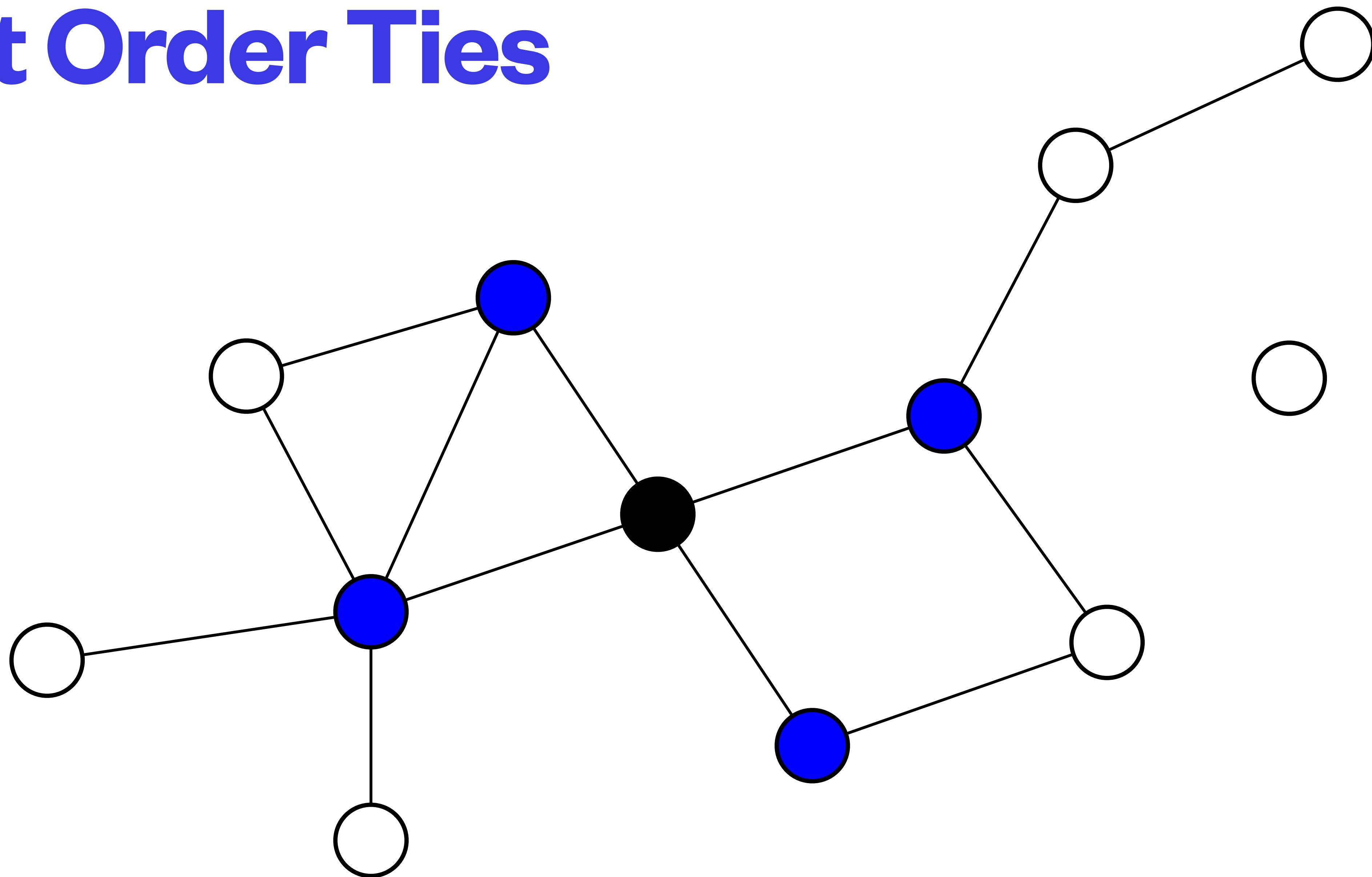
**TIES, LINKS, BONDS, RELATIONS, ARCS**





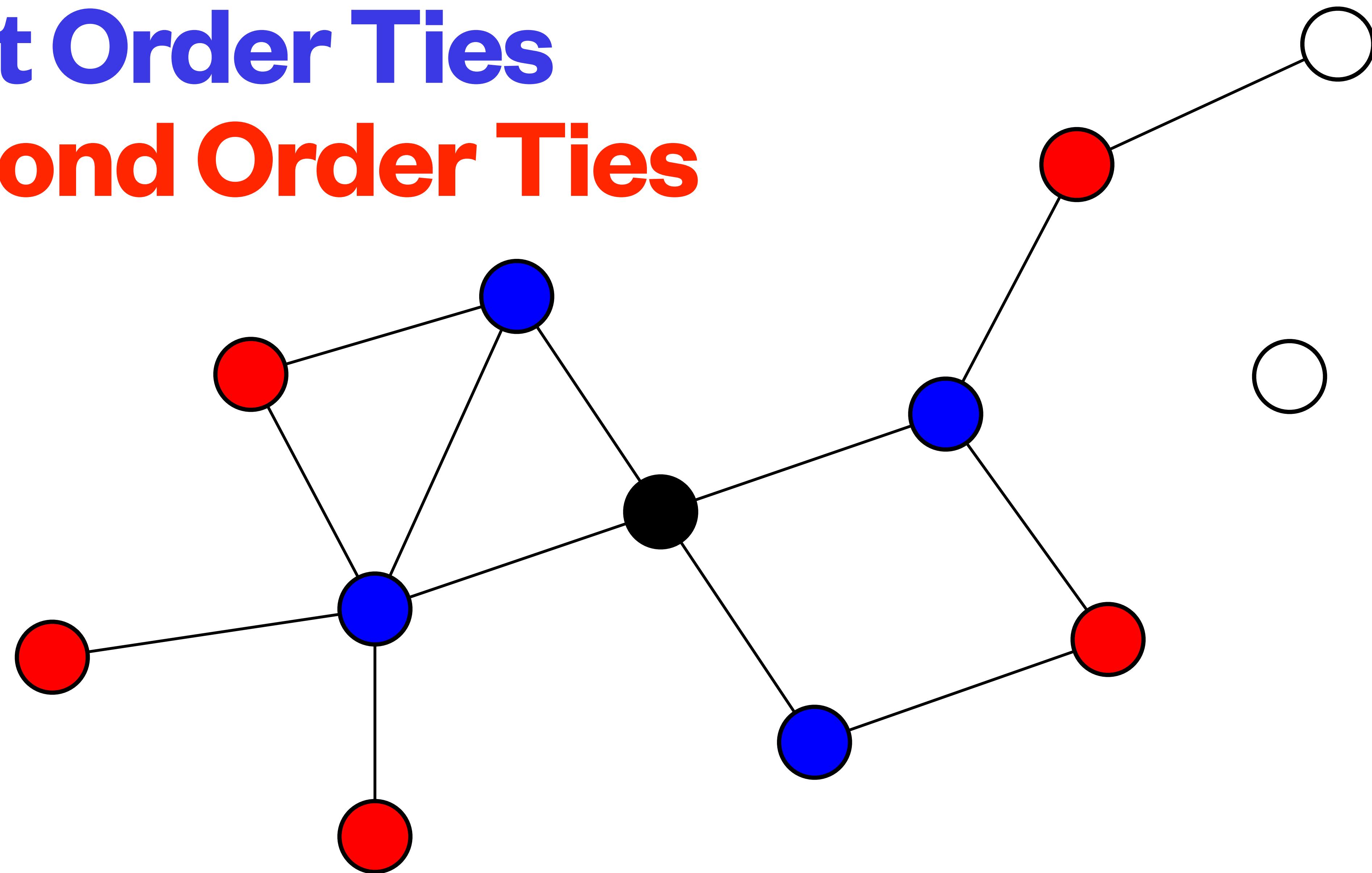


# First Order Ties

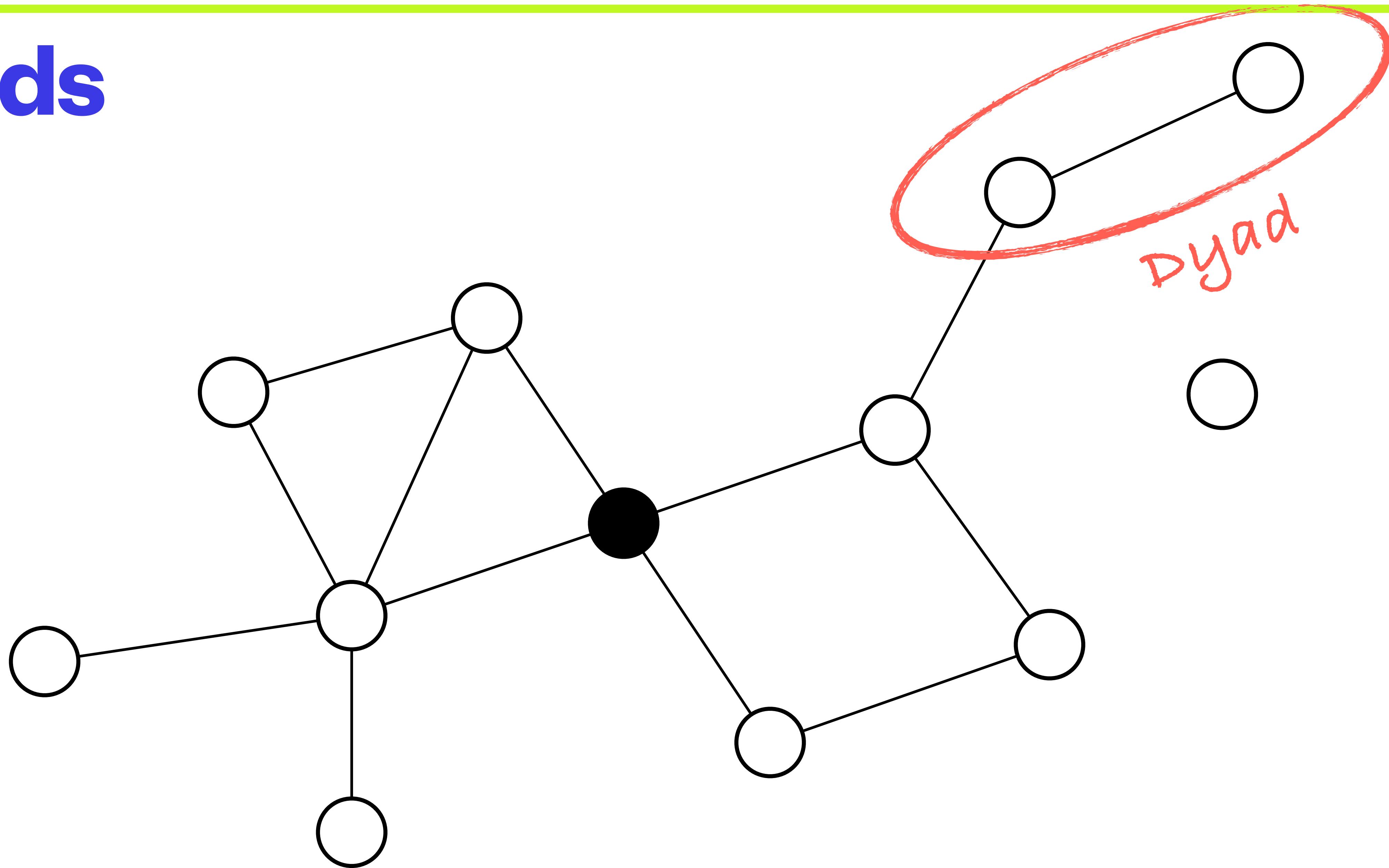


# First Order Ties

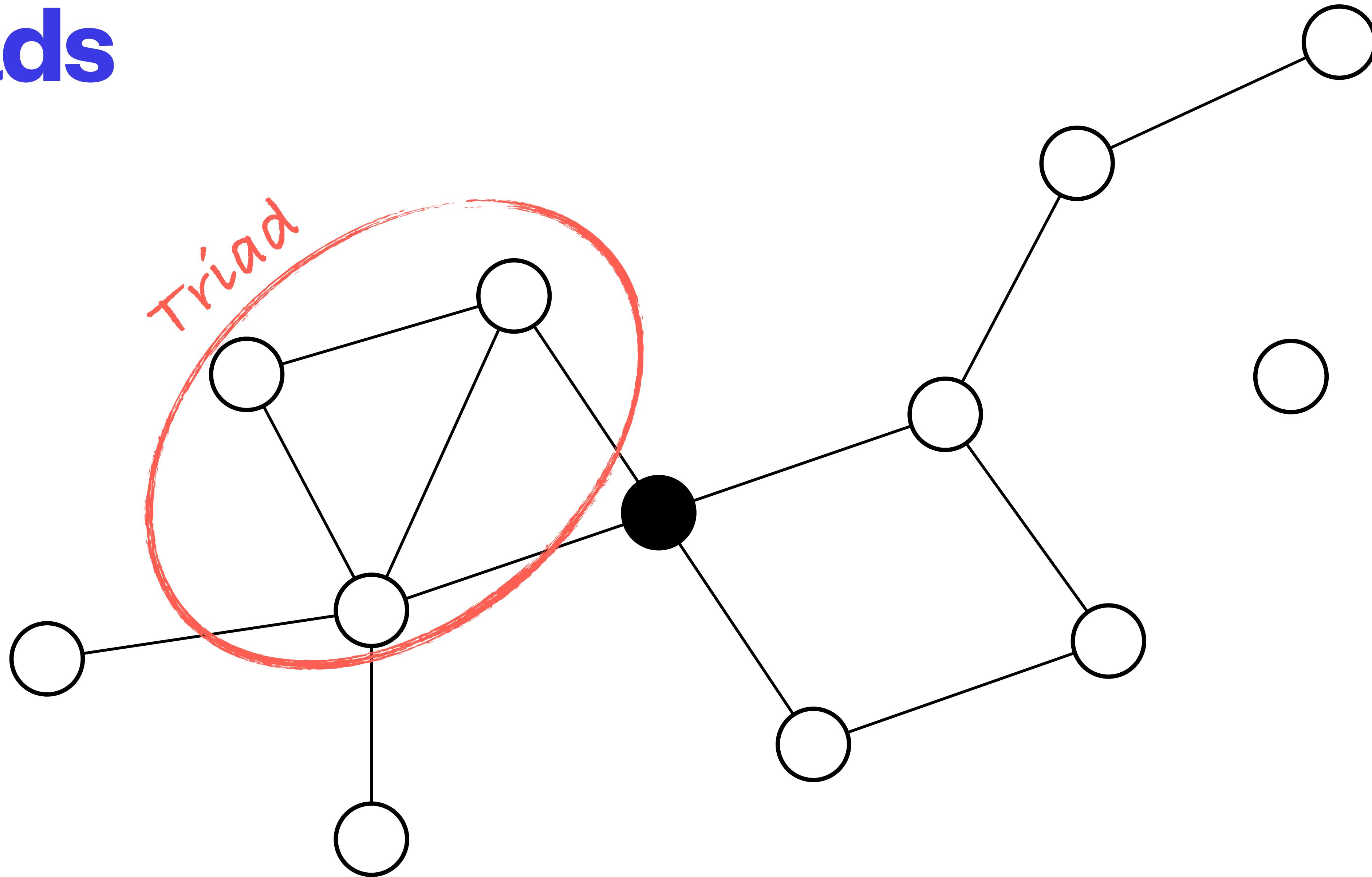
## Second Order Ties



# Dyads

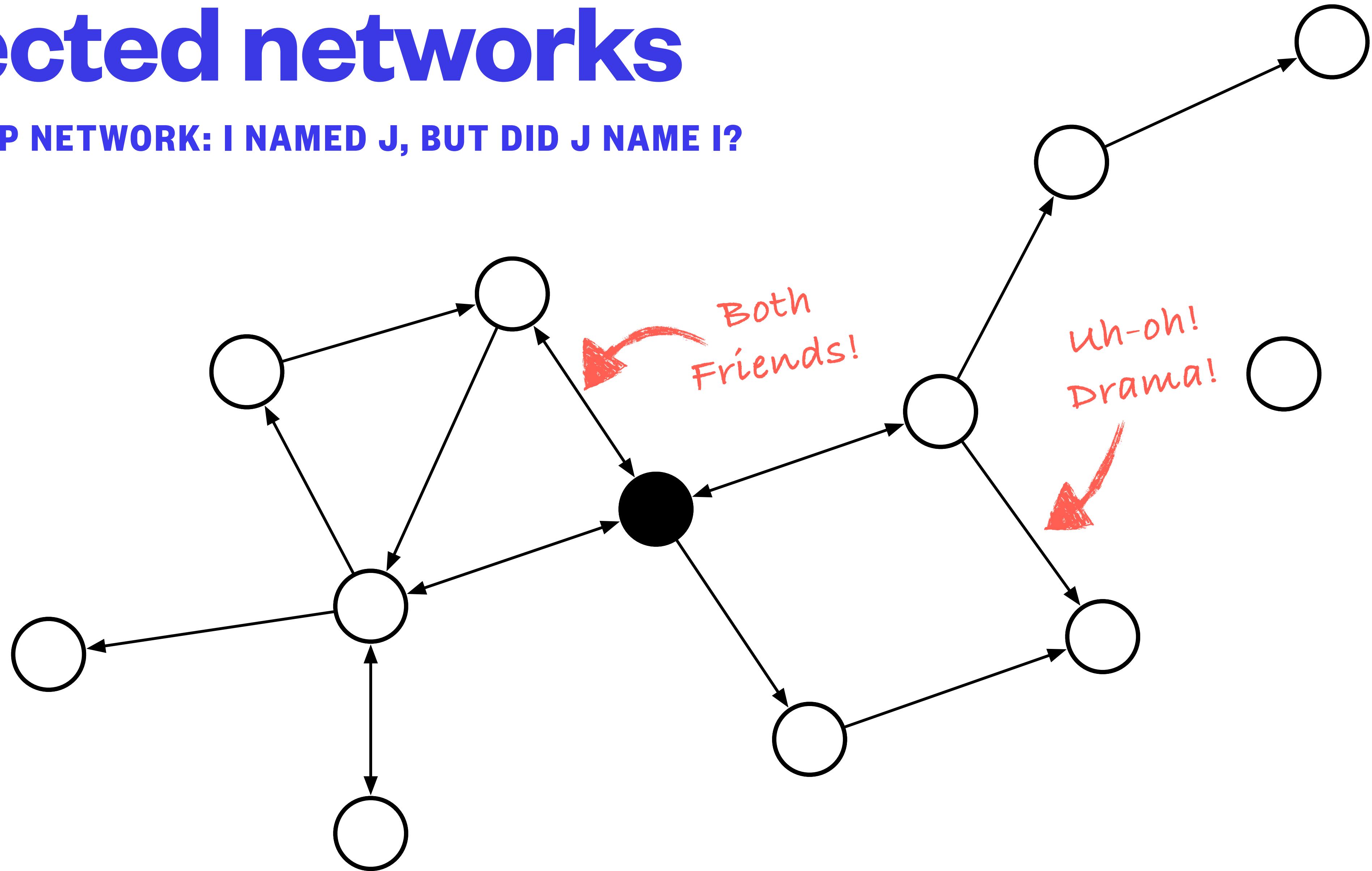


# Triads



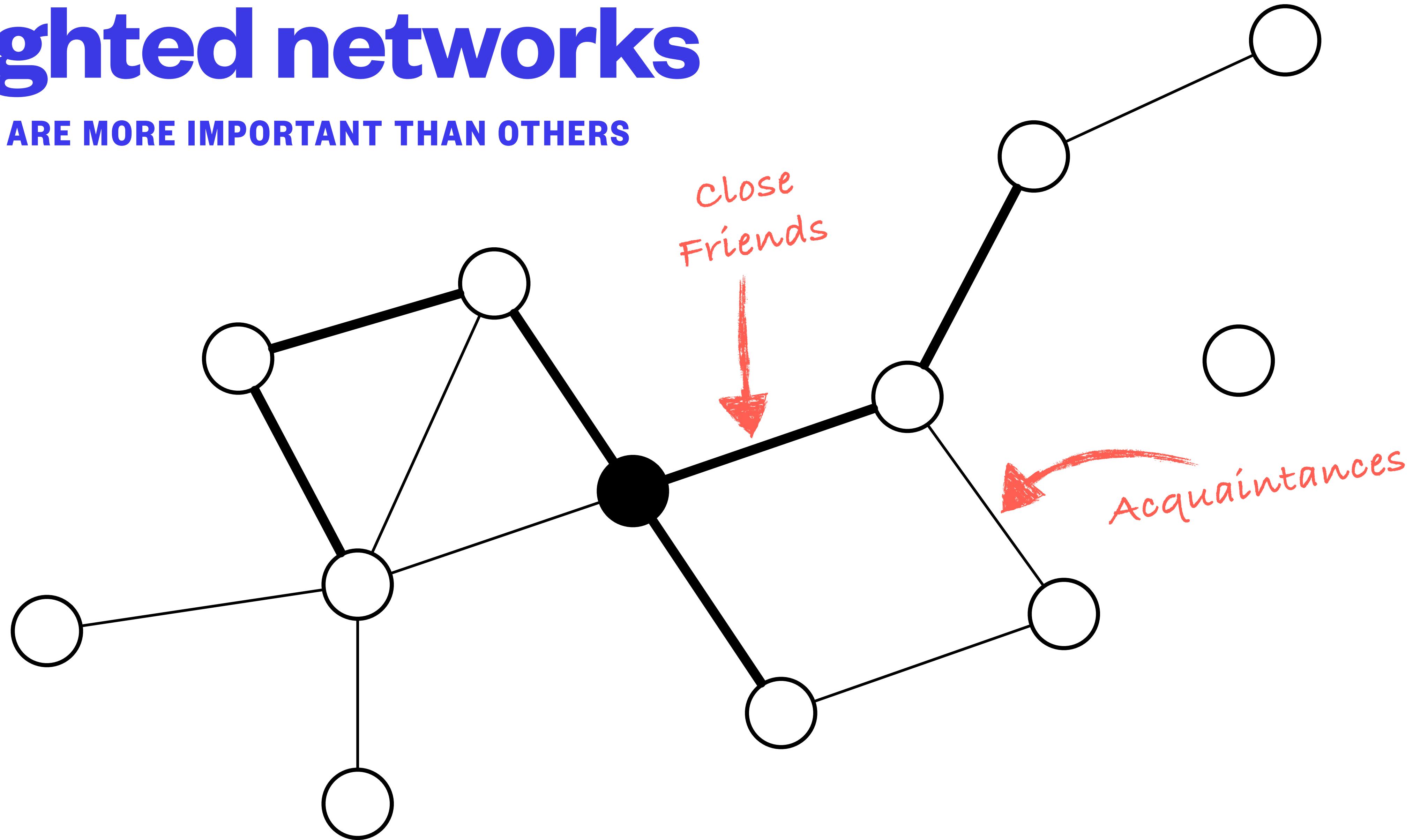
# Directed networks

FRIENDSHIP NETWORK: I NAMED J, BUT DID J NAME I?



