SNA Basics IX
Brokers and Bridges

@ Sean F. Everton

Place a header at the top of your script that tells you what you called it, what it accomplishes, etc.

```
##################################################
# What: Brokers and Bridges in R
# File: snab9.R
# Created: 02.28.29
# Revised: 07.09.18
##################################################
```

Data

For this exercise, we will use the communication network of a wood-processing facility where workers rejected a new compensation package and eventually went on strike. Management then brought in an outside consultant to analyze the employee's communication structure because it felt that information about the package was not being effectively communicated to all employees by the union negotiators. The outside consultant asked all employees to indicate, on a 5-point scale, the frequency that they discussed the strike with each of their colleagues, ranging from 'almost never' (less than once per week) to 'very often' (several times per day). The consultant used 3 as a cut-off value in order to identify a tie between two employees. If at least one of two persons indicated they discussed work with a frequency of three or more, a tie between them was included in the network. The data accompany the book, "Exploratory Social Network Analysis with Pajek," also published by Cambridge.

Setup

Clear the workspace each time before beginning.

```
rm(list=ls())
```

Set your working directory to where the data are, so you don't have to include the entire path when importing and exporting data, files, etc.

```
setwd("~/Dropbox/Networks and Religion (Book)/Website/Labs/SNA Basics 9")
```

Brokers and Bridges in *statnet*

Load the statnet libraries along with the intergraph library; we also need the "car" library, which includes a helpful and simple recode function

```
library(sna)
```

```
## Loading required package: statnet.common

##
## Attaching package: 'statnet.common'

## The following object is masked from 'package:base':
##
##     order
## Loading required package: network
```

```
## network: Classes for Relational Data
## Version 1.13.0.1 created on 2015-08-31.
## copyright (c) 2005, Carter T. Butts, University of California-Irvine
##                     Mark S. Handcock, University of California -- Los Angeles
##                     David R. Hunter, Penn State University
##                     Martina Morris, University of Washington
##                     Skye Bender-deMoll, University of Washington
##  For citation information, type citation("network").
##  Type help("network-package") to get started.
```

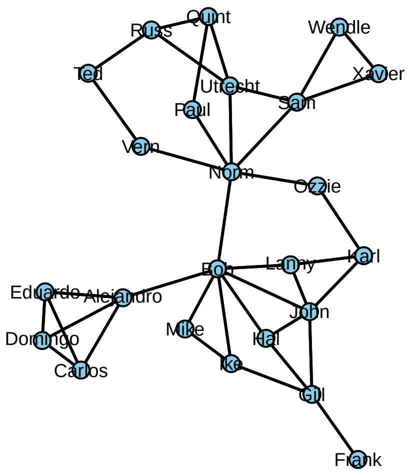
```
## sna: Tools for Social Network Analysis
## Version 2.4 created on 2016-07-23.
## copyright (c) 2005, Carter T. Butts, University of California-Irvine
##  For citation information, type citation("sna").
##  Type help(package="sna") to get started.
```

```
library(network)
library(intergraph)
library(car)
```

```
## Loading required package: carData
```

Read the network in as a network object, plot it, and then save the coordinates.

```
strike.net <- as.network(read.paj("Strike.net"),directed=FALSE)
coords <- gplot(strike.net,gmode="graph",label=network.vertex.names(strike.net),
                vertex.col="Sky Blue",label.col="black",label.cex=0.6,label.pos=5,
                mode="kamadakawai")
```

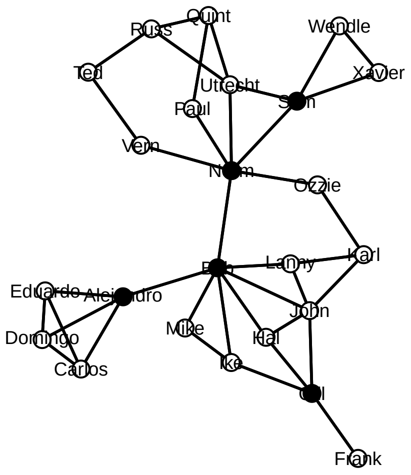


Identify cutpoints and the plot the graph with the cutpoints given a different color.

```
strike.cut <- cutpoints(strike.net,mode="graph",return.indicator=TRUE)
strike.cut
```

