Centrality Metrics

Contents

Zachary karate club 7

Very simple examples 19

Political blogs 21


```r
#install.packages('xtable')
library('igraph')
```

Centrality measures:

- Undirected graphs:
  - Actor centrality - involvement (connections) with other actors
- Directed graphs:
  - Actor centrality - source of the ties (outgoing edges)
  - Actor prestige - recipient of many ties (incoming edges)

Degree centrality: number of nearest neighbors.

$$C_D(i) = k(i) = \sum_j A_{ij} = \sum_j A_{ji}$$

Normalized degree centrality

$$C_D^*(i) = \frac{1}{n-1} C_D(i) = \frac{k(i)}{n-1}$$

High centrality degree - direct contact with many other actors.

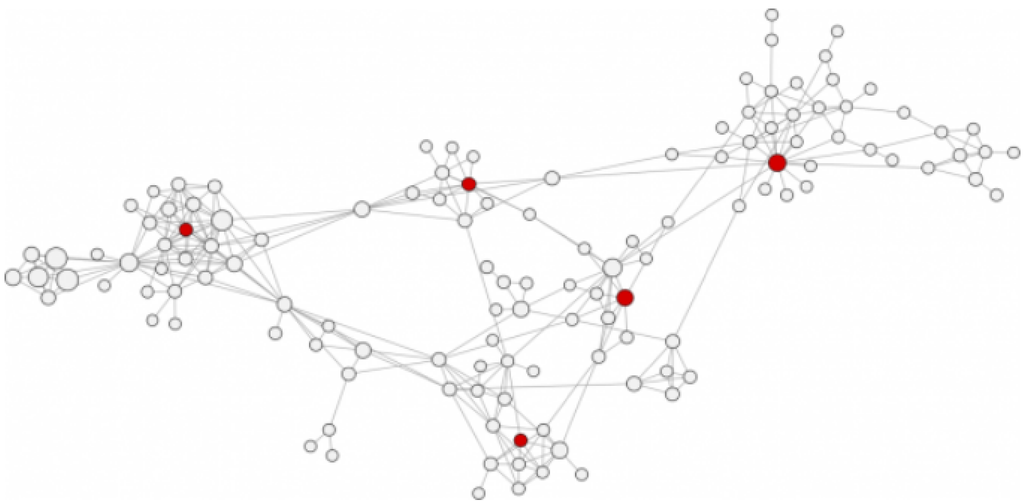
Closeness centrality: how close an actor to all the other actors in network.

Normalized closeness centrality:

$$C_C^*(i) = (n-1)C_c(i) = \frac{n-1}{\sum_j d(i,j)}$$

High closeness centrality - short communication path to others, minimal number of steps to reach others.

Closeness centrality



igraph:closeness()

Betweenness centrality: number of shortest paths going through the actor $\sigma_{st}(i)$

$$C_B(i) = \sum_{s \neq t \neq i} \frac{\sigma_{st}(i)}{\sigma_{st}}$$

Normalized betweenness centrality:

$$C_B^* = \frac{2}{(n-1)(n-2)} C_B(i) = \frac{2}{(n-1)(n-2)} \sum_{s \neq t \neq i} \frac{\sigma_{st}(i)}{\sigma_{st}}$$

High betweenness centrality - vertex lies on many shortest paths. Probability that a communication from s to t will go through i.

Betweenness centrality

igraph:betweenness()

Eigenvector centrality:

Importance of a node depends on the importance of its neighbors (recursive definition).

$$v_i \leftarrow \sum_j A_{ij} v_j$$

$$v_i = \frac{1}{\lambda} \sum_j A_{ij} v_j$$

$$Av = \lambda v$$

Select an eigenvector associated with largest eigenvalue $\lambda = \lambda_1, v = v_1$

Eigenvector centrality

igraph:evcent()