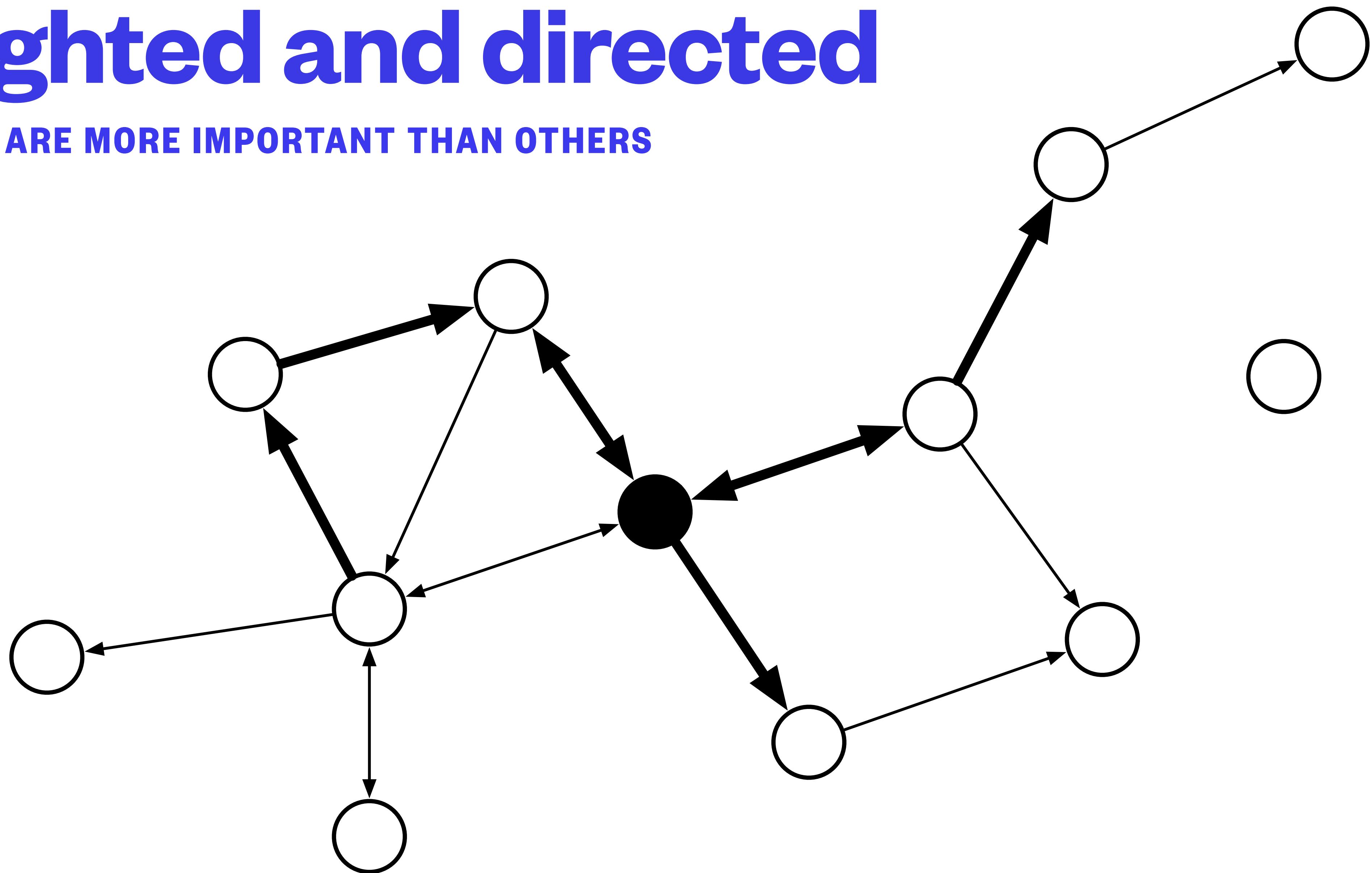
# Weighted networks

SOME TIES ARE MORE IMPORTANT THAN OTHERS



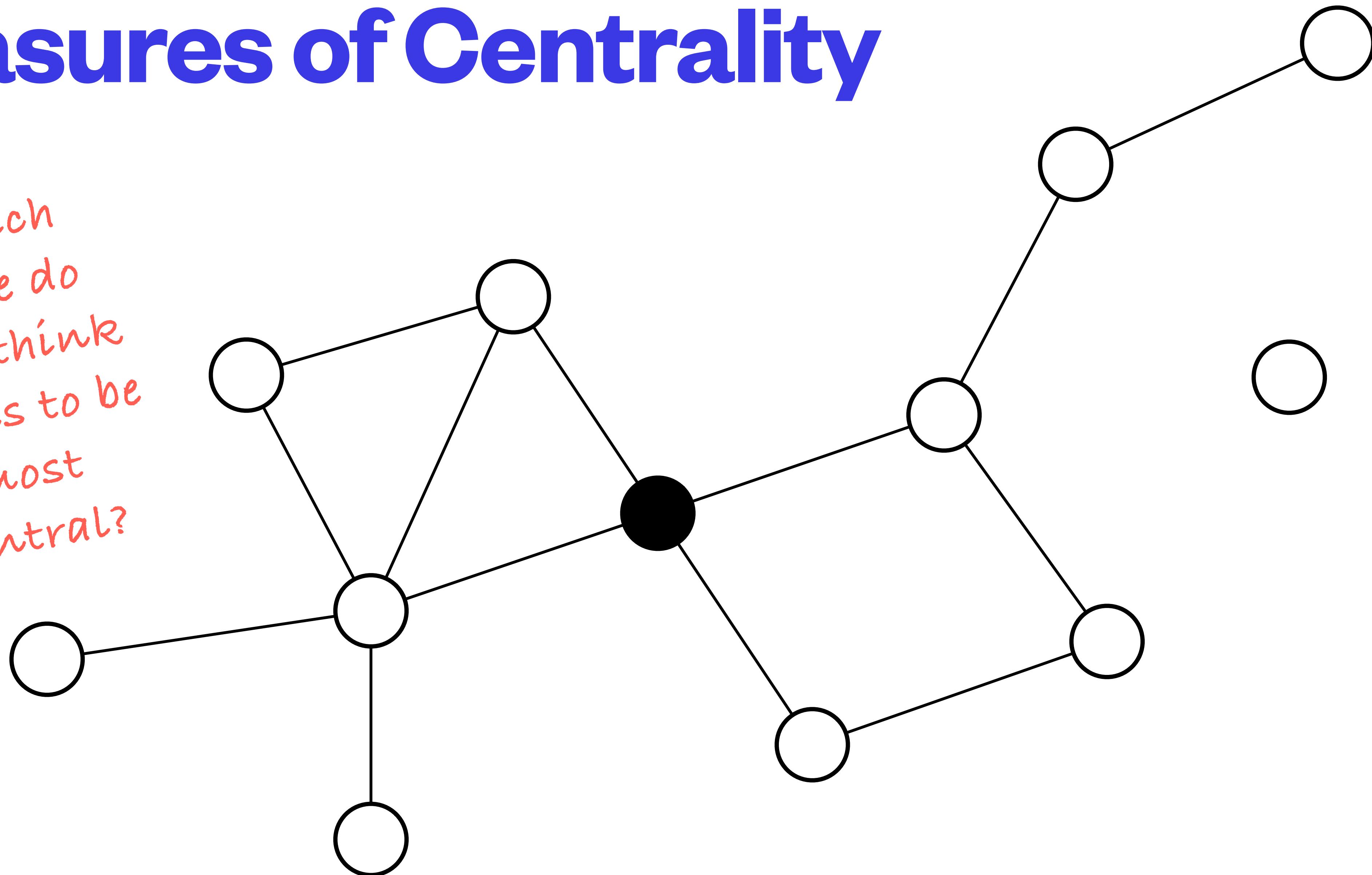
# Weighted and directed

SOME TIES ARE MORE IMPORTANT THAN OTHERS



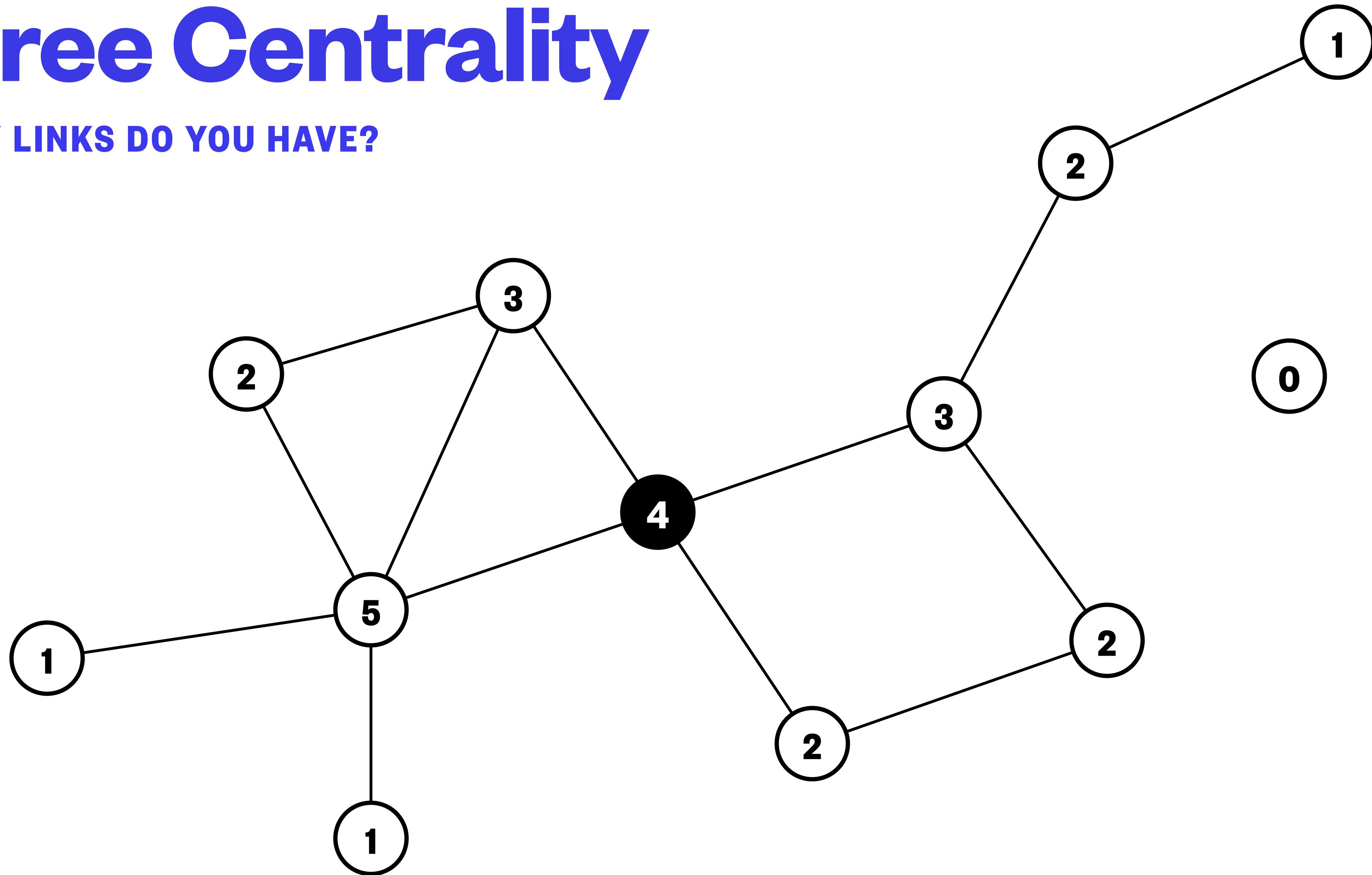
# Measures of Centrality

which node do you think seems to be most central?



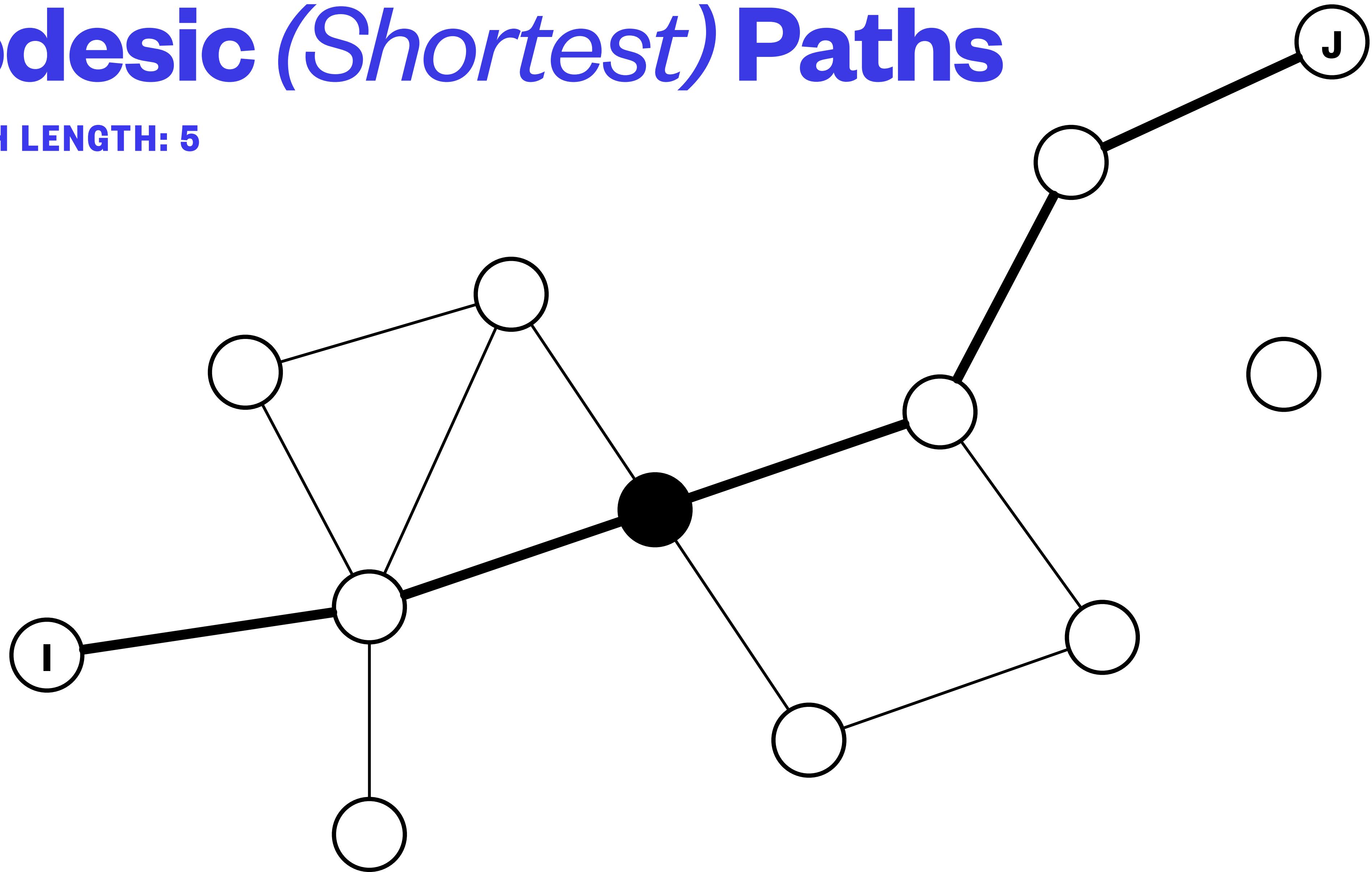
# Degree Centrality

HOW MANY LINKS DO YOU HAVE?



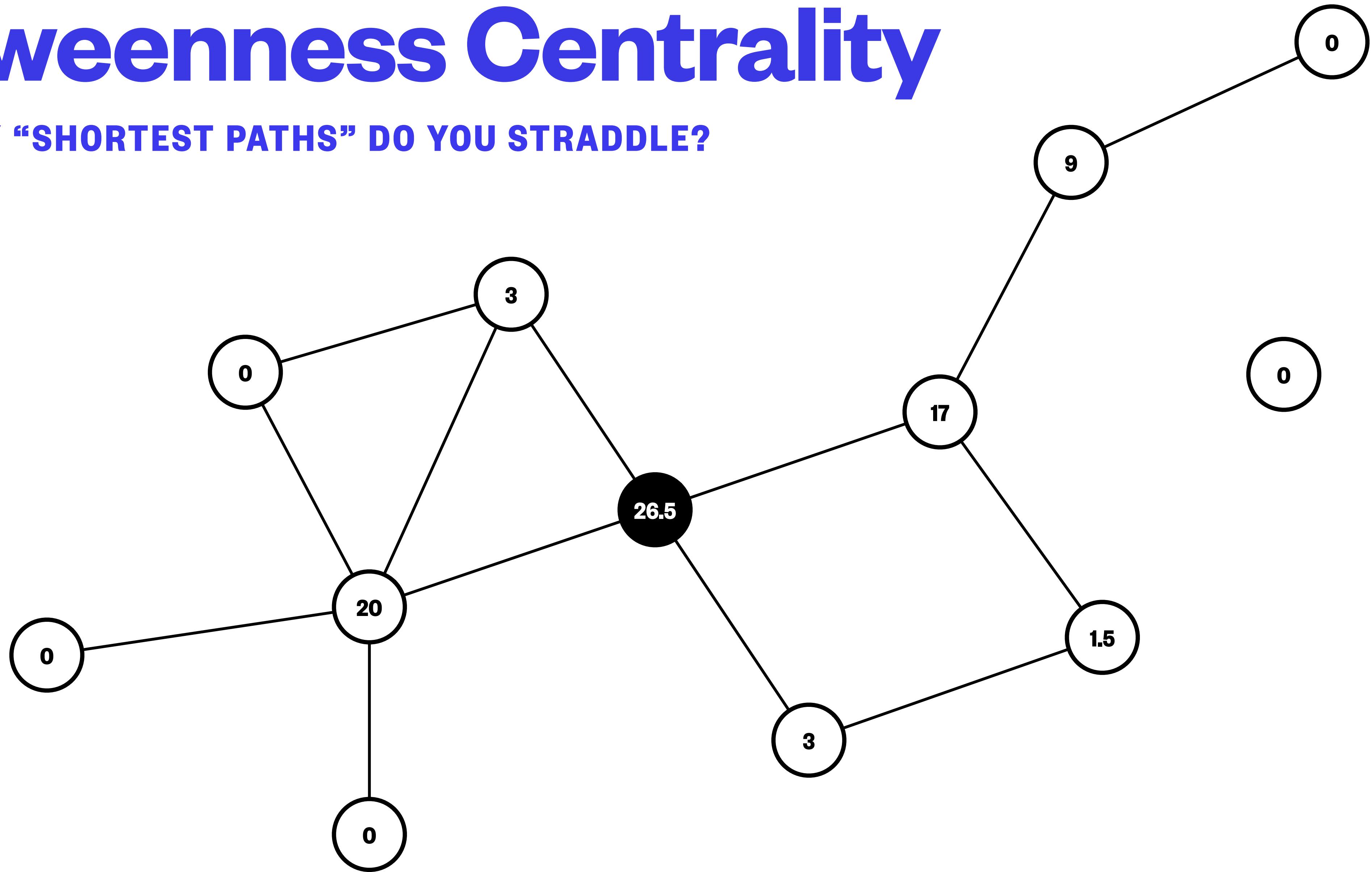
# Geodesic (Shortest) Paths

I TO J PATH LENGTH: 5



# Betweenness Centrality

HOW MANY “SHORTEST PATHS” DO YOU STRADDLE?