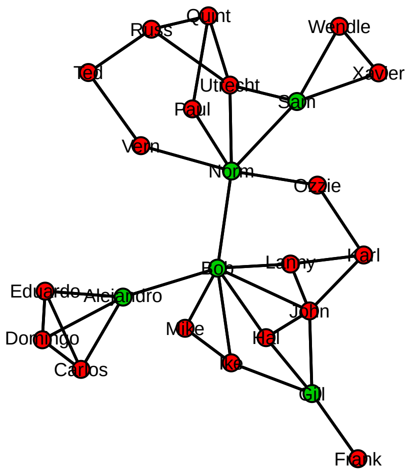
```
##     1     2     3     4     5     6     7     8     9    10    11    12
## FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE
##    13    14    15    16    17    18    19    20    21    22    23    24
## FALSE FALSE  TRUE FALSE  TRUE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE
```

```
gplot(strike.net,gmode="graph",label=network.vertex.names(strike.net),coord=coords,
      label.col="black",label.cex=0.6,vertex.col=strike.cut,label.pos=5)
```

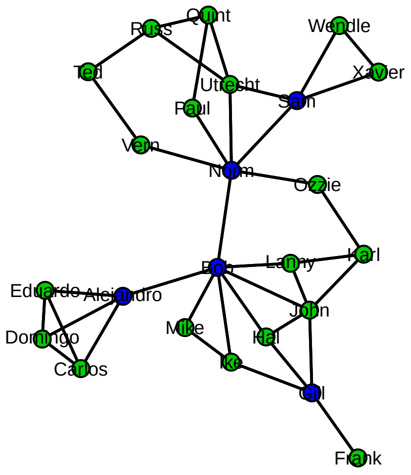


Let's change-up the color a bit. We'll add "2" to the strike.cut vector. The default colors for "TRUE" and "FALSE" are black and white. Add "2" to the vector, and they become a little Christmassy. Add 3 you get green and blue, add 4 you get blue and light blue, and so on

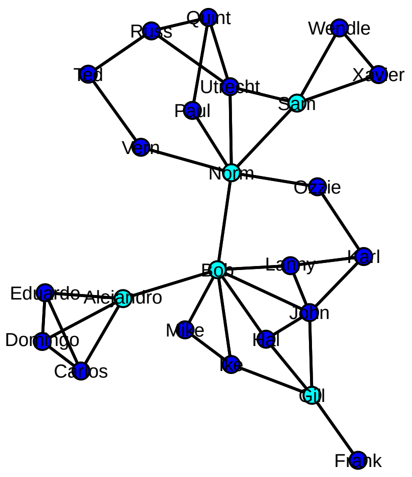
```
gplot(strike.net,gmode="graph",label=network.vertex.names(strike.net),coord=coords,
      label.col="black",label.cex=0.6,vertex.col=strike.cut+2,label.pos=5)
```



```
gplot(strike.net,gmode="graph",label=network.vertex.names(strike.net),coord=coords,
      label.col="black",label.cex=0.6,vertex.col=strike.cut+3,label.pos=5)
```

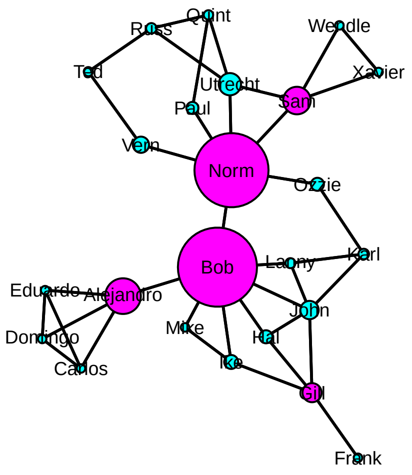


```
gplot(strike.net,gmode="graph",label=network.vertex.names(strike.net),coord=coords,
      label.col="black",label.cex=0.6,vertex.col=strike.cut+4,label.pos=5)
```