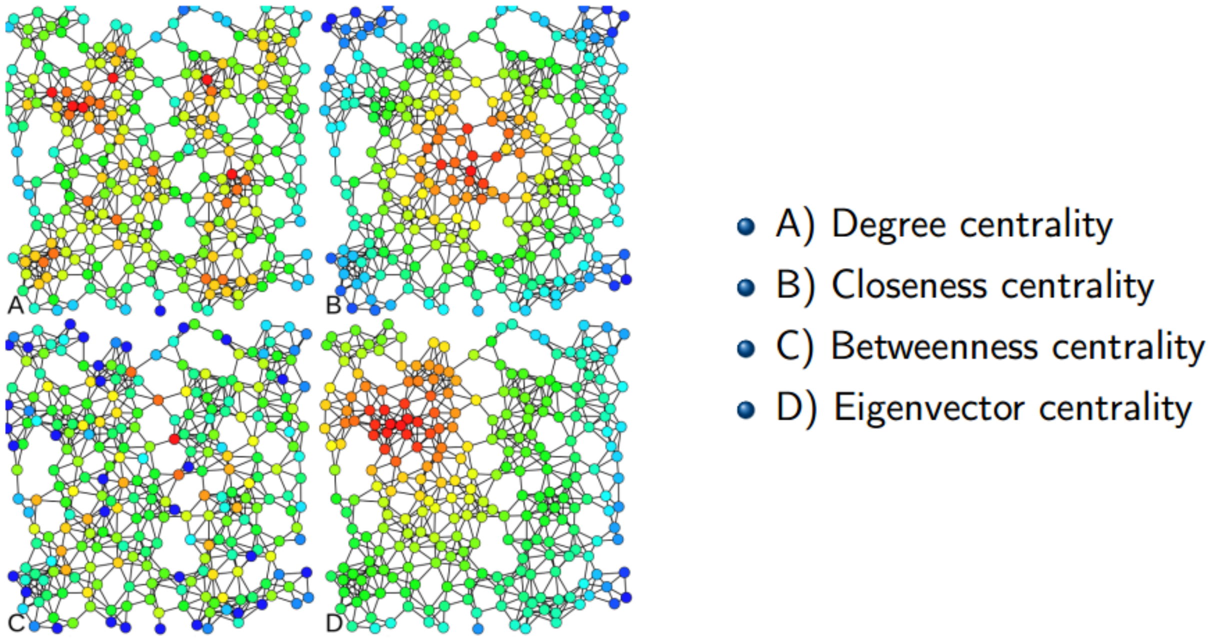
Centralization:

Centralization (network measure) - how central the most central node in the network in relation to all other nodes.

$$C_x = \frac{\sum_i^N [C_X(p_*) - C_x(p_i)]}{\max \sum_i^N [C_X(p_*) - C_x(p_i)]}$$

C_x - one of the centrality measures, p_* - node with the largest centrality value, max - is taken over all graphs with the same number of nodes (for degree, closeness and betweenness the most centralized structure is the star graph).

igraph: centralization.degree(), centralization.closeness(), centralization.betweenness(), centralization.evcent()



from Claudio Rocchini

Directional relations:

Directed graph: distinguish between choices made (outgoing edges) and choices received (incoming edges)

Centrality measures:

All based on outgoing edges.

- Degree centrality (normalized):

$$C_D^*(i) = \frac{k^{out}(i)}{n-1}$$

- Closeness centrality (normalized):

$$C_C^*(i) = \frac{n-1}{\sum_j d(i,j)}$$

- Betweenness centrality (normalized):

$$C_B^*(i) = \frac{1}{(n-1)(n-2)} \sum_{s \neq t \neq i} \frac{\sigma_{st}(i)}{\sigma_{st}}$$

Status / Rank Prestige:

Node prestige depends on prestige of directly connected actors.

- iterate:

$$p_i \leftarrow \sum_{j \in N(i)} p_j = \sum_j A_{ji} p_j$$

$$p^{t+1} = A^t p^t, p^{t=0} = p_0$$

- Difficulties:
  - Absorbing nodes
  - Source nodes
  - Cycles

Solution to $p = A^T p$ might not exist. Nontrivial solution only if $det(I - A^T) = 0$. Need to constraint matrix.

Random walk:

Random walk on graph:

$$p_i^{t+1} = \sum_{j \in N(i)} \frac{p_j^t}{d_j^{out}} = \sum_j \frac{A_{ji}}{d_j^{out}} p_j$$

$$P = D^{-1} A, D_{ii} = diag d_i^{out}$$

$$p^{t+1} = P^T p^t$$

- with teleportation

$$p^{t+1} = \alpha P^T p^t + (1 - \alpha) \frac{e}{n}$$

Perron-Frobenius Theorem guarantees existence and uniqueness of the solution to

$$p = \alpha P^T p + (1 - \alpha) \frac{e}{n}$$

Hubs and Authorities (HITS)