# Networks language

- We now know how to talk about networks
  - Main elements (nodes, edges)
  - Core variations (direction, weight)
  - Descriptors of components (dyads, triads)
  - Basic statistics (degree, centrality)
- We're ready

## Part Three

---

# How do we visualize networks?

---

# Network Data

- The core element of a network is defining the **relationship(s) between the actors**
- Two main datasets to any network:
  - Data defining attributes of the **nodes**
  - Data defining attributes of the **edges**
- Let's motivate with a simple example: the network of agreement among members of the Supreme Court in the 2020-2021 term
  - <https://harvardlawreview.org/supreme-court-statistics/>

# Node data: the node list

**supreme\_nodelist.csv**

<b>id</b>	<b>justice</b>	<b>abbrev</b>	<b>type</b>	<b>party</b>	<b>president</b>	<b>year</b>	<b>seniority</b>
j01	Roberts	JR	Chief	Republican	Bush Jr	2005	16
j02	Thomas	CT	Associate	Republican	Bush Sr	1991	30
j03	Breyer	SB	Associate	Democrat	Clinton	1994	27
j04	Alito	SA	Associate	Republican	Bush Jr	2006	15
j05	Sotomayor	SS	Associate	Democrat	Obama	2009	12
j06	Kagan	EK	Associate	Democrat	Obama	2010	11
j07	Gorsuch	NG	Associate	Republican	Trump	2017	4
j08	Kavanaugh	BK	Associate	Republican	Trump	2018	3
j09	Barrett	ACB	Associate	Republican	Trump	2020	1

- A pretty ordinary looking dataset, similar to what we'd see in cross-sectional analysis
- Each row corresponds to a justice, each column corresponds to an attribute of the justice

# Edge data: the edge list

**supreme\_edgelist.csv**

from	to	type	weight
j01	j02	agreement	0.463
j01	j03	agreement	0.488
j01	j04	agreement	0.625
j01	j05	agreement	0.366
j01	j06	agreement	0.463
j01	j07	agreement	0.512
j01	j08	agreement	0.927
j01	j09	agreement	0.743
j02	j03	agreement	0.293
j02	j04	agreement	0.525
j02	j05	agreement	0.171
j02	j06	agreement	0.341

- Each row corresponds to an edge/link in the dataset
  - IDs link the node list and edge list
- Each column is an attribute of the edge
- This network is *undirected* (the agreement between j01 and j02 is the same as the agreement between j02 and j01)
  - Only a single row per relation
  - A *directed* network would have not just a j01-j02 row, but a j02-j01 row as well

# More edge data: the adjacency matrix

`supreme_adjacency.csv`

	j01	j02	j03	j04	j05	j06	j07	j08	j09
j01	.	0.463	0.488	0.625	0.366	0.463	0.512	0.927	0.743
j02	.	.	0.293	0.525	0.171	0.341	0.707	0.488	0.600
j03	.	.	.	0.200	0.805	0.854	0.366	0.537	0.429
j04	.	.	.	.	0.125	0.175	0.600	0.600	0.588
j05	.	.	.	.	.	0.805	0.244	0.415	0.314
j06	.	.	.	.	.	.	0.415	0.512	0.486
j07	.	.	.	.	.	.	.	0.561	0.686
j08	.	.	.	.	.	.	.	.	0.800
j09	.	.	.	.	.	.	.	.	.

- Each cell corresponds to an edge/link in the dataset
  - The weight of each edge is given by the value of the cell
- This network is *undirected* (the agreement between j01 and j02 is the same as the agreement between j02 and j01)
  - Only a single cell per relation
  - A *directed* network would have not just a j01-j02 cell, but a j02-j01 cell as well
- Unlike an edge list, only one attribute per link

# Using R (and iGraph) for Social Networks

- R's advantages:
  - Easily integrates with the rest of a data workflow
  - More programmatically flexible (for the programmatically inclined) than GUIs
  - Reproducible
  - Free!
- R's “disadvantages”:
  - Needs to be scripted/programmed
  - Network visualization not built in; must use additional libraries, which themselves have different strengths and weaknesses
- **iGraph** vs. **Statnet**; assorted others for interactive and dynamic networks

# Loading the data

```
#Load library
library("igraph")

#Load nodelist and edgelist as objects
nodes <- read.csv("supreme_nodes.csv", header=T, as.is=T)
links <- read.csv("supreme_edgelist.csv", header=T, as.is=T)

#Convert to an iGraph network. "d" identifies the edgelist; "vertices" identifies the nodelist
net <- graph_from_data_frame(d=links, vertices=nodes, directed=F)

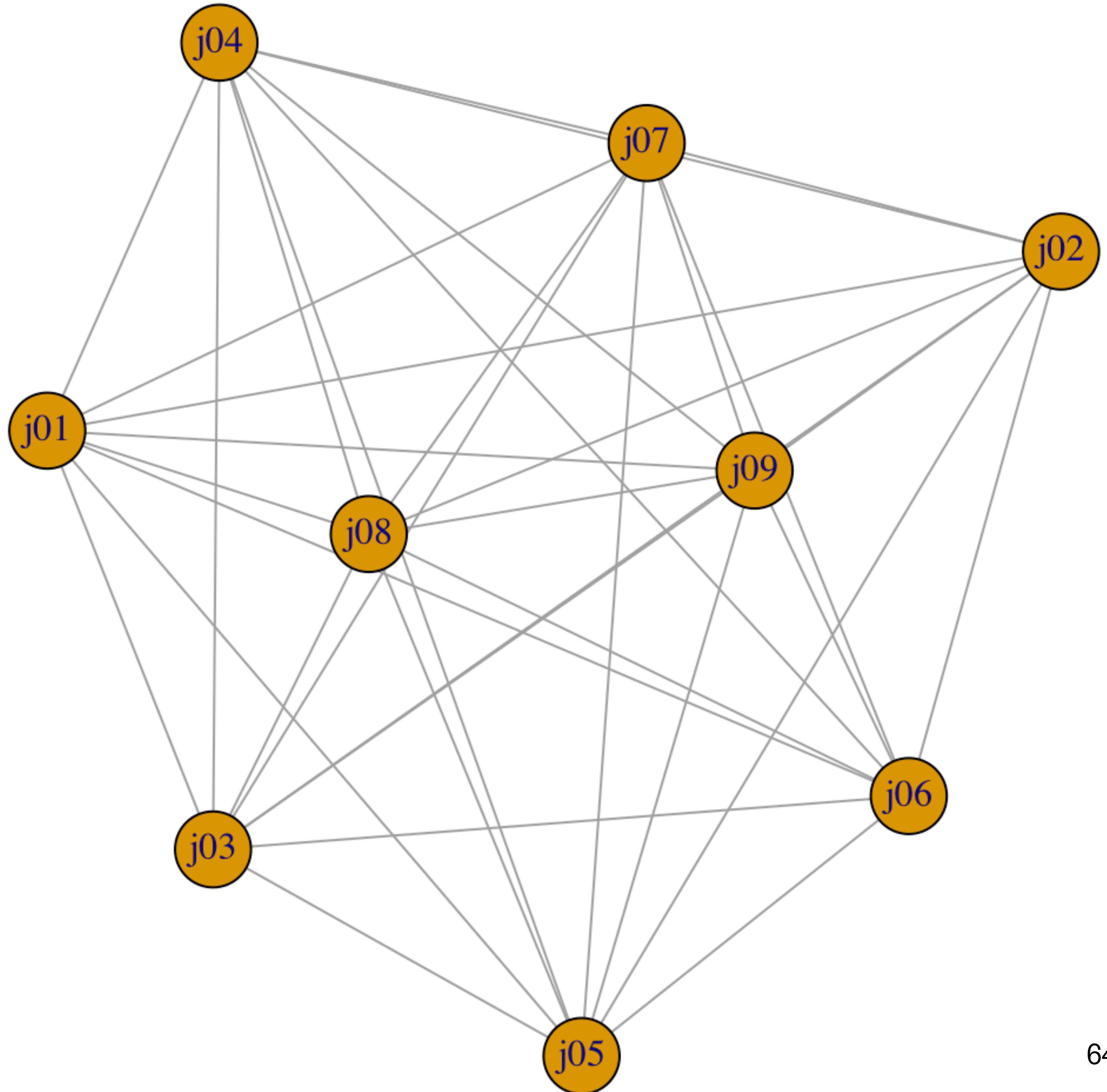
#If we want to output an adjacency matrix, we can
#(we just need to identify the attribute to encode)
adjacency<-as_adjacency_matrix(net, attr="weight")
```

# Working with the data

Command	Usage
E(net)	Access the edges of the network
V(net)	Access the nodes of the network
E(net)\$weight	Access a particular attribute (weight) of the edges of the network
V(net)\$seniority	Access a particular attribute (seniority) of the nodes of the network
V(net)[president=="Obama"]	Returns all vertices with appointing president "Obama"
E(net)[weight>.9]	Returns all edges with weight greater than .9
net[1, ]	Returns first row of network matrix
net[5, 7]	Returns cell in the fifth row, seventh column
plot.igraph(net) <b>or</b> plot(net)	<b>Plots the network (!)</b>

# plot.igraph(net)

- And . . . ok, but, it's kind of . . .
  - uninspired
- What's wrong?
  - Links are overplotted
  - Links are undifferentiated
  - Node labels are uninspiring
  - Colors
- How should we fix it?

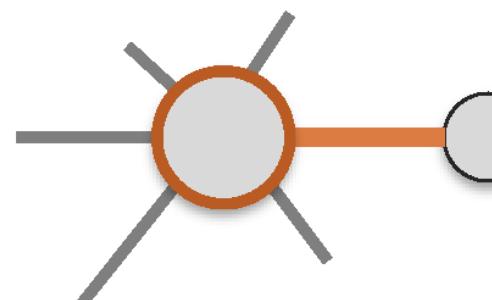


# Graphics guidelines

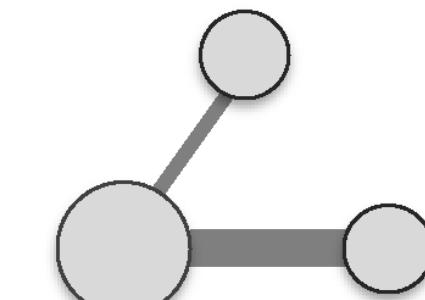
- Data does not speak for itself; visual attention must be properly drawn
- Like in regular graphics and visualization, plotting networks is a matter of taste
  - *Tastes can differ, but . . .*
- Healy (2020) identifies three common graphics challenges to consider as we set our visualization goals:
  - **Aesthetic**
  - **Substantive**
  - **Perceptual**