If we vary node size by betweenness centrality, we can see that a correlation exists between cutpoints and betweenness. It isn't perfect, however.

```
strike.bet <- betweenness(strike.net)
gplot(strike.net,gmode="graph",label=network.vertex.names(strike.net),coord=coords,
      label.col="black",label.cex=0.6,vertex.col=strike.cut+5,label.pos=5,
      vertex.cex=(strike.bet/75+.5))
```



Identify bicomponents, and the list (ls) the items the command generates. The command only identifies bicomponents of size 3 or greater, which you can see by typing, "strike.bc$csize". You'll also note that when a node belongs to more than one bicomponent, the command randomly assigns it to one of bicomponents.

```
strike.bc <- bicomponent.dist(strike.net)
ls(strike.bc)
```

```
## [1] "cdist"      "csize"      "members"    "membership"
```

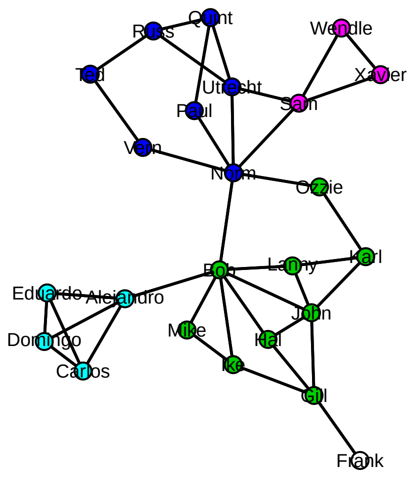
```
strike.bc$csize
```

```
##  1  2  3  4
## 10  8  4  3
```

```
strike.bc$membership
```

```
##  [1]  4  2 NA  3  2  1  2  1  1  2  4  1  1  2  4  2  1  1  1  3  3  2  3
## [24]  1
```

```
gplot(strike.net,gmode="graph",label=network.vertex.names(strike.net),coord=coords,
      label.col="black",label.cex=0.6,vertex.col=strike.bc$membership+2,label.pos=5)
```



For the Gould and Fernandez algorithm, we need the strike group membership data

```
strike.mat <- as.matrix(read.csv(("Strikegroups.csv"),header=TRUE,row.names=1,
                                  check.names=FALSE))
```

Calculate brokerage scores and then display the raw scores; where w_I = Coordinator; w_O = Itinerant; b_I0 = Representative; b_OI = Gatekeeper; b_0 = Liaison; t = Total

```
strike.gf <- brokerage(strike.net,strike.mat)
ls(strike.gf)
```

```
##  [1] "cl"      "clid"    "exp.gli" "exp.grp" "exp.nli" "n"       "N"
##  [8] "raw.gli" "raw.nli" "sd.gli"  "sd.grp"  "sd.nli"  "z.gli"   "z.nli"
```

```
strike.gf$raw.nli
```

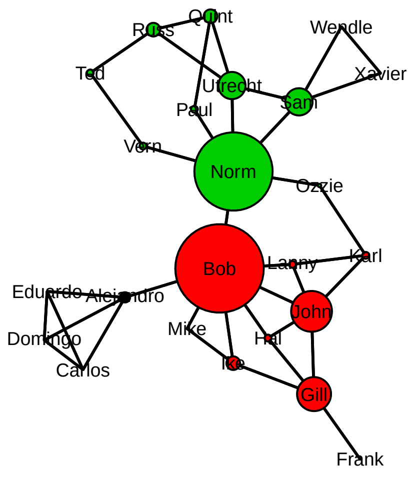
```
##           w_I w_O b_IO b_OI b_O  t
## Xavier      0   0    0    0   0  0
## Utrecht     8   0    0    0   0  8
## Frank       0   0    0    0   0  0
## Domingo     0   0    0    0   0  0
## Norm       18   0    5    5   0 28
## Hal         2   0    0    0   0  2
## Russ        4   0    0    0   0  4
## Karl        0   0    2    2   0  4
## Bob        14   0   10   10   2 36
## Quint       4   0    0    0   0  4
## Wendle      0   0    0    0   0  0
## Ozzie       0   0    1    1   0  2
## Ike         4   0    0    0   0  4
## Ted         2   0    0    0   0  2
## Sam         8   0    0    0   0  8
## Vern        2   0    0    0   0  2
## Gill       10   0    0    0   0 10
## Lanny        2   0    0    0   0  2
## Mike         0   0    0    0   0  0
## Carlos       0   0    0    0   0  0
## Alejandro    0   0    3    3   0  6
## Paul         2   0    0    0   0  2
## Eduardo      0   0    0    0   0  0
## John        12   0    0    0   0 12
```