Citation networks. Reviews vs original research (authoritative) papers

- authorities, contain useful information, a_i
- hubs, contains links to authorities, h_i

Mutual recursion

- Good authorities referred by good hubs

$$a_i \leftarrow \sum_j A_{ji} h_j$$

- Good hubs point to good authorities

$$h_i \leftarrow \sum_j A_{ij} a_j$$

System of linear equations

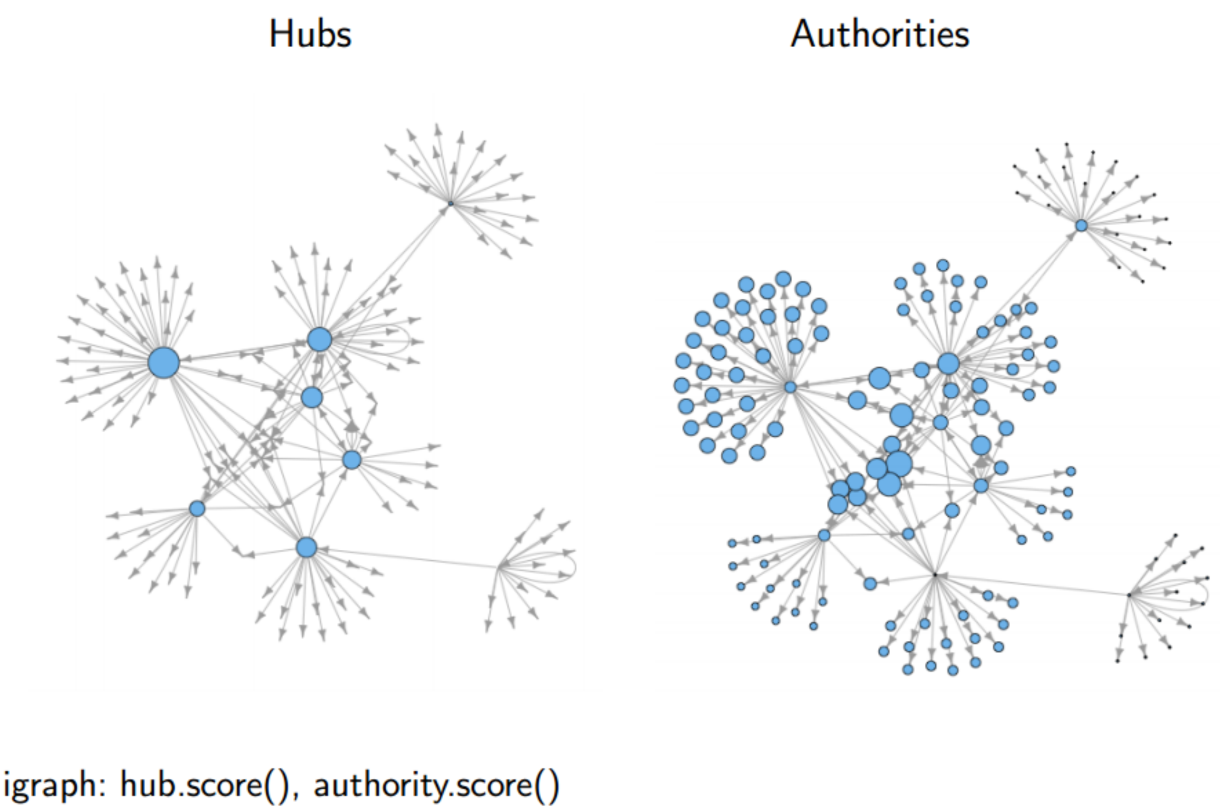
$$a = \alpha A^T h$$
$$h = \beta A a$$

Symmetric eigenvalue problem

$$(A^T A)a = \lambda a$$
$$(AA^T)h = \lambda h$$

where eigenvalue

$$\lambda = (\alpha\beta)^{-1}$$



First, let's look at a simple example graph:

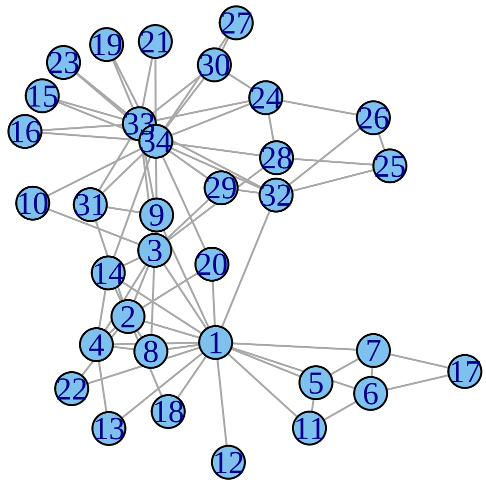
Zachary karate club

Remember Zachary karate club? You looked at it on your second lab.