## Network visualization goals

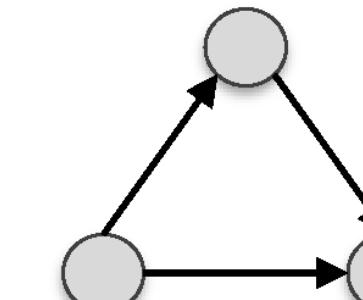
**Key actors and links**



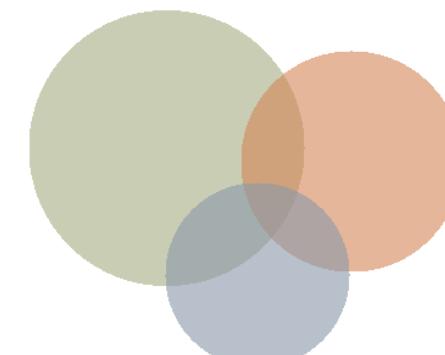
**Relationship strength**



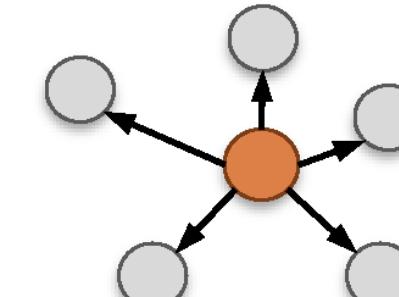
**Structural properties**



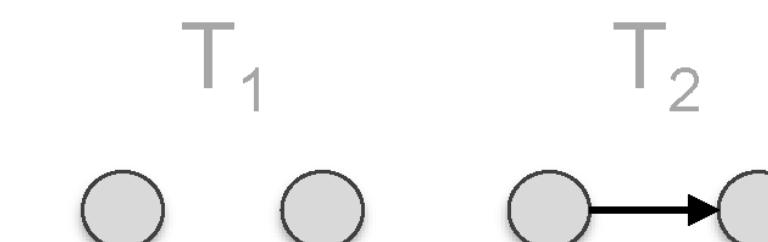
**Communities**



**Diffusion patterns**



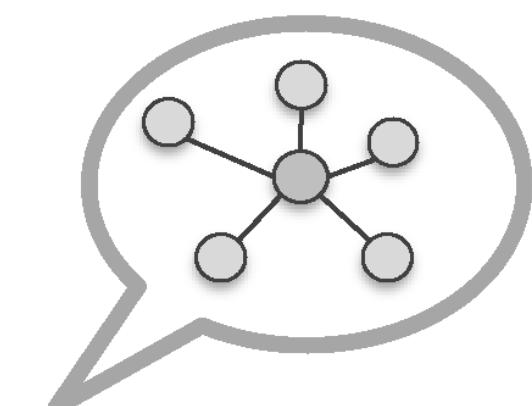
**Network evolution**



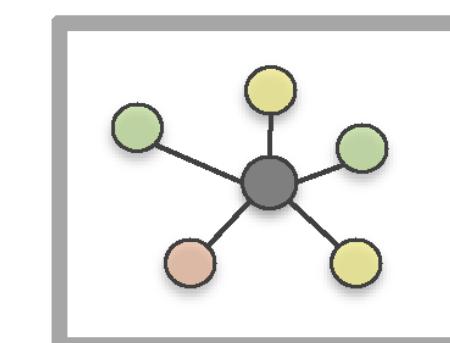
**Networks as maps**



**Networks as persuasion**

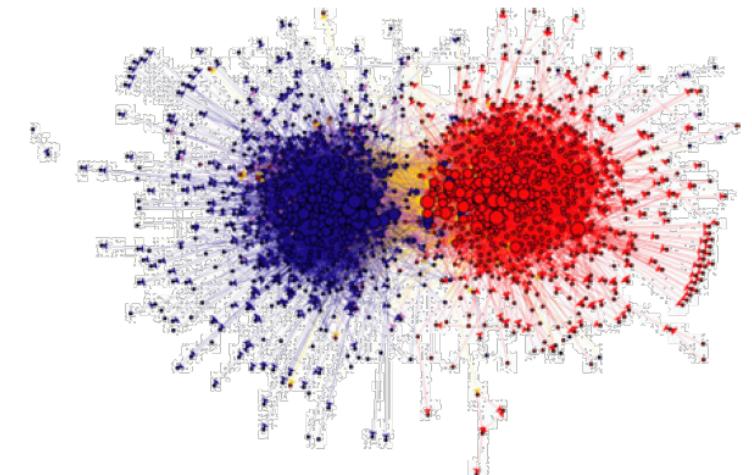


**Networks as art**

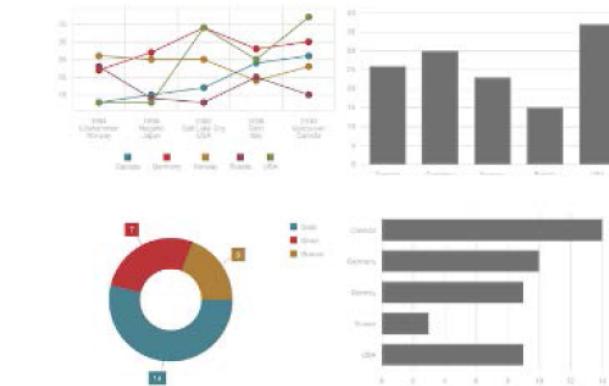


# Some network visualization types

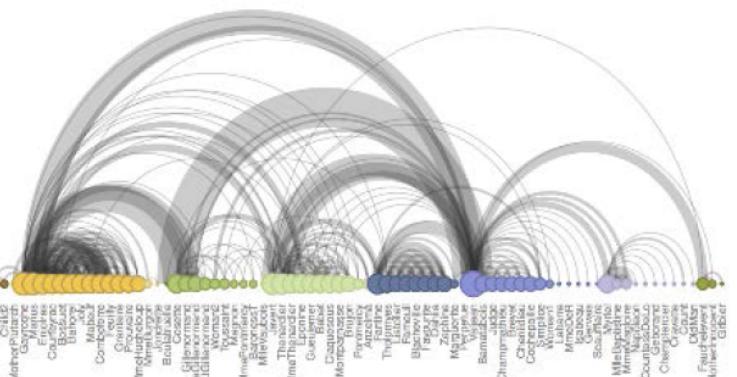
**Network Maps**



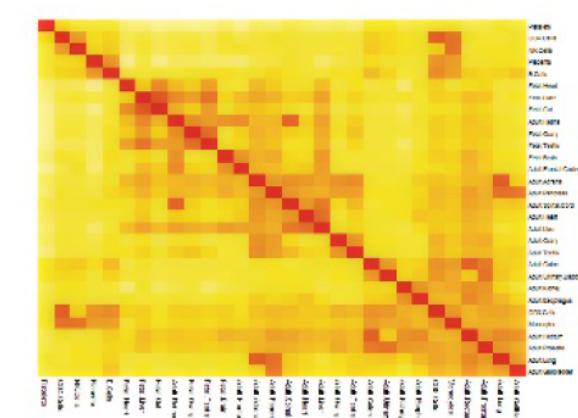
**Statistical charts**



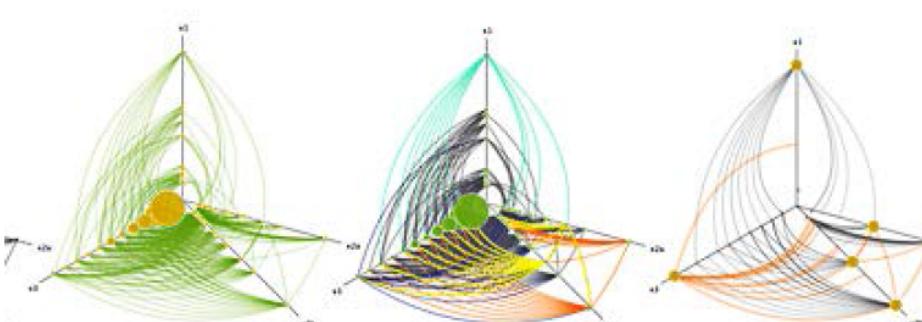
**Arc diagrams**



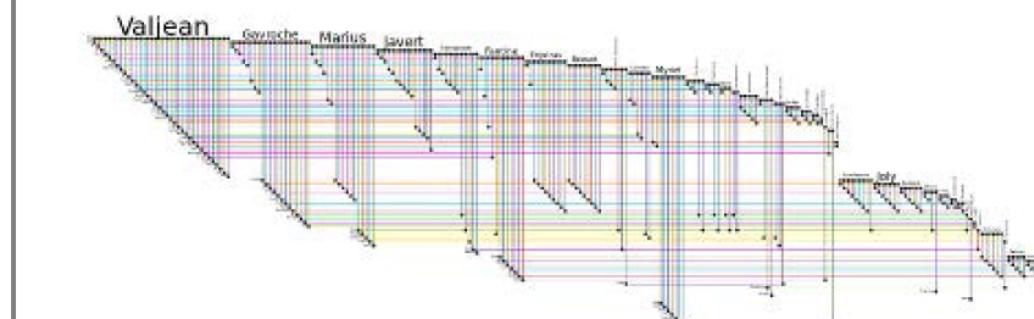
**Heat maps**



**Hive plots**

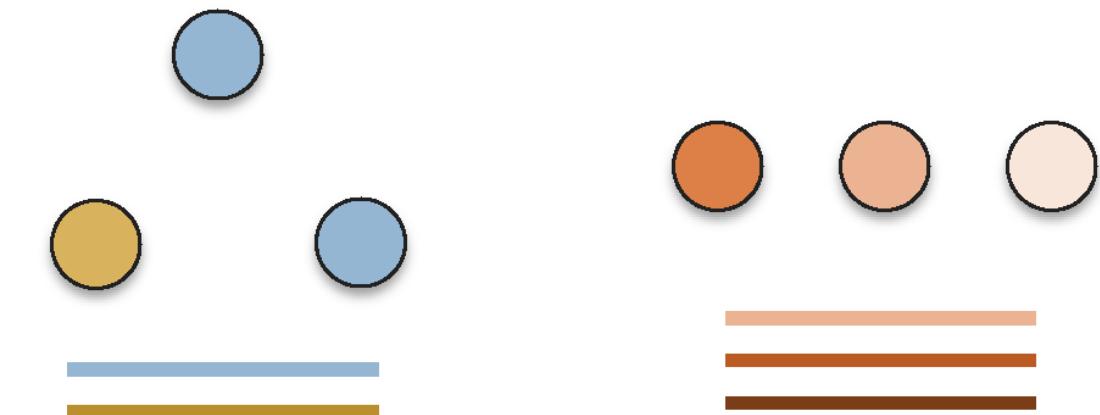


**Biofabric**

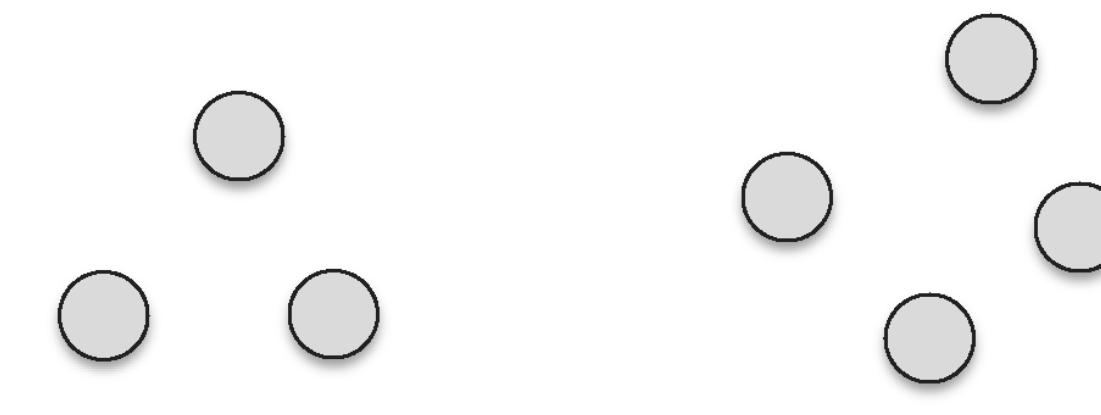


## Network visualization controls

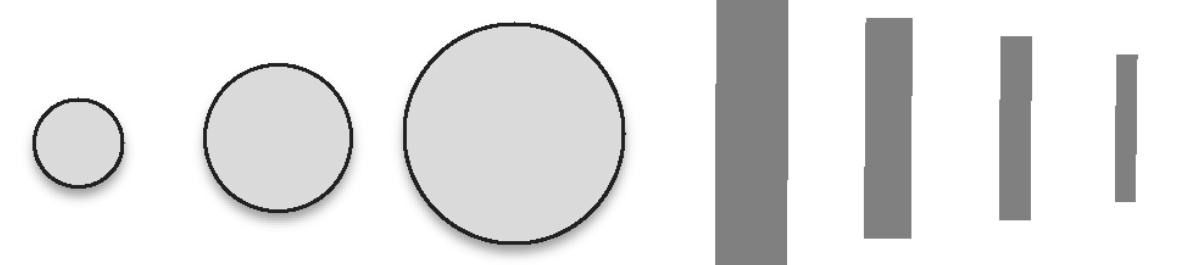
**Color**



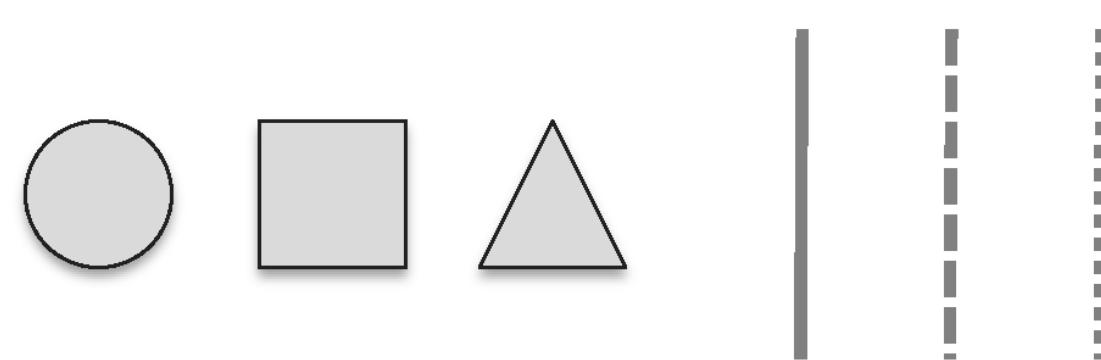
**Position**



**Size**



**Shape**

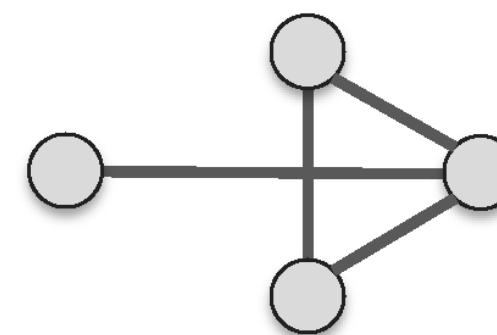


**Honorable mention:** arrows (direction) and labels (identification)

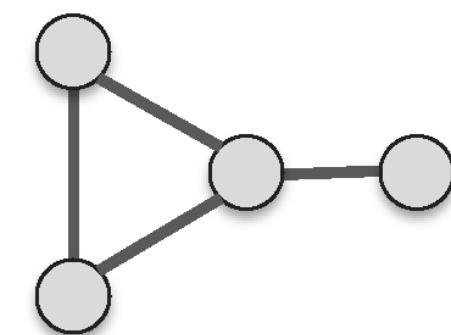
## Layout aesthetics

### Minimize edge crossing

No

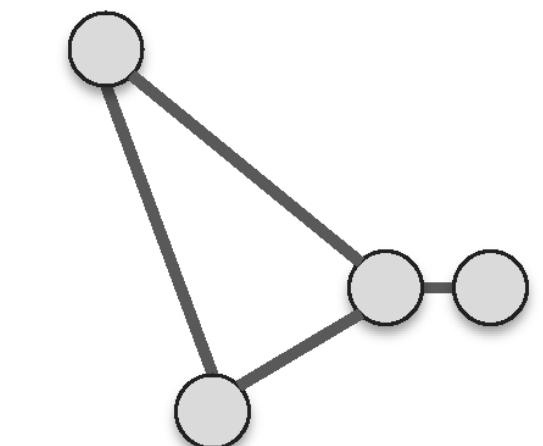


Yes

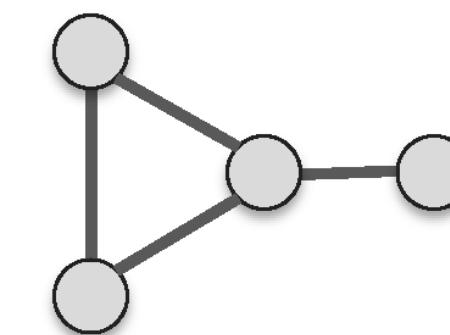


### Uniform edge length

No

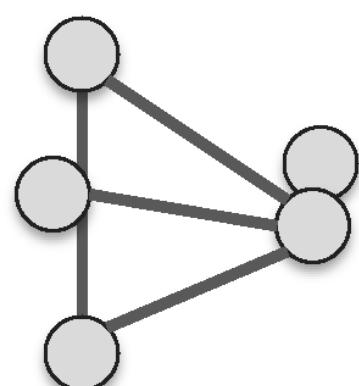


Yes

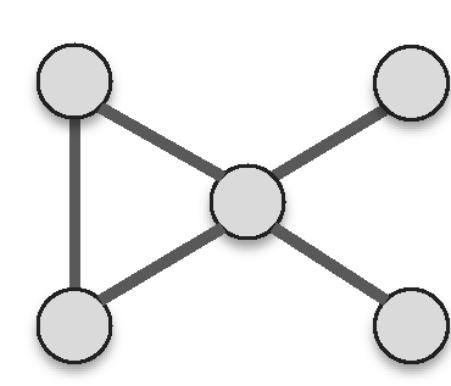


### Prevent overlap

No

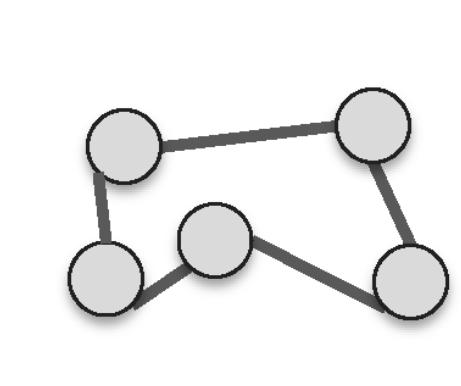


Yes

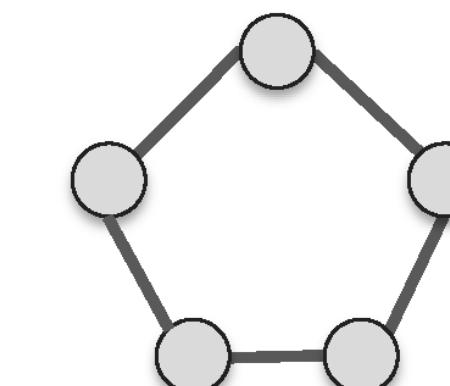


### Symmetry

No

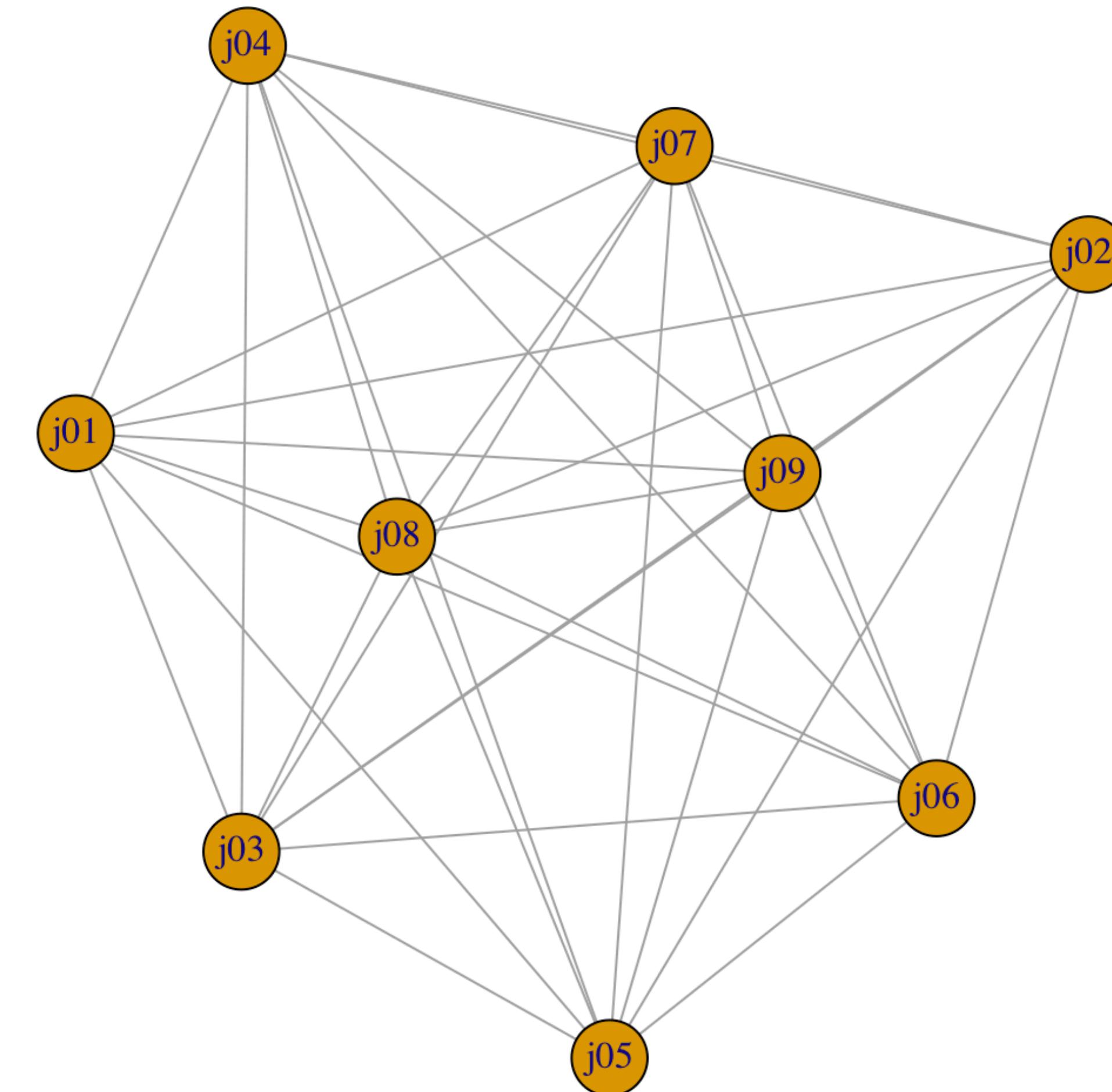


Yes



# Goals in the Supreme Court network

- Identify **who** the justices are
- Illustrate external **political factors** relevant to the Supreme Court
- Show who the most **central** (likely “decisive”) justices are
- Show **agreement** and **disagreement** among the justices



# Goals in the Supreme Court network

- Identify **who** the justices are → • *Label the nodes*
- Illustrate external **political factors** relevant to the Supreme Court → • *Color the nodes to show party of appointment*
- Show who the most **central** (likely “decisive”) justices are → • *Calculate centrality and illustrate it visually*
- Show **agreement** and **disagreement** among the justices → • *Do something with the ties/edges - thickness and/or color*

***Along the way, don't forget about other aesthetics: color, tie overlap, etc.***

# Controlling iGraph graphic elements

- Two ways of editing network graphs
  - Use parameters of the `plot.igraph` command
  - Write information about graphical attributes to the network object itself (that `plot.igraph` will then read)
- Command order and priority
  - Parameters of the `plot.igraph` command itself will overwrite information written into the network object

# Plotting parameters

## NODES

**vertex.color** Node color

**vertex.frame.color** Node border color

**vertex.shape** One of “none”, “circle”, “square”, “csquare”, “rectangle”, “crectangle”, “vrectangle”, “pie”, “raster”, or “sphere”

**vertex.size** Size of the node (default is 15)

**vertex.size2** The second size of the node (e.g. for a rectangle)

**vertex.label** Character vector used to label the nodes

**vertex.label.family** Font family of the label (e.g. “Times”, “Helvetica”)

**vertex.label.font** Font: 1 plain, 2 bold, 3, italic, 4 bold italic, 5 symbol

**vertex.label.cex** Font size (multiplication factor, device-dependent)

**vertex.label.dist** Distance between the label and the vertex

**vertex.label.degree** The position of the label in relation to the vertex, where 0 is right, “pi” is left, “pi/2” is below, and “-pi/2” is above

# Plotting parameters

## 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”)

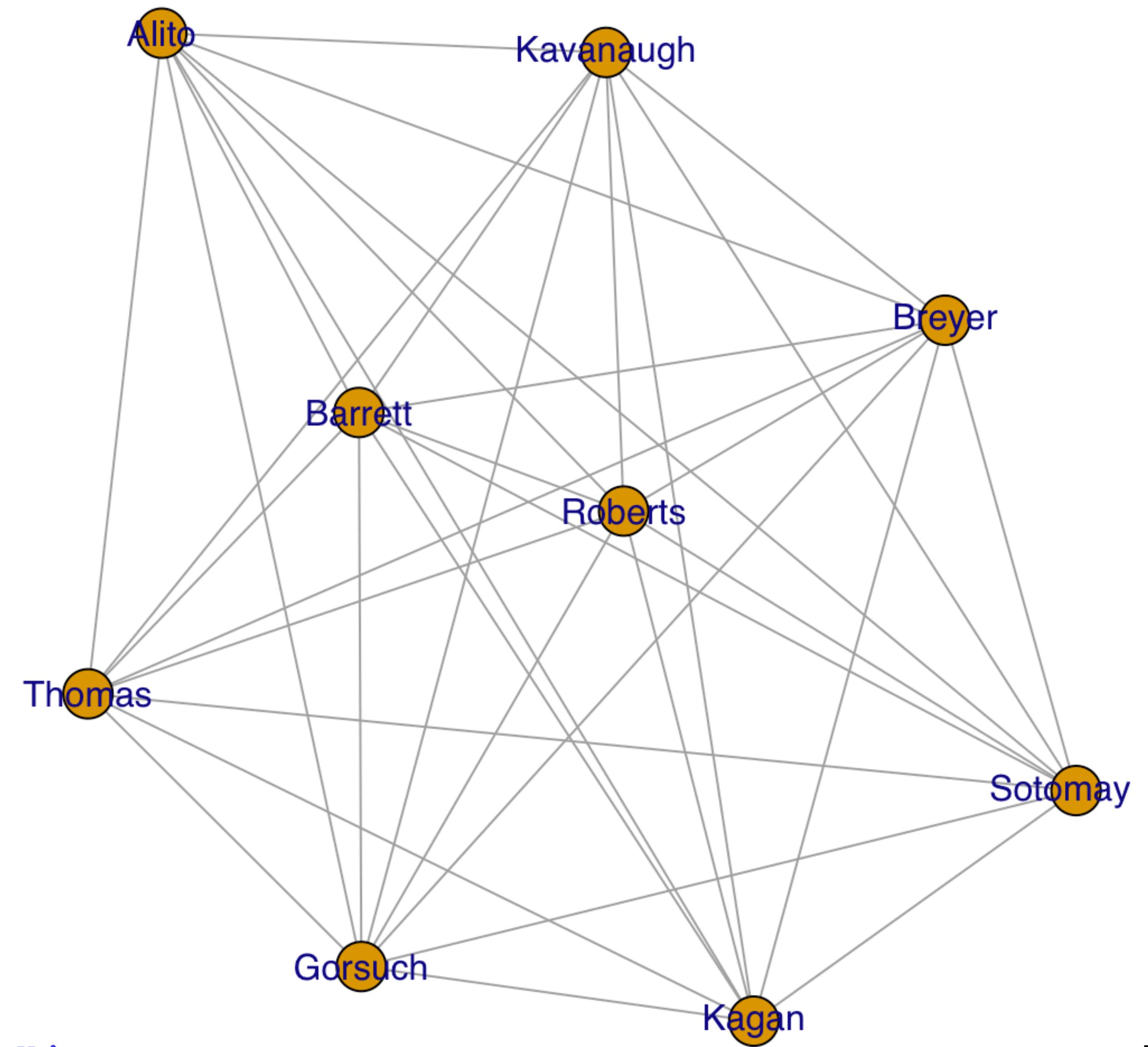
**edge.label.font** Font: 1 plain, 2 bold, 3, italic, 4 bold italic, 5 symbol

**edge.label.cex** Font size for edge labels

**edge.curved** Edge curvature, range 0-1 (FALSE sets it to 0, TRUE to 0.5)

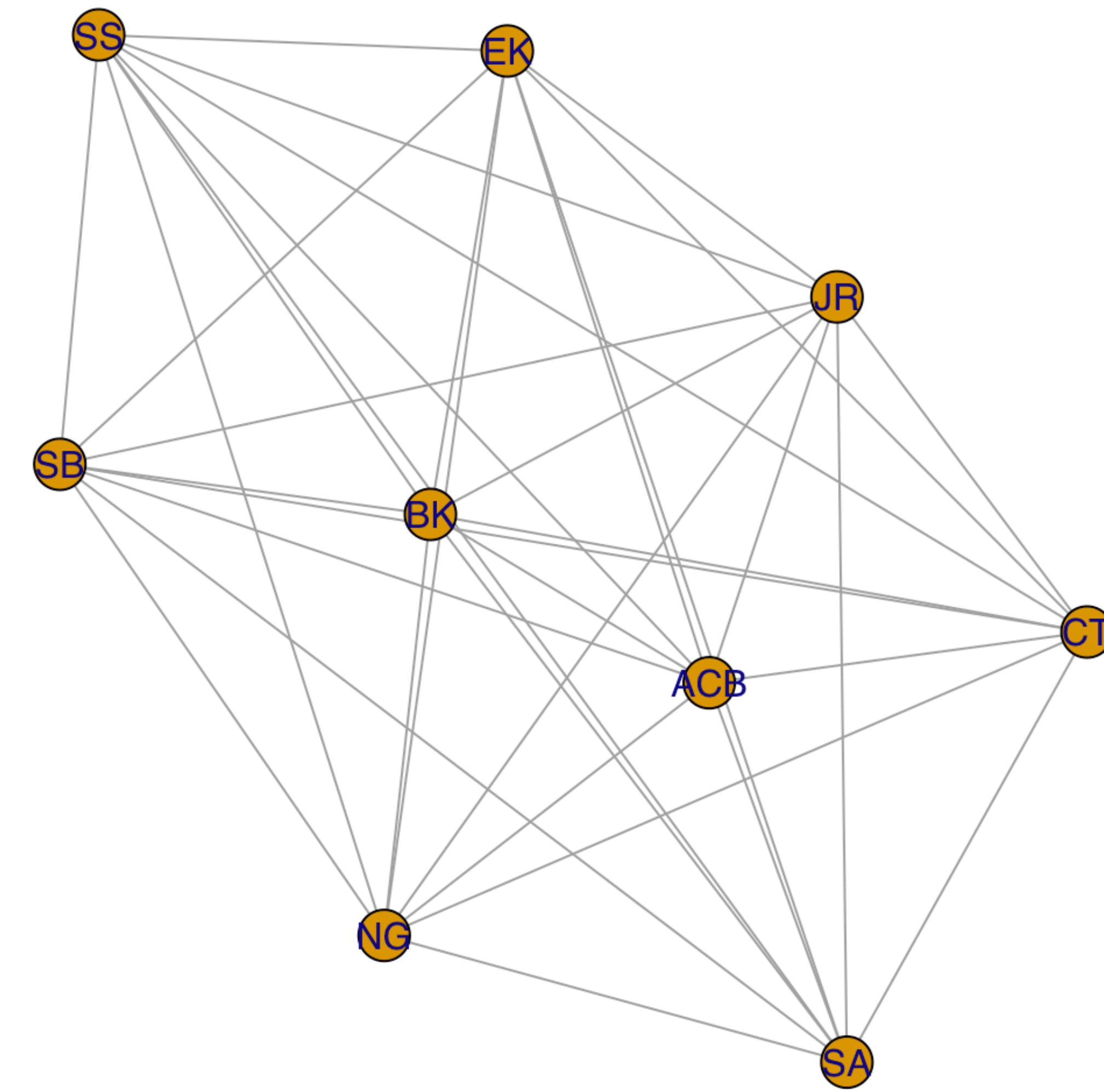
**arrow.mode** Vector specifying whether edges should have arrows,  
possible values: 0 no arrow, 1 back, 2 forward, 3 both