Get total brokerage score but only count representative/gatekeeper once since it is undirected network (the total score is in the 6th column, while the gatekeeper score is in the fourth).

```
strike.gft <- strike.gf$raw.nli[,6] - strike.gf$raw.nli[,4]
```

Now, visualize where node size = total brokerage score

```
gplot(strike.net,gmode="graph",label=network.vertex.names(strike.net),coord=coords,
      label.col="black",label.cex=.6,vertex.col=strike.mat[,1],label.pos=5,
      vertex.cex=strike.gft/5)
```



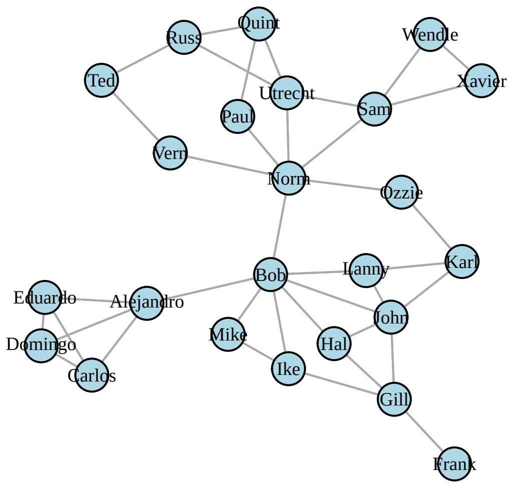
Brokers and Bridges in *igraph*

Let's move to *igraph* and see how to do things there; it also allows us to calculate edge betweenness, which *statnet* does not. We need to load *igraph* and detach *sna*, and we use *intergraph* to tranform the network into an *igraph* object. We also assign the vertex names as node labels.

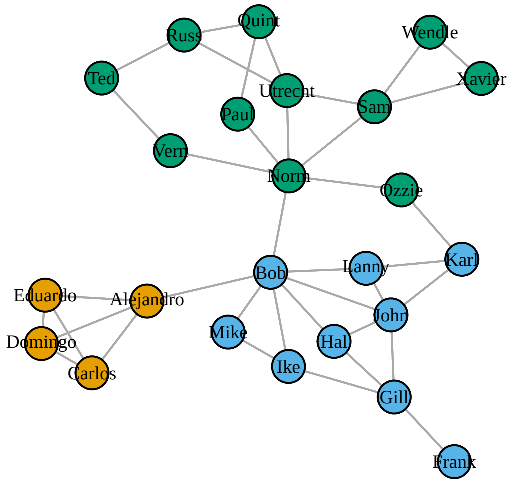
```
library(igraph)
detach("package:sna", unload=TRUE)

strike.ig <- asIgraph(strike.net)
V(strike.ig)$label = V(strike.ig)$vertex.names
```

Initial plot without and then with strike groups highlighted:

```
plot(strike.ig,layout=coords,vertex.label.cex=.6,vertex.label.color="black",
     edge.arrow.mode=0,vertex.color="light blue")
```



```
plot(strike.ig,layout=coords,vertex.label.cex=.6,vertex.label.color="black",
     edge.arrow.mode=0,vertex.color=(strike.mat))
```



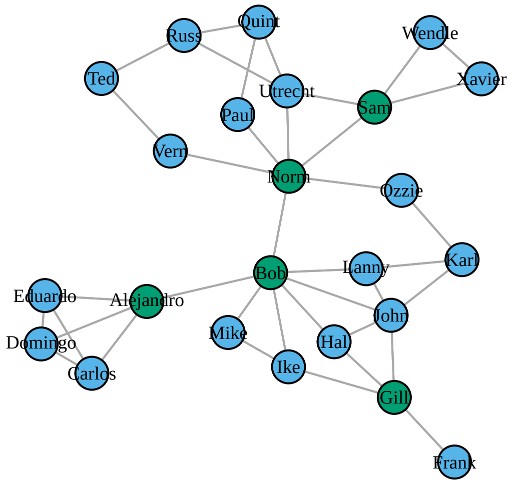
igraph has two commands that we can use (articulation.points and biconnected.components), but biconnected.components identifies both cutpoints (aka, articulation points) and bicomponents.

```
strike.bicomp <- biconnected.components(strike.ig)
ls(strike.bicomp)
## [1] "articulation_points" "component_edges"     "components"
## [4] "no"                  "tree_edges"
```