# Label nodes



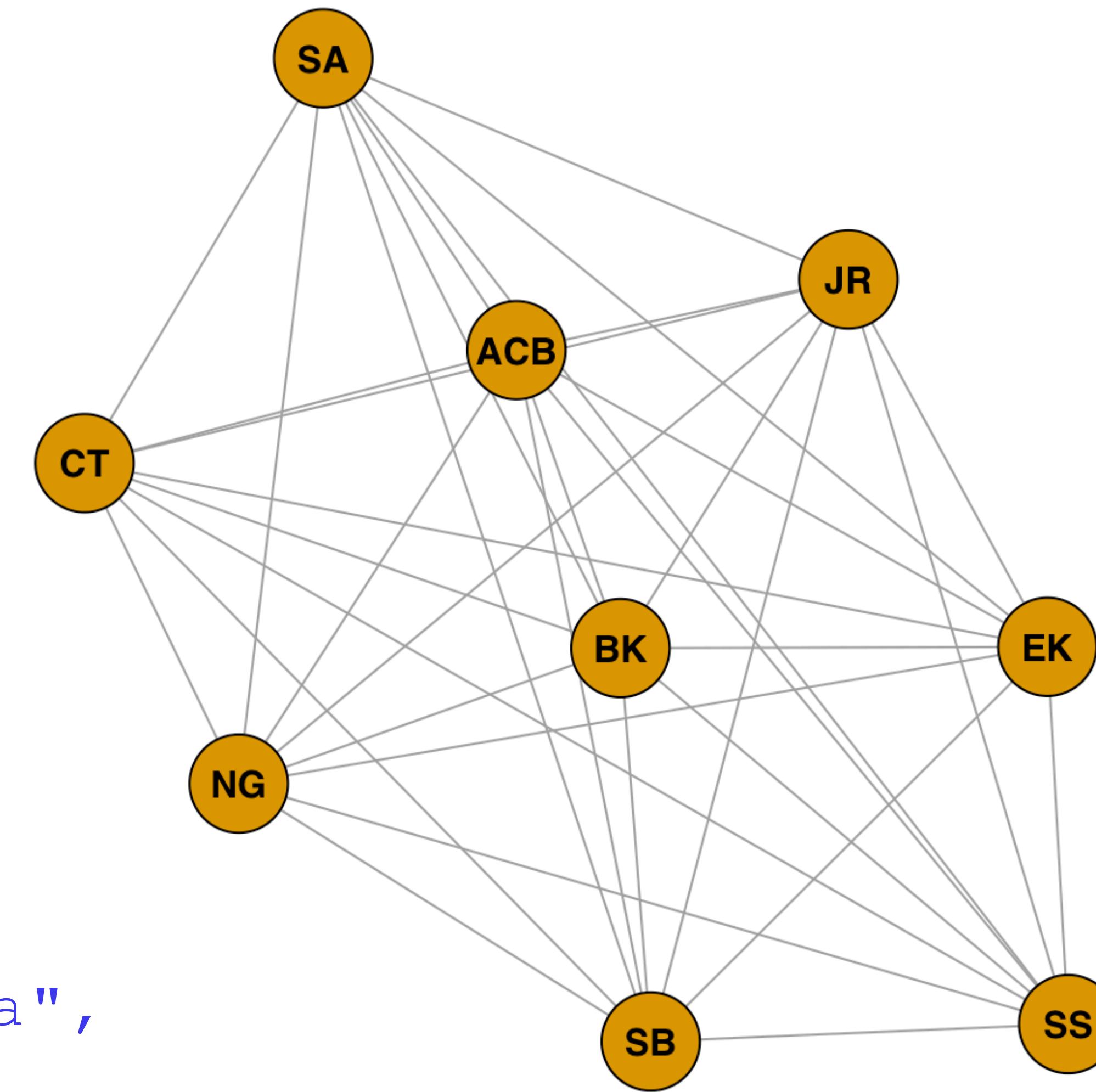
```
plot.igraph(net,  
vertex.label=v(net)$justice,  
vertex.label.family="Helvetica")
```

# Label nodes



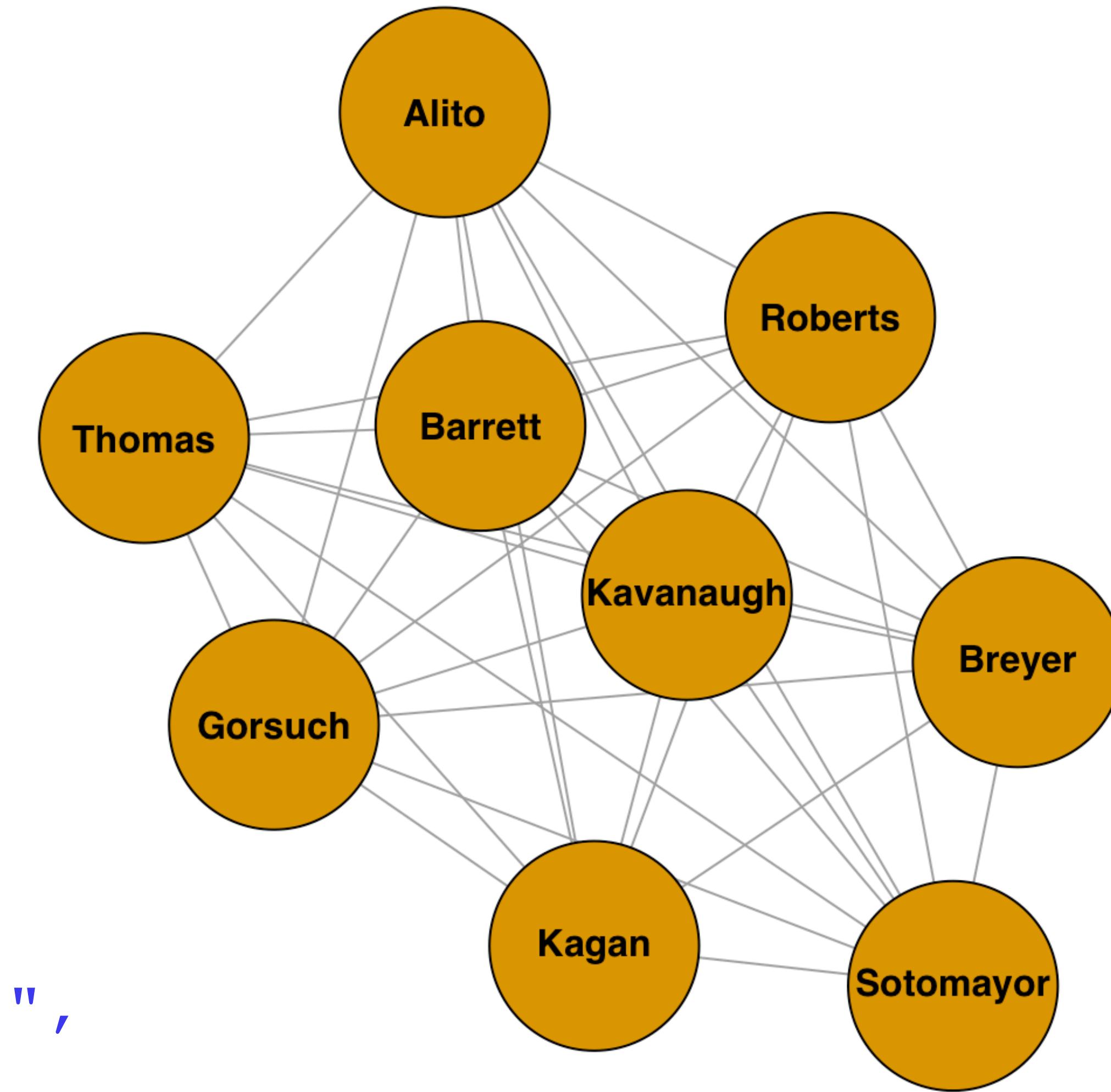
```
plot.igraph(net,  
vertex.label=V(net)$abbrev,  
vertex.label.family="Helvetica")
```

# Label nodes



```
plot.igraph(net,  
vertex.label=v(net)$abbrev,  
vertex.label.family="Helvetica",  
vertex.label.font=2,  
vertex.label.color="black",  
vertex.size=20)
```

# Label nodes



```
plot.igraph(net,  
vertex.label=v(net)$justice,  
vertex.label.family="Helvetica",  
vertex.label.font=2,  
vertex.label.color="black",  
vertex.size=48)
```

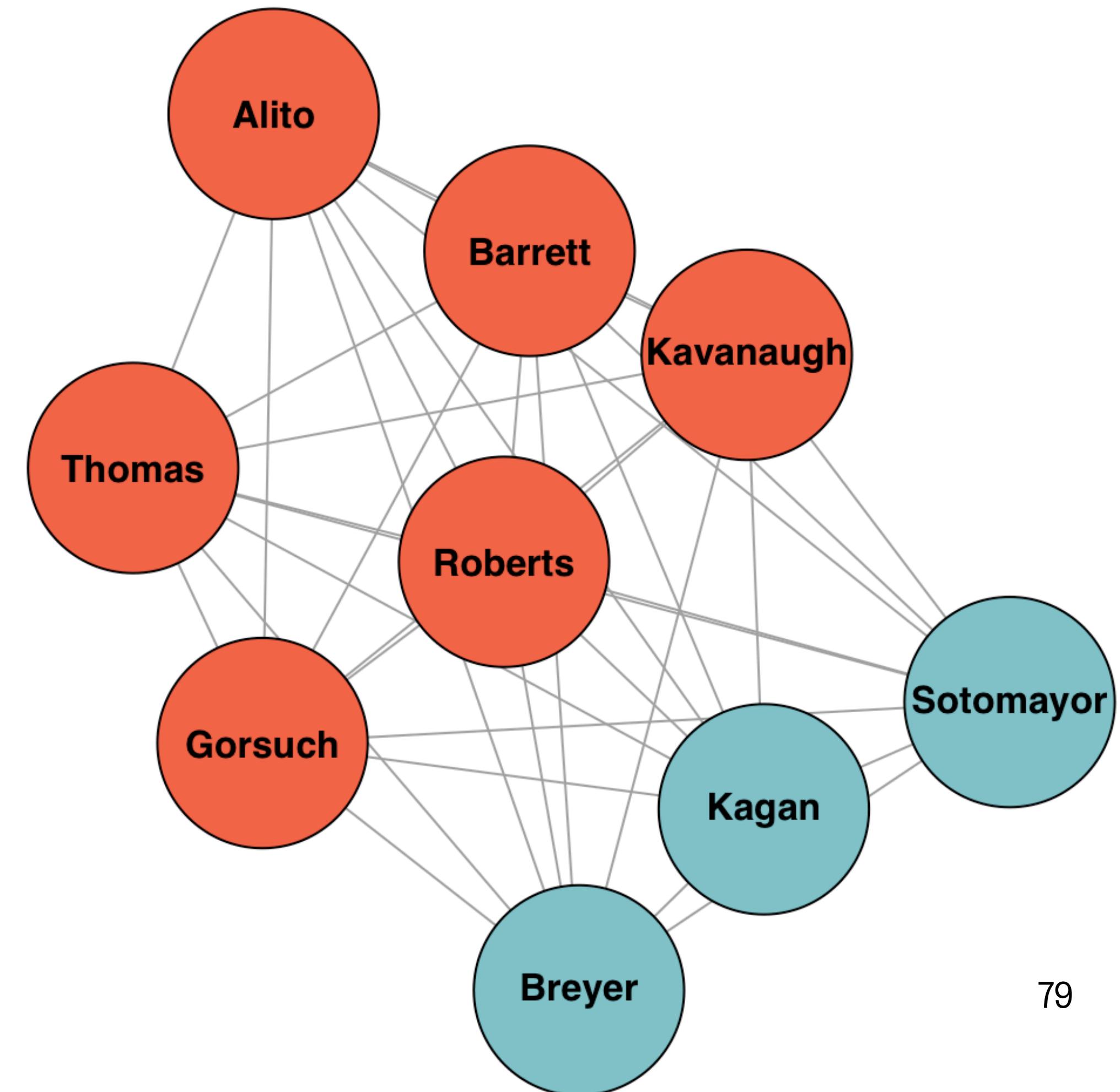
# Color by appointment party

```
#Generate colors based on party and write into network:
```

```
v(net)$color <- ifelse(v(net)$party=="Republican","coral1","cadetblue3")
```

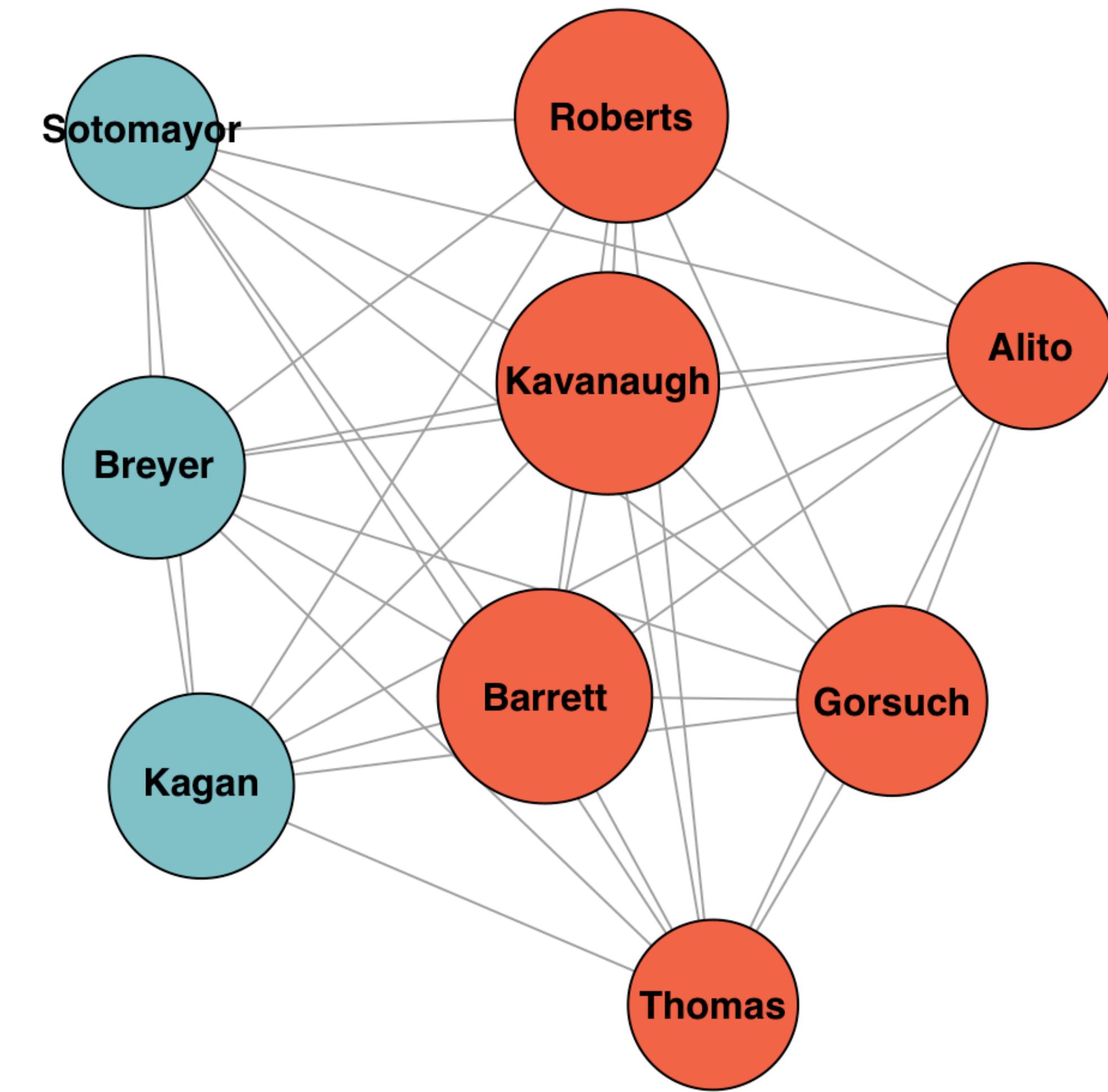
```
plot.igraph(net,  
           vertex.label=V(net)$justice,  
           vertex.label.family="Helvetica",  
           vertex.label.font=2,  
           vertex.label.color="black",  
           vertex.size=48)
```

- Note that we no longer have to “keep” color in the main line of graph code, it’s “written in” to the network object itself



# Scale nodes by centrality

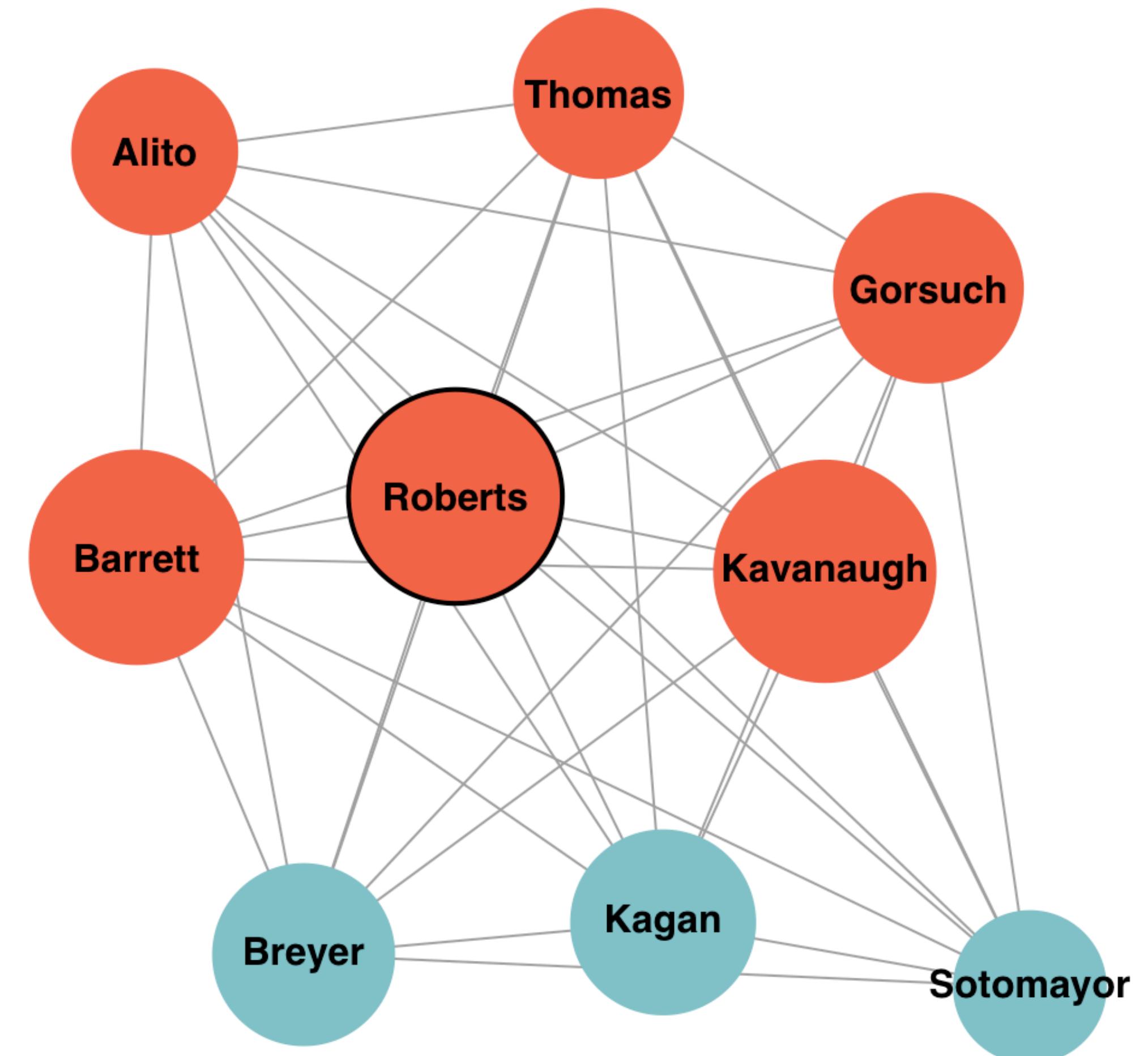
```
V(net)$eigen<-evcent(net)$vector  
  
plot.igraph(net,  
vertex.label=V(net)$justice,  
vertex.label.family="Helvetica",  
vertex.label.font=2,  
vertex.label.color="black",  
vertex.size=v(net)$eigen*50)
```



# While we're at it

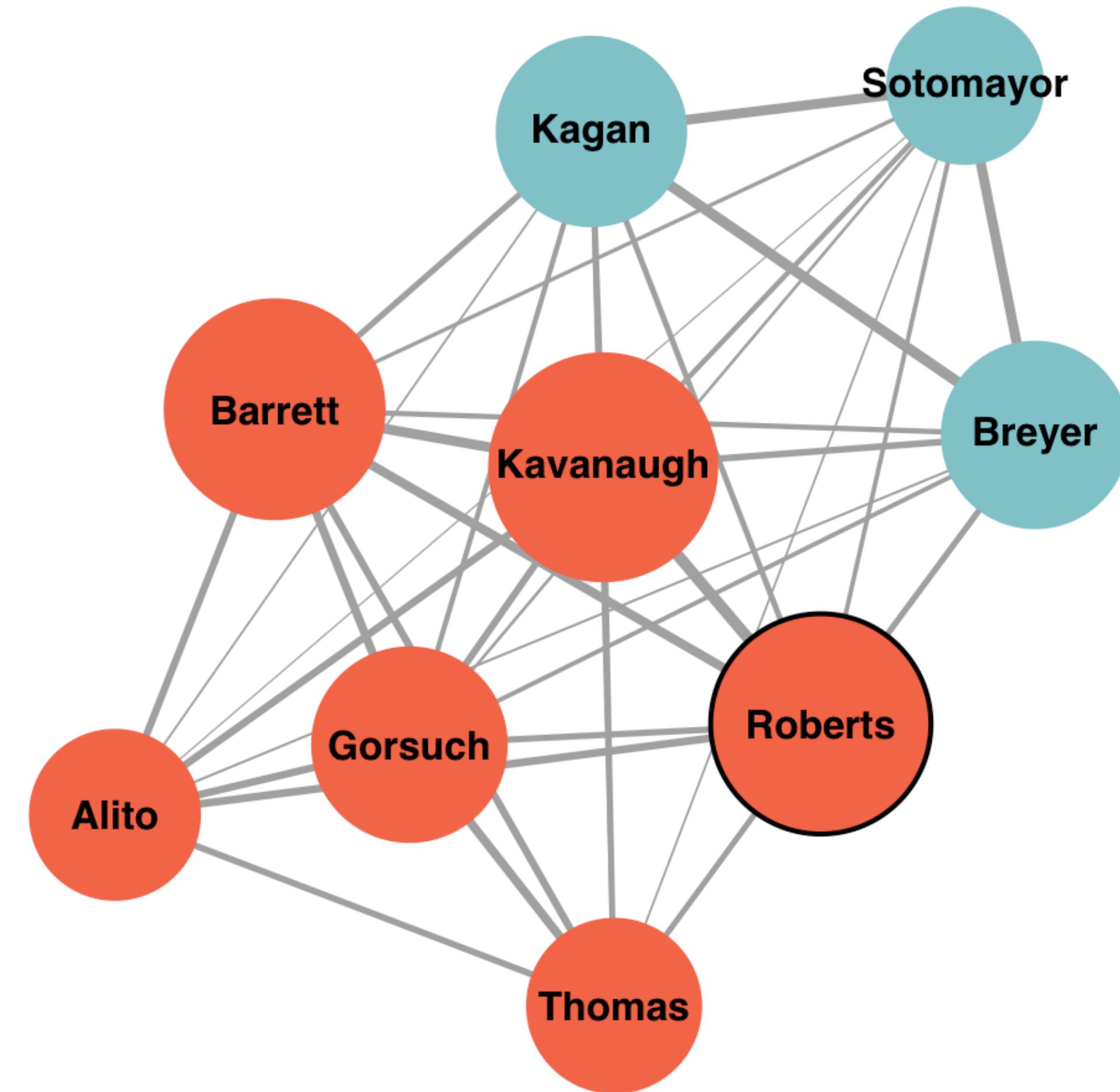
```
v(net)$width<-
  ifelse(V(net)$type=="Chief",2,1)

plot.igraph(net,
  vertex.label=V(net)$justice,
  vertex.label.family="Helvetica",
  vertex.label.font=2,
  vertex.label.color="black",
  vertex.size=V(net)$eigen*50,
  vertex.frame.width=V(net)$width)
```



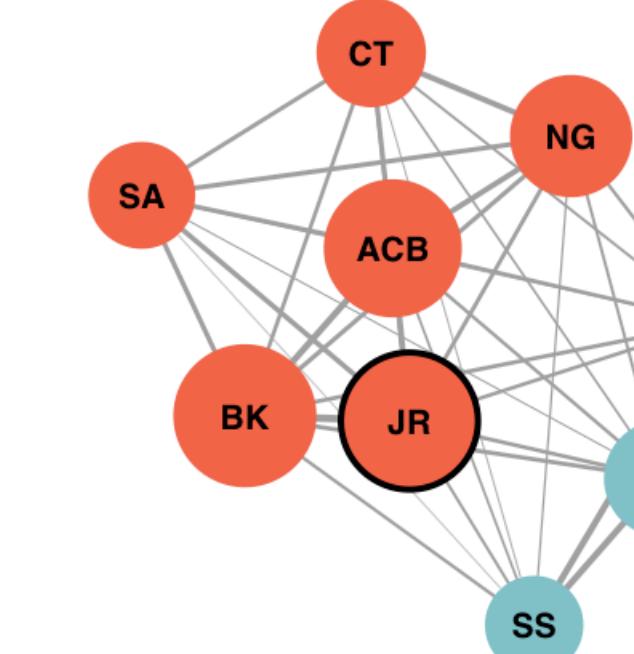
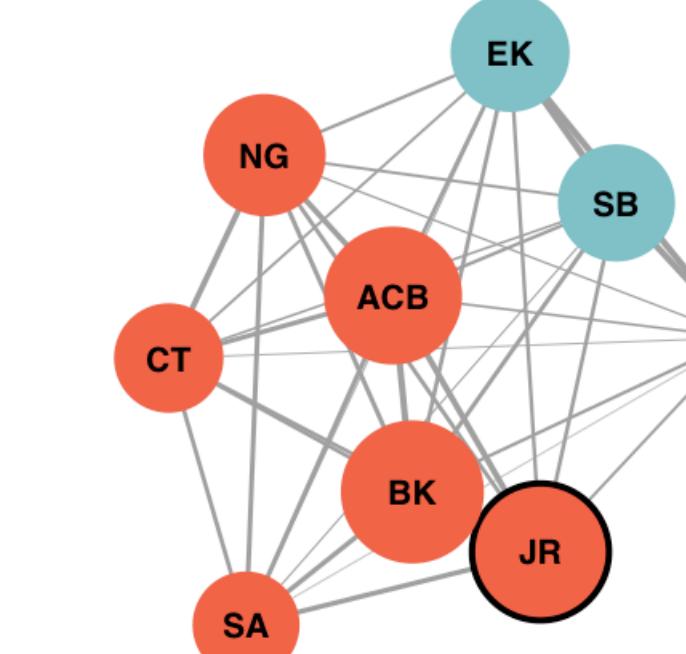
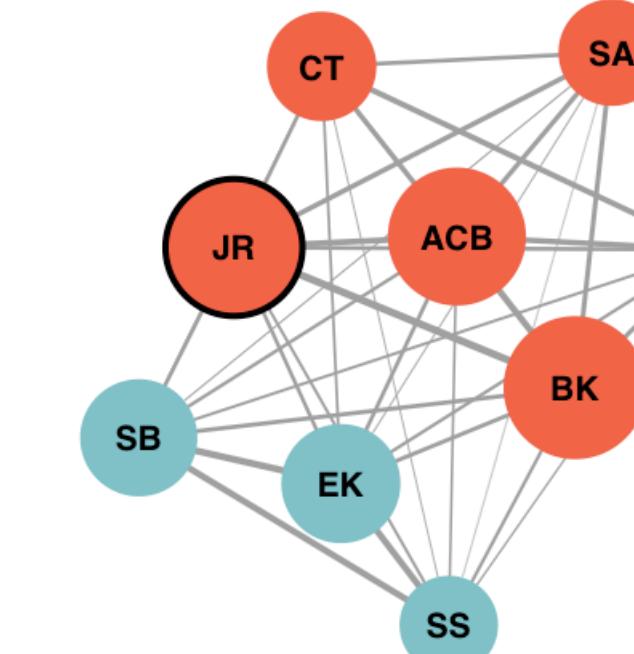
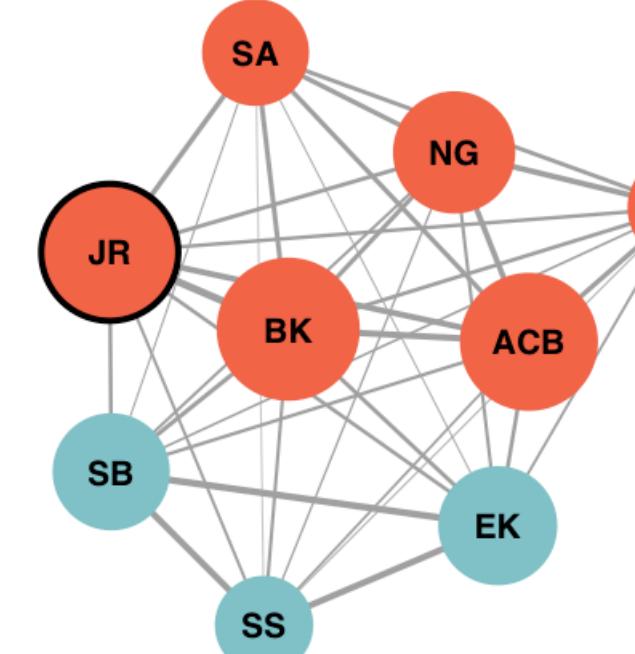
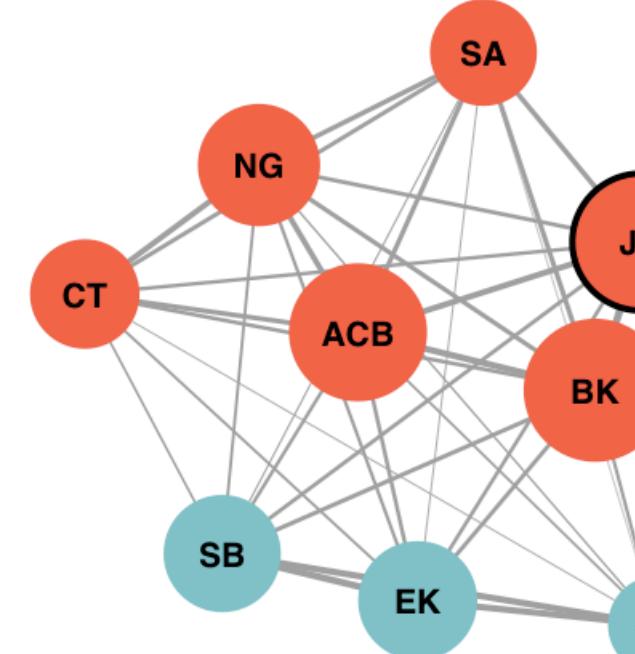
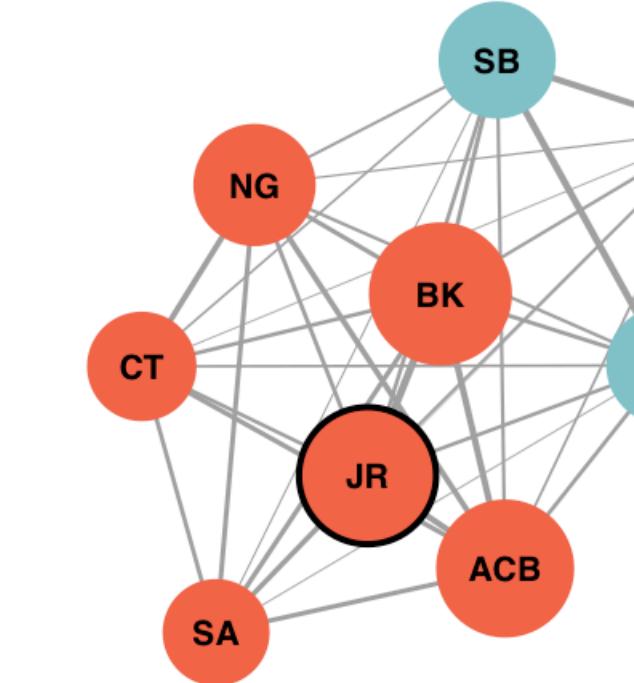
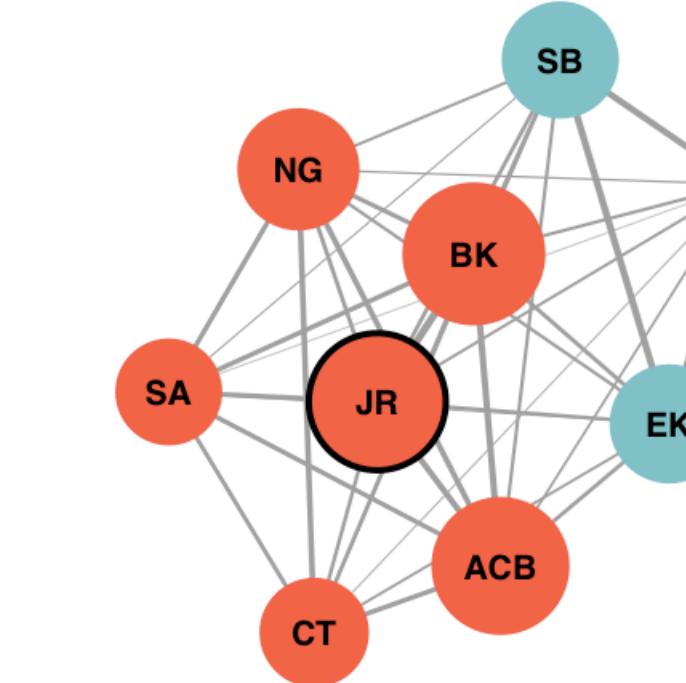
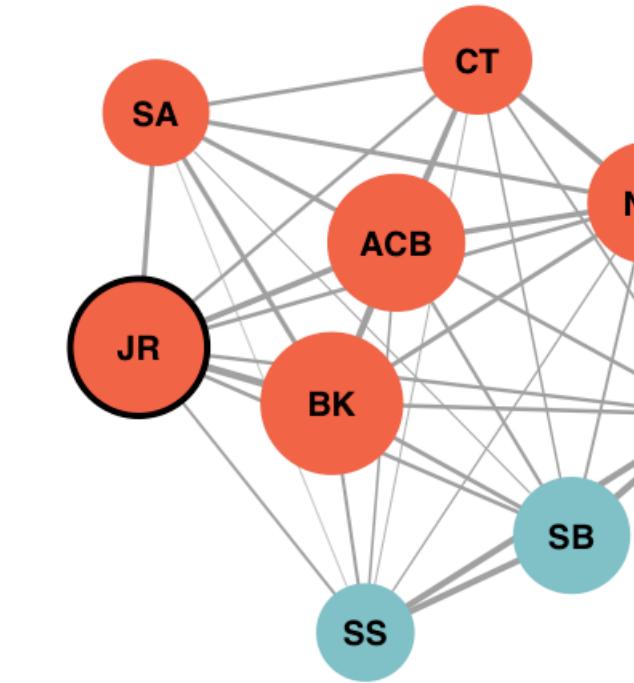
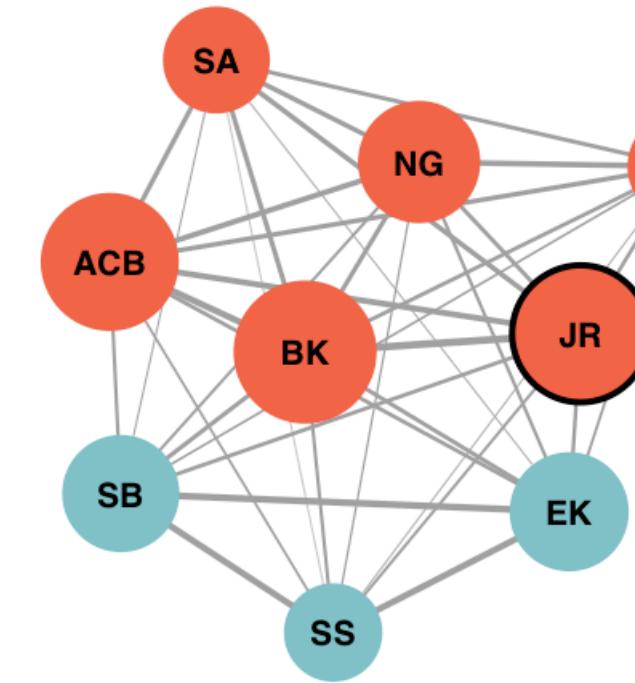
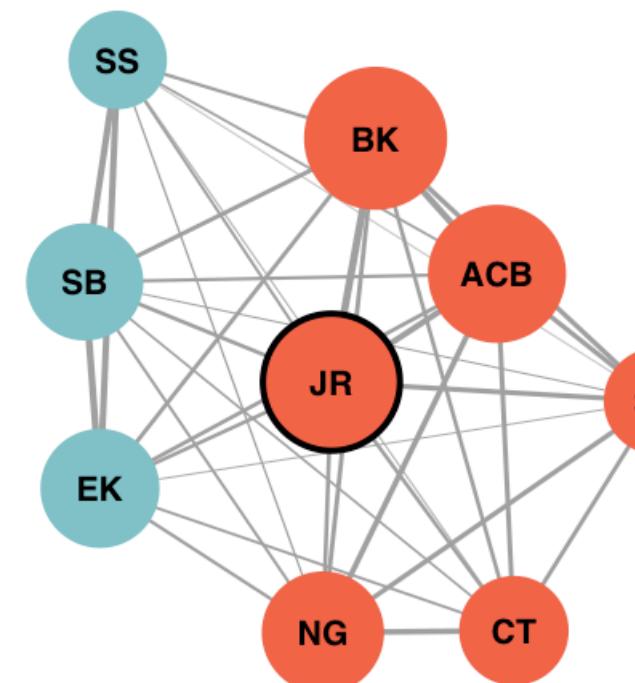
# Finally: tie strength

```
plot.igraph(net,  
            edge.width=E(net)$weight*5,  
            vertex.label=V(net)$justice,  
            vertex.label.family="Helvetica",  
            vertex.label.font=2,  
            vertex.label.color="black",  
            vertex.size=V(net)$eigen*50,  
            vertex.frame.width=V(net)$width)
```



# Graph layouts

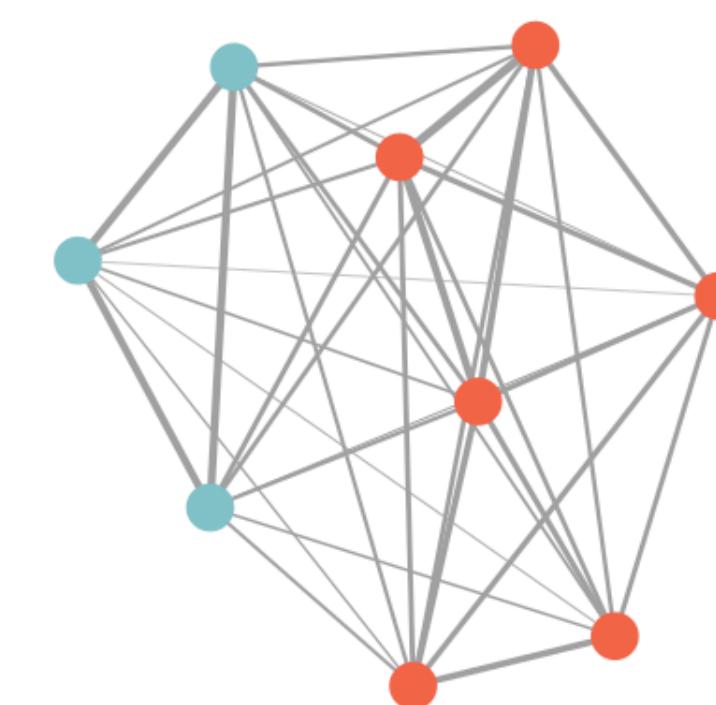
- iGraph generates layouts quasi-randomly within similar constraints; each time the network is drawn, the layout will look slightly different