Get a list of which actors belong to which bicompoment (note that some belong to more than one – these are cutpoints) and a list of cutpoints; note that *igraph* identifies bicomponents of size 2 or greater, while *statnet* only identifies bicomponents of size 3 or greater.

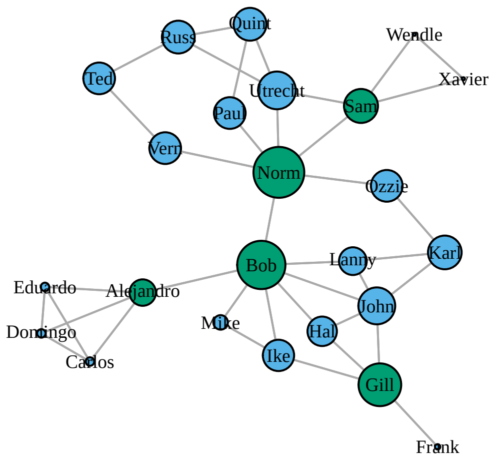
```
strike.bicomp$components
## [[1]]
## + 2/24 vertices, from 8ca18d9:
## [1] 17  3
##
## [[2]]
## + 4/24 vertices, from 8ca18d9:
## [1] 20 23  4 21
##
```

```
## [[3]]
## + 2/24 vertices, from 8ca18d9:
## [1]  9 21
##
## [[4]]
## + 10/24 vertices, from 8ca18d9:
##  [1]  8 18 12 24 17 13 19  6  9  5
##
## [[5]]
## + 8/24 vertices, from 8ca18d9:
## [1] 10 22  7 14 16  5  2 15
##
## [[6]]
## + 3/24 vertices, from 8ca18d9:
## [1] 11 15  1
strike.bicomp$articulation_points
## + 5/24 vertices, from 8ca18d9:
## [1] 17 21  9  5 15

plot(strike.ig,layout=coords,vertex.label.cex=.6,vertex.label.color="black",
     edge.arrow.mode=0,vertex.color=strike.cut+2)
```



Calculate Burt’s constraint and its additive inverse (autonomy)

```
strike.con <- constraint(strike.ig)
strike.aut <- 1-strike.con

plot(strike.ig,layout=coords,vertex.label.cex=.6,vertex.label.color="black",
     edge.arrow.mode=0,vertex.color=strike.cut+2,vertex.size=strike.aut*30)
```

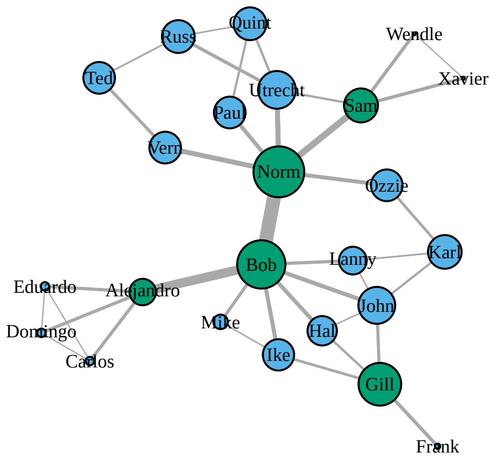


Now, calculate edge betweenness

```
strike.edge <- edge.betweenness(strike.ig,directed = FALSE,weights = NULL)
```

And then plot where edge width equals edge betweenness

```
plot(strike.ig,layout=coords,vertex.label.cex=.6,vertex.label.color="black",
     edge.arrow.mode=0,vertex.color=strike.cut+2,vertex.size=strike.aut*30,
     edge.width=strike.edge/10+.5)
```



That's all for now