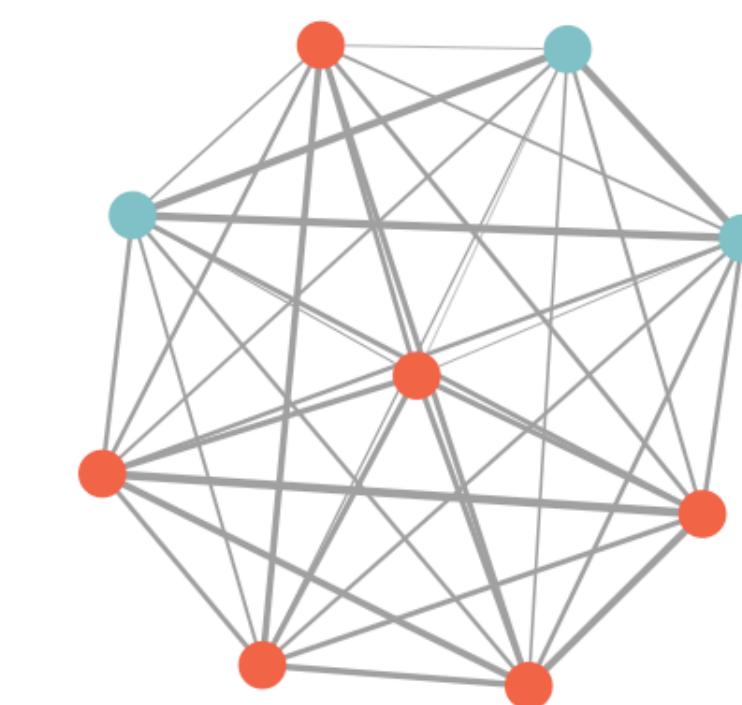
# An assortment of network layouts

- We control with the `layout` option within `plot.igraph`

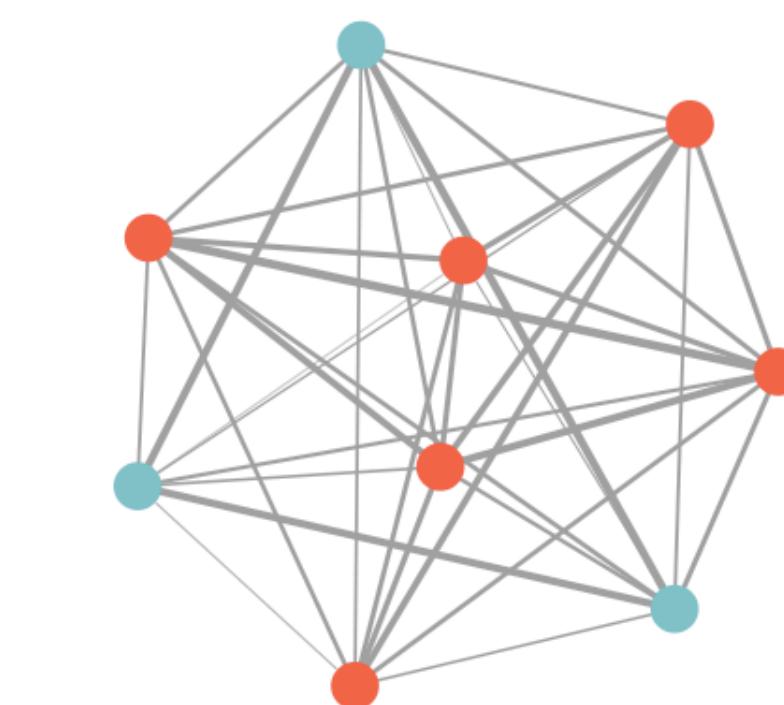
`layout_with_fr`



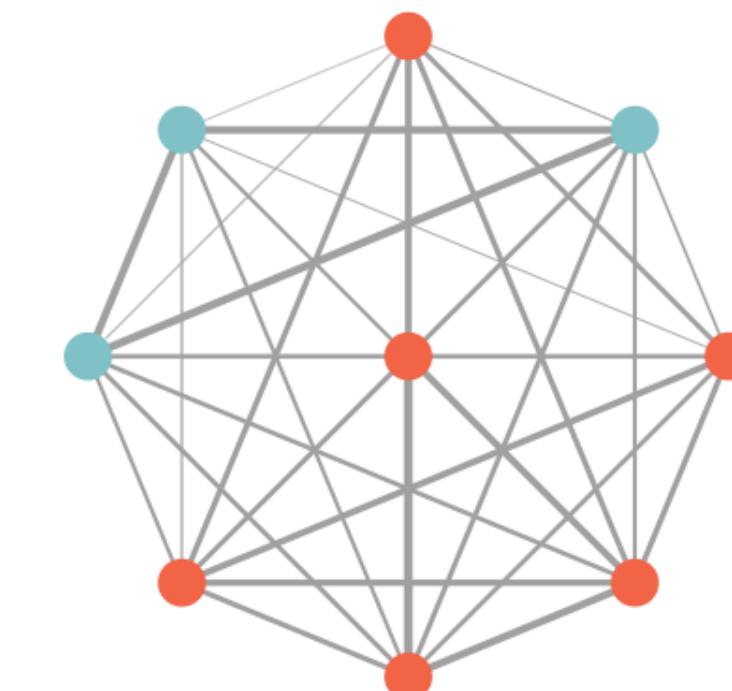
`layout_with_gem`



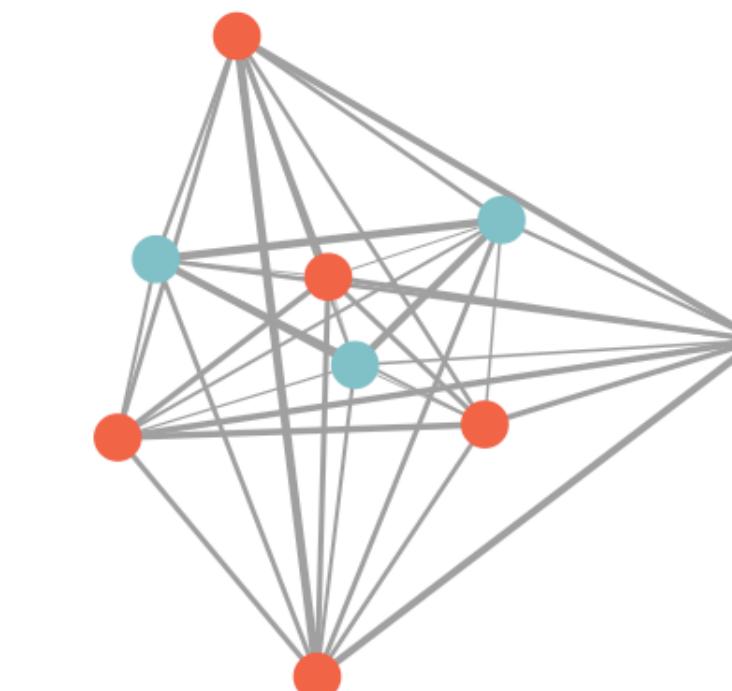
`layout_with_graphopt`



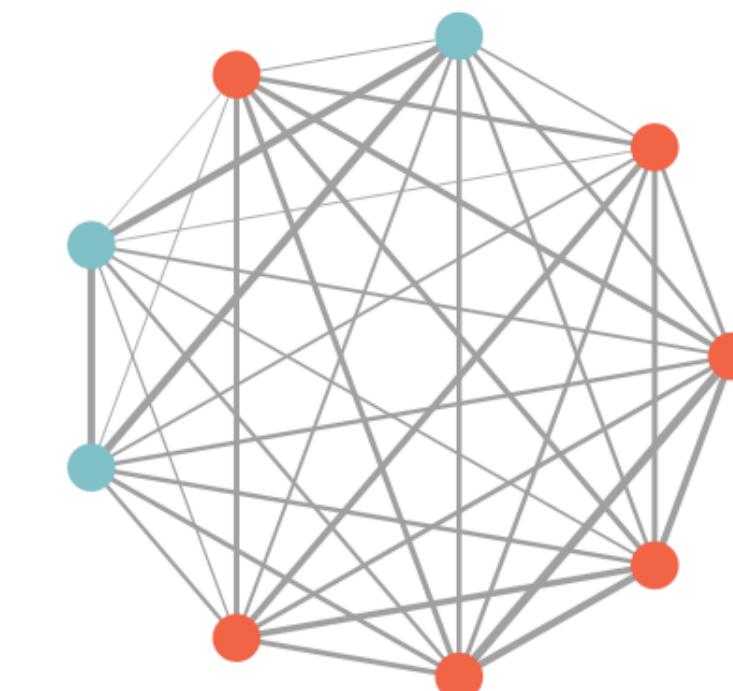
`layout_as_star`



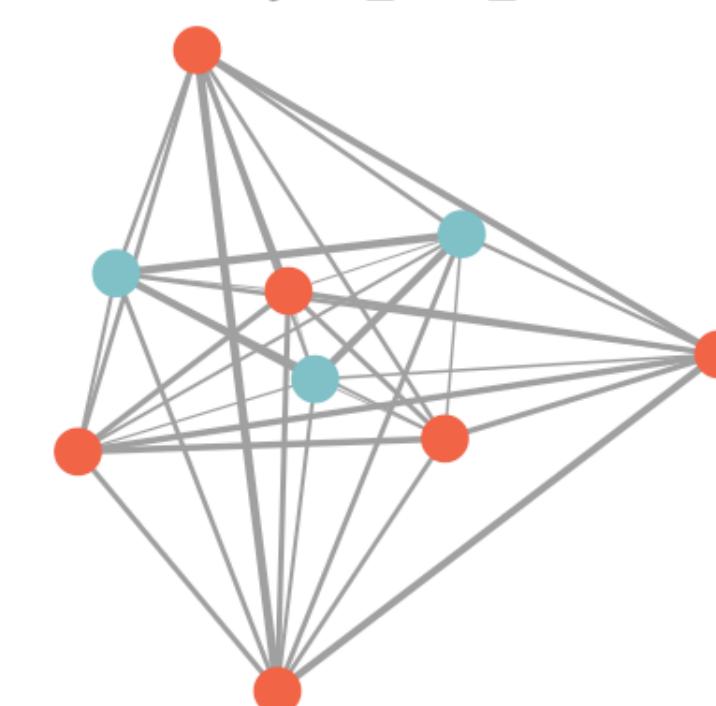
`layout_components`



`layout_in_circle`



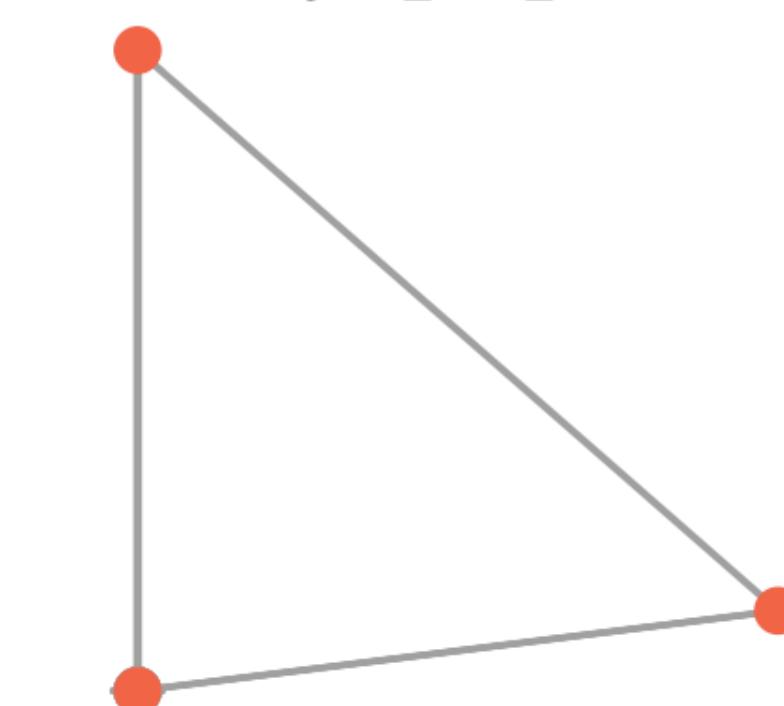
`layout_with_kk`



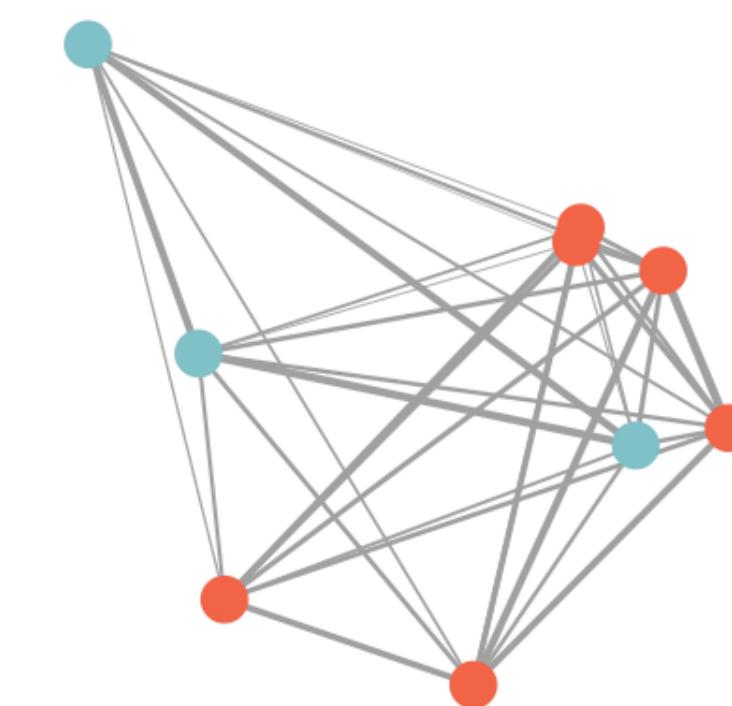
`layout_with_lgl`



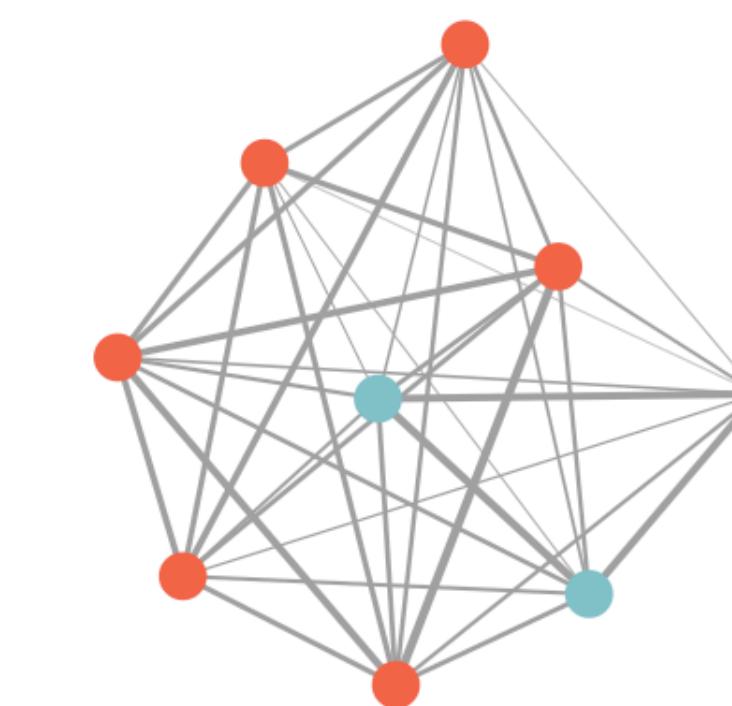
`layout_with_mds`



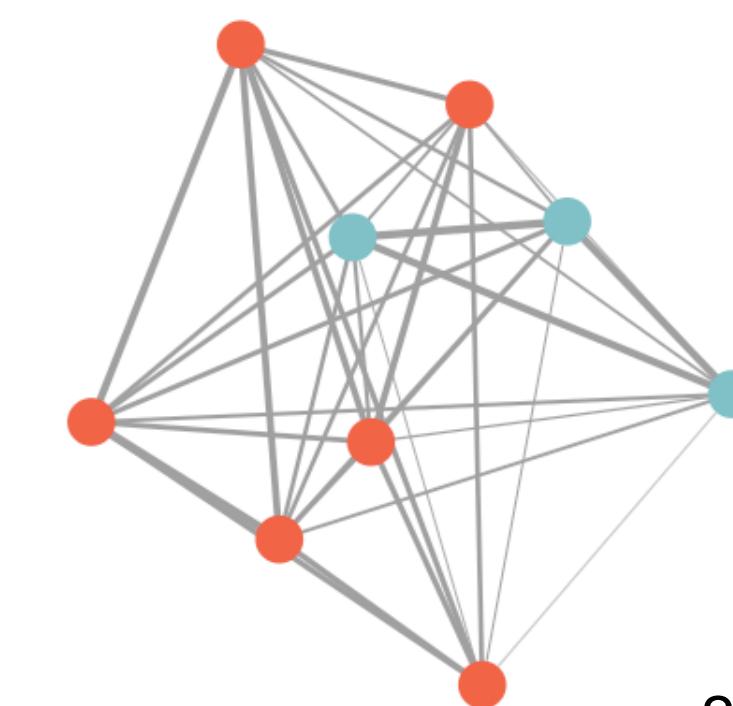
`layout_randomly`



`layout_with_dh`

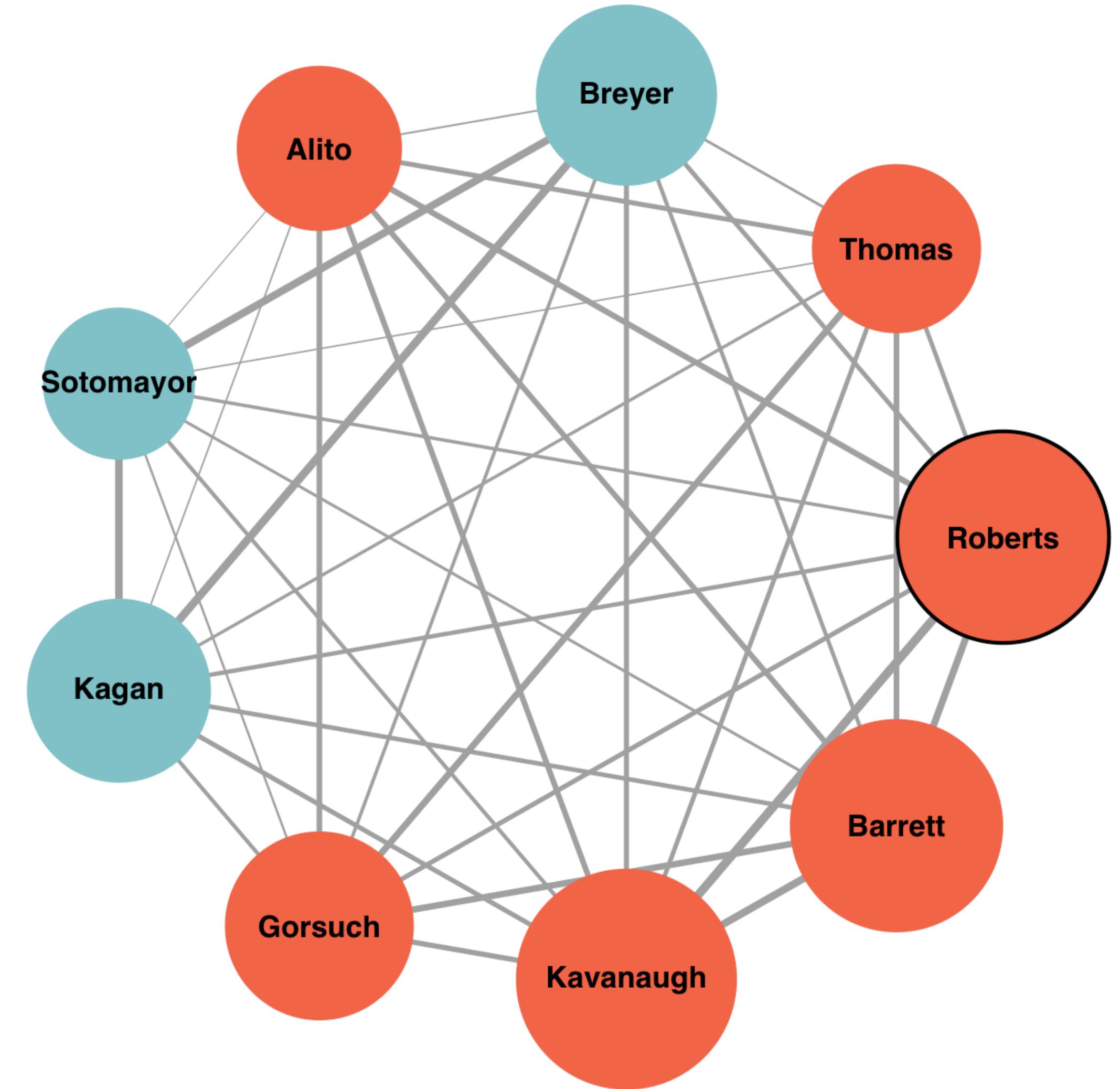


`layout_with_drl`



# The circle layout

```
plot.igraph(net,
edge.width=E(net)$weight*5,
vertex.label=V(net)$justice,
vertex.label.family="Helvetica",
vertex.label.font=2,
vertex.label.color="black",
vertex.size=V(net)$eigen*50,
vertex.frame.width=V(net)$width,
layout=layout_in_circle)
```

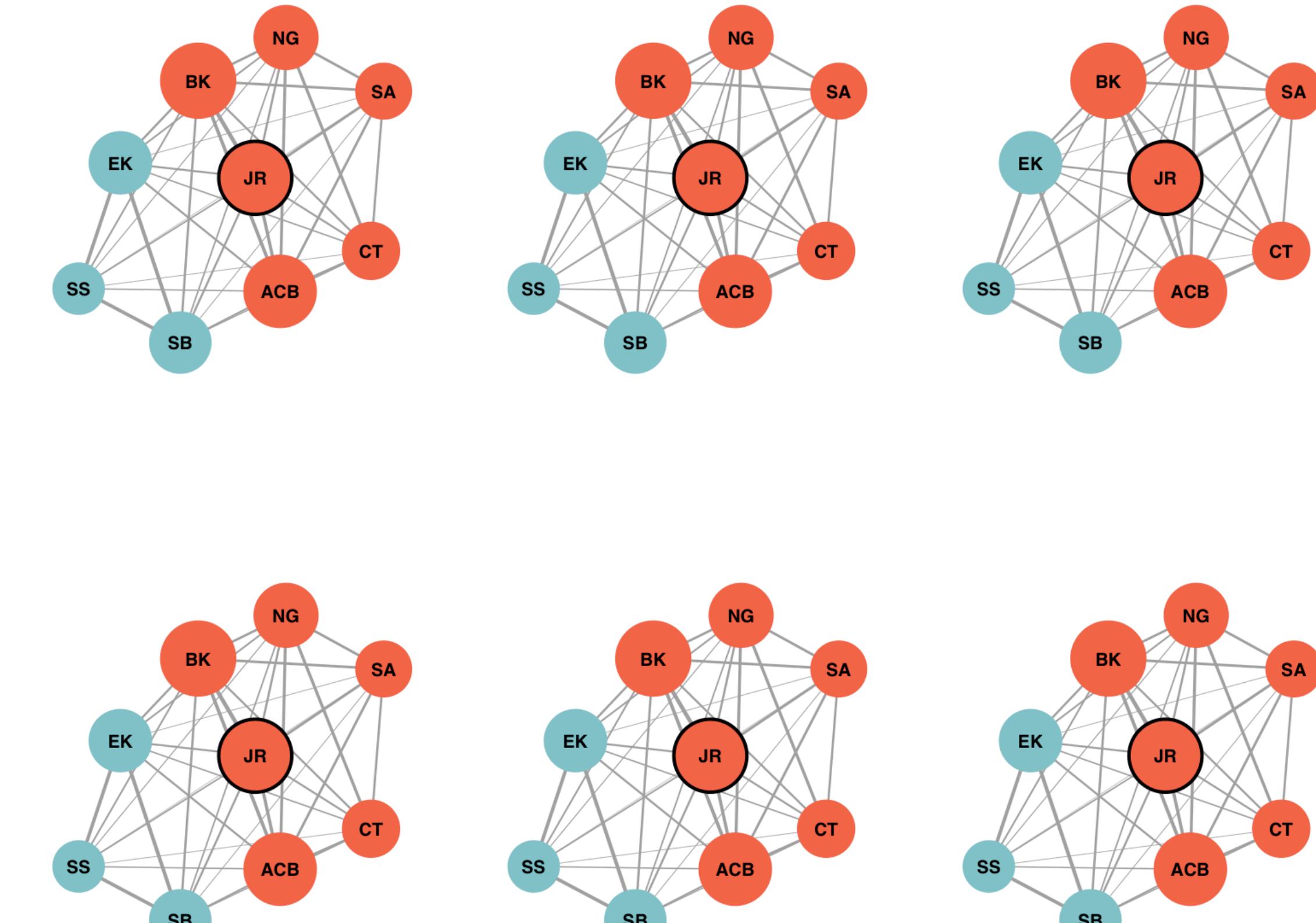


# Locking in layouts

- We can create and store a particular layout to use if we wish (in this case, a layout linked to tie strength), so that the graph appears the same every time

```
nicelayout <- layout_with_fr(net,  
  weights=E(net)$weight)
```

```
plot.igraph(net,  
  edge.width=E(net)$weight*5,  
  vertex.label=V(net)$abbrev,  
  vertex.label.family="Helvetica",  
  vertex.label.font=2,  
  vertex.label.color="black",  
  vertex.size=V(net)$eigen*50,  
  vertex.frame.width=V(net)$width,  
  layout=nicelayout)
```



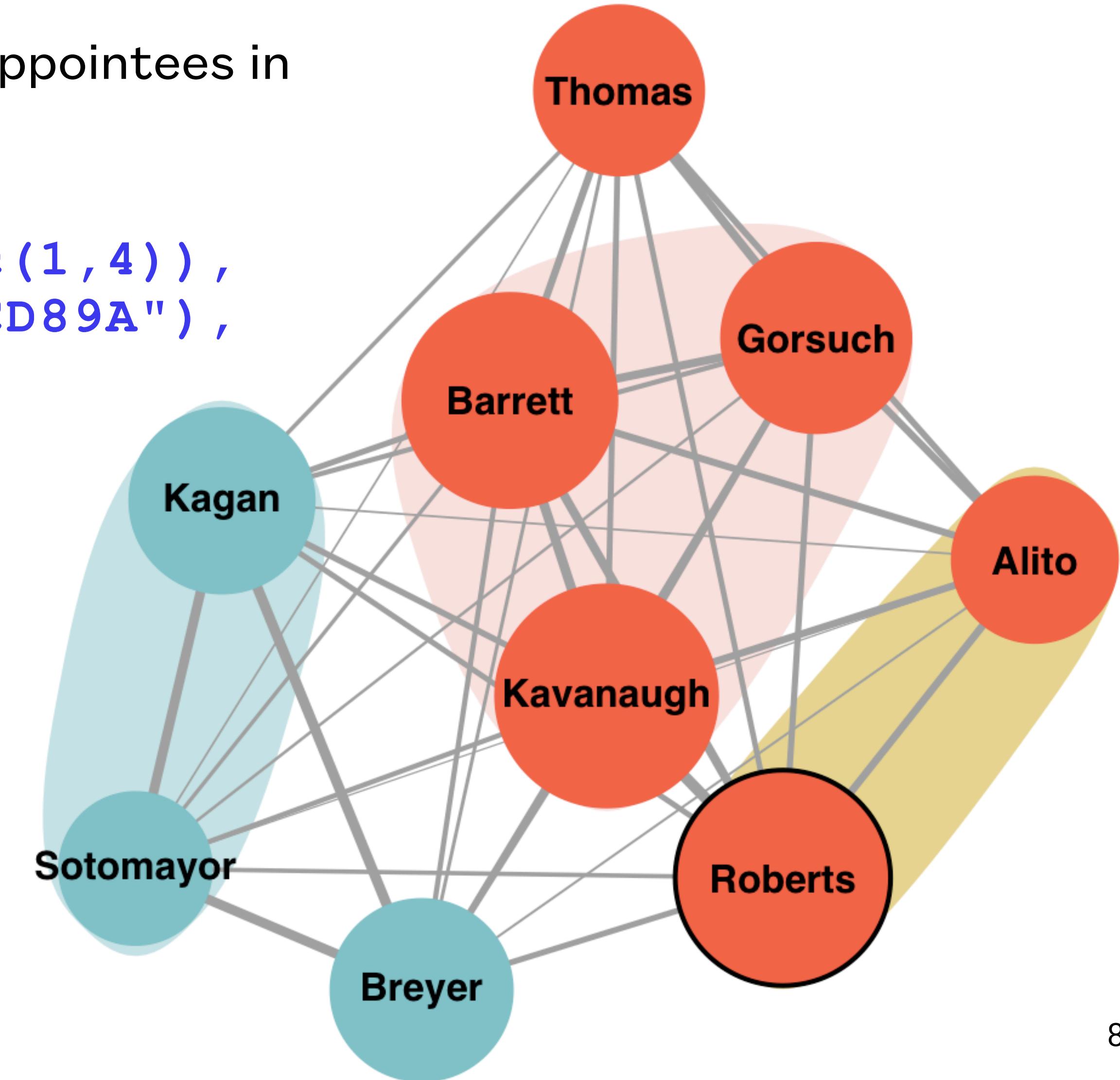
- And can store it to drive for next time

```
write.csv(nicelayout, "layout.csv")  
nicelayout <- as.matrix(read.csv("layout.csv"))
```

# Highlighting nodes

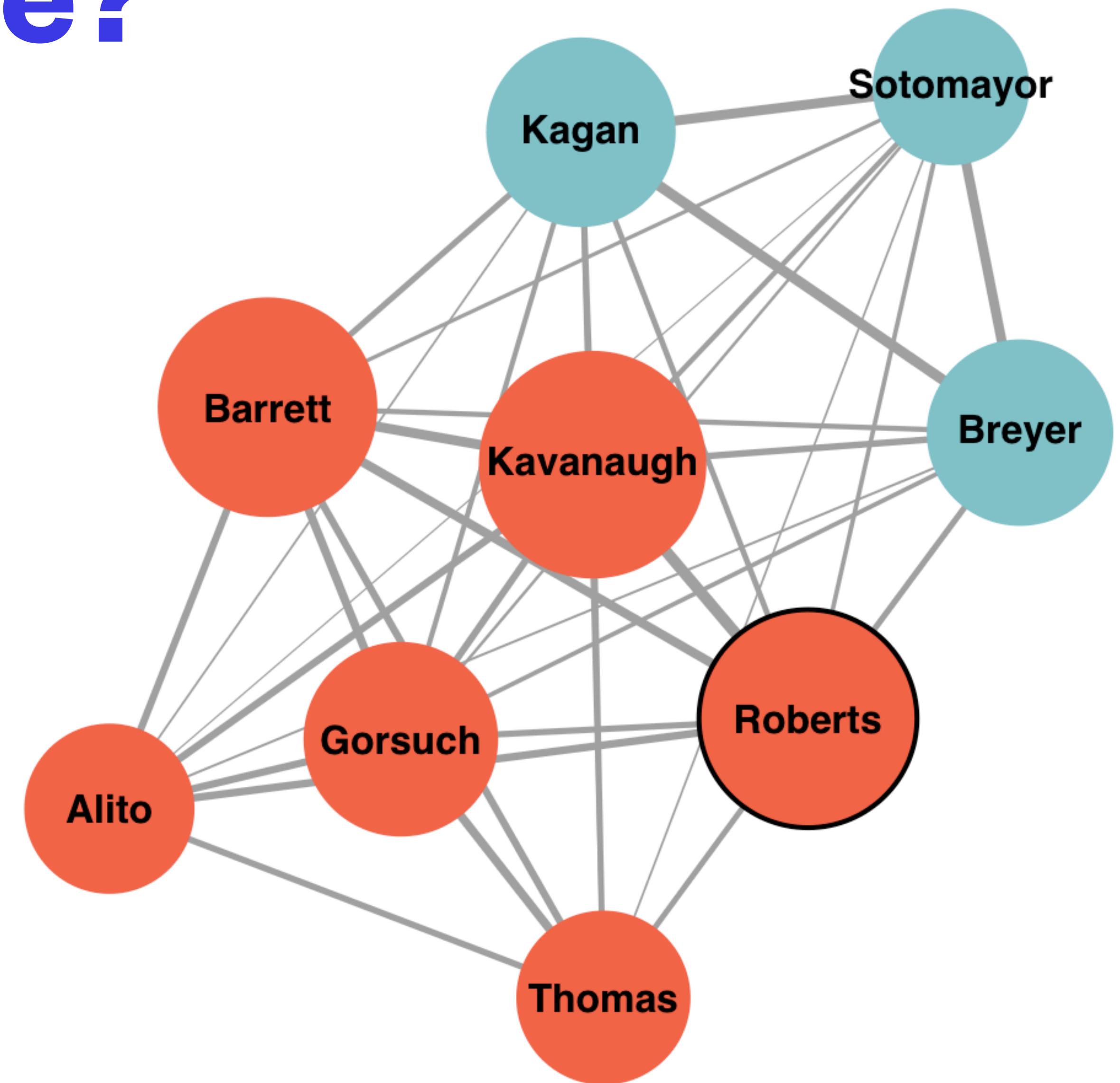
- Here, we highlight Trump, Obama, and Bush Jr. appointees in red, blue, and yellow, respectively.

```
plot.igraph(net,
mark.groups=list(c(7,8,9), c(5,6), c(1,4)),
mark.col=c("#FFE4E1", "#C5E5E7", "#ECD89A"),
mark.border=NA,
edge.width=E(net)$weight*5,
vertex.label=V(net)$justice,
vertex.label.family="Helvetica",
vertex.label.font=2,
vertex.label.color="black",
vertex.size=V(net)$eigen*50,
vertex.frame.width=V(net)$width)
```



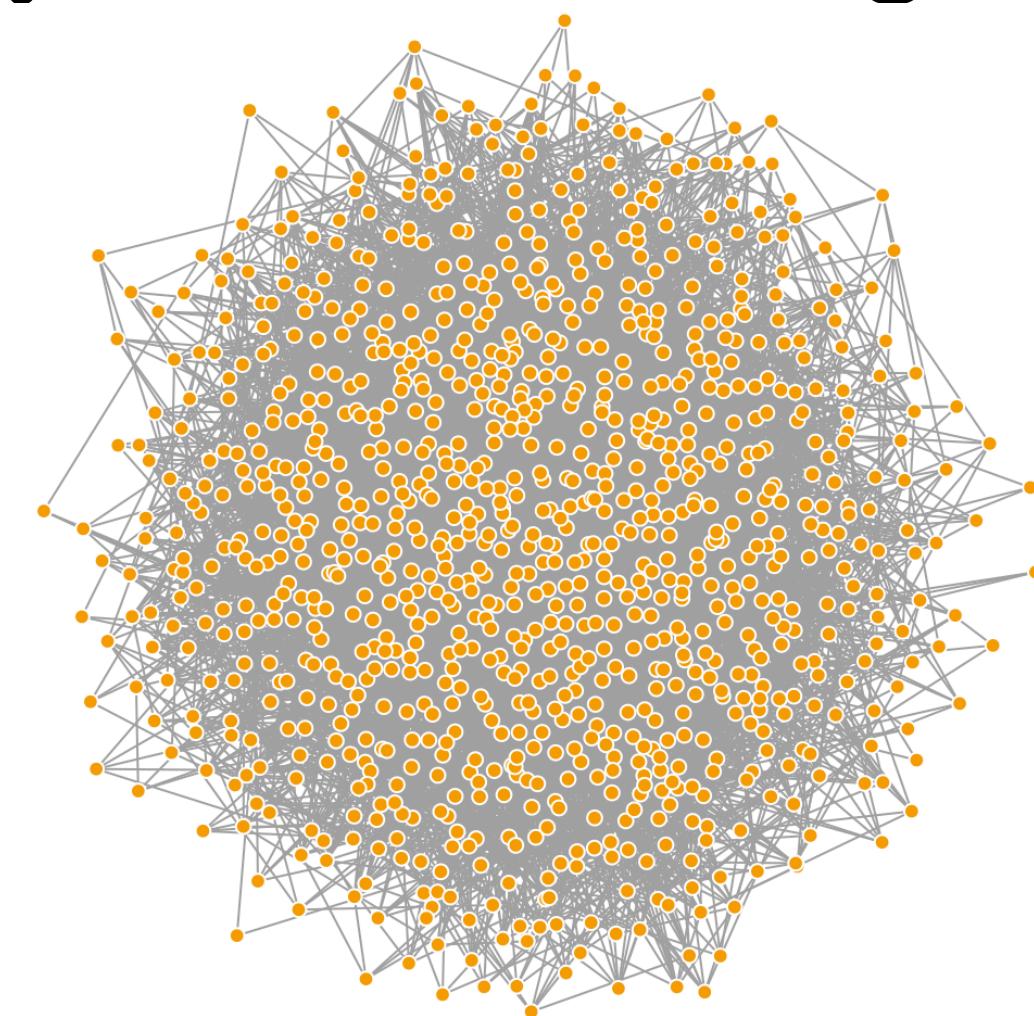
# So what can we see?

- *Strong graphics present ready information quickly, at a glance*
  - *While rewarding further inspection with more detail*
- Centrality/importance (node size, graph position from layout tied to edge weights)
- Party (color)
- Tie strength (line width)
- Identities of justices



# The example is just an example

- The Supreme Court is a nice (small, simple) network to begin with
- But the techniques here can be used for larger networks just as easily (albeit with things possibly slowing down as the scale and size of your network expands)
  - Larger networks bring different considerations:
    - Smaller (or no!) nodes between links
    - Focus on structure over individual elements
    - Easier to draw “hairballs”
  - Also, don’t let yourself be caught by thinking you *must* draw a network with network data

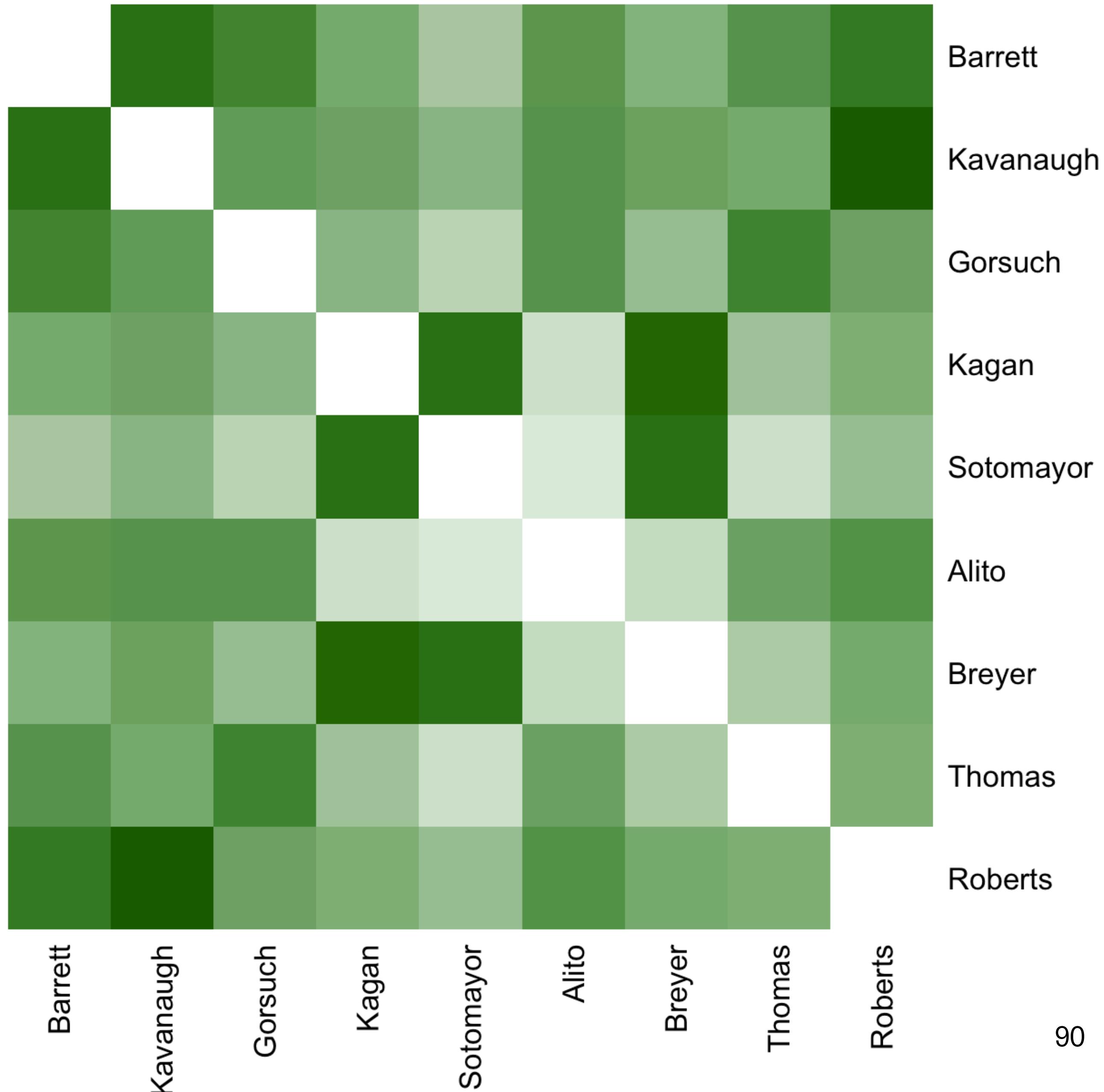


# More than one

## WAY TO VISUALIZE A NETWORK

- This is a heatmap of network agreement (note that it is symmetrical/undirected)

```
palf <-  
  colorRampPalette(  
    c("white", "darkgreen"))  
  
netm <- get.adjacency(net,  
  attr="weight", sparse=F)  
  
colnames(netm) <- V(net)$justice  
rownames(netm) <- V(net)$justice  
  
heatmap(netm, Rowv = NA,  
  Colv = NA, col = palf(100),  
  scale="none", margins=c(10,10) )
```



## Part Four

---

### **Wrapping up: where do I go from here?**

---

# Alternatives to R

- Programmatic
  - Python's **NetworkX** library (and iGraph again)
  - Julia's **JuliaGraphs** library
- GUIs
  - UCI-NET
  - SocNetV
  - Gephi
  - Cytoscape
  - Pajek
  - GraphVis

# Resources

- Materials from this lecture (slides, data, code): <https://jacklreilly.github.io/networkscrashcourse>
- Core references in these slides
  - Ognyanova, K. (2021) Network visualization with R. Retrieved from [www.kateto.net/network-visualization](http://www.kateto.net/network-visualization).
  - Healy, K. (2018) Data Visualization: [A Practical Introduction](#). Princeton. (Also available: [socviz.co](http://socviz.co).)
- Other books
  - The classic: Wasserman and Faust, 1994. *Social Network Analysis*.
  - The modern classic: Jackson, 2010. *Social and Economic Networks*.
  - For undergraduate courses: Scott, 2017. *Social Network Analysis*.
  - More formal: Menczer, Fortunato, and Davis, 2020. *A First Course in Network Analysis*.

# Teaching in context

## WHERE DOES THIS FIT?

- Not an exhaustive overview of networks, of course
- Placed in a course, this content would be different in a few ways
  - Interspersed with hands-on R workshops
  - In undergraduate classes, include active learning (with yarn!)
- Where would this fit in a class?
  - This is a more concentrated dose of presentations I do in my current advanced undergraduate course on social networks
  - Normally, it would be divided more (not covering data and visualization together, for one)

# Teaching in context

- What would come around this?
  - Depends - social networks exists at the cross-section of methodology and substance
    - In a more substantive course, reading substantive applications goes hand-in-hand with learning core elements of network data, description, and visualization
    - In a more methodologically focused course, this would precede a deeper discussion of proper network analysis: network formation, blockmodels and community detection, contagion and influence, and eventually, models for inference (ERGMs)
  - It might be done with intro to R content at the beginning, as well

# Thank you!

---

**Materials:** <https://jacklreilly.github.io/networkscrashcourse>

---

**Contact:** [jreilly@ncf.edu](mailto:jreilly@ncf.edu)

---

# Questions?