The data was collected from the members of a university karate club by Wayne Zachary in 1977. Each node represents a member of the club, and each edge represents a tie between two members of the club. Zachary studied conflict and fission in this community network, as the karate club was split into two separate clubs. The network is very small: it has 34 vertices and 78 undirected edges.

This is how it looks:

```
z <- graph.famous("Zachary")
plot(z,layout = layout.fruchterman.reingold)
```



Degree centrality

Now we will compute various centrality measures. First, degree centrality:

```
deg=degree(z)
```

We will plot the karate club graph with node sizes proportional to different centrality metrics and node colors changing depending on centrality.

First, let's fix node coordinates to be able to compare graphs. We will produce coordinates and use them as layout for plotting:

```
# And we will keep the same layout:
lay <- layout.fruchterman.reingold(z)
lay
```

```
##             [,1]        [,2]
##  [1,] 22.441637 -11.6267342
##  [2,] 15.905352 -18.3588093
##  [3,] 12.596984  -9.9070052
##  [4,] 20.820913 -17.2078729
##  [5,] 33.481772  -9.2989170
##  [6,] 29.417370  -1.0912729
##  [7,] 32.984940  -3.3942852
##  [8,] 19.565882 -14.1484871
```

```
##  [9,]  9.236011  -9.1172627
## [10,]  3.316061 -17.0017376
## [11,] 29.837409  -6.6374449
## [12,] 33.276903 -16.3222906
## [13,] 28.294750 -19.6569427
## [14,] 14.551056 -12.8044179
## [15,] -7.707525  -4.2922978
## [16,] -3.171690 -14.0364541
## [17,] 34.905974   4.1859713
## [18,] 23.695729 -22.8336703
## [19,] -6.853443  -8.0106702
## [20,] 11.054763 -16.1083411
## [21,] -6.273273 -11.7084597
## [22,] 19.455613 -24.3402741
## [23,] -5.530952  -0.9148843
## [24,]  3.713736   4.0779482
## [25,] 14.679841   9.3266334
## [26,]  9.837339  10.4524134
## [27,] -4.246560   5.3198293
## [28,]  9.415463   2.2764834
## [29,]  9.441192  -2.9968715
## [30,] -1.106614   2.1761830
## [31,]  5.494282 -13.5569600
## [32,] 12.437251  -0.1458473
## [33,]  0.765992  -5.6697390
## [34,]  2.819303  -6.0850713
```

Now, we want node colors to change depending on the centrality. Say, high centrality nodes will be green, and low centrality nodes will be yellow. We set the pale

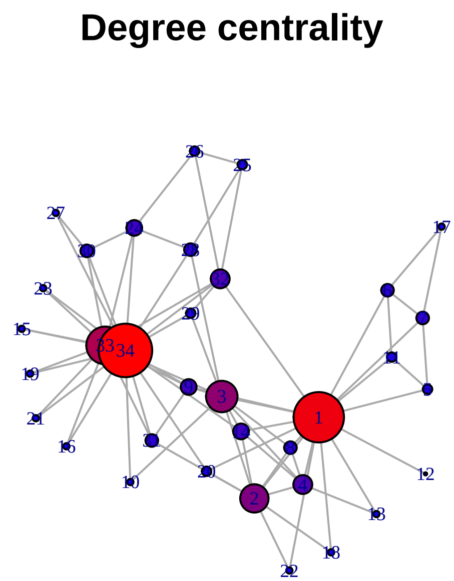
```
fine = 500 # this will adjust the resolving power.
palette = colorRampPalette(c('blue','red'))
```

Now we produce a vector of color codes:

```
degCol = palette(fine)[as.numeric(cut(deg,breaks = fine))]
```

We also want node sizes to be proportional to node centrality. This is straightforward, except that we will use a proportionality coefficient to improve the layout. Now produce a graph:

```
plot(z, layout=lay, vertex.color=degCol, vertex.size=deg*1.5, vertex.label.cex=0.6, main="Degree central
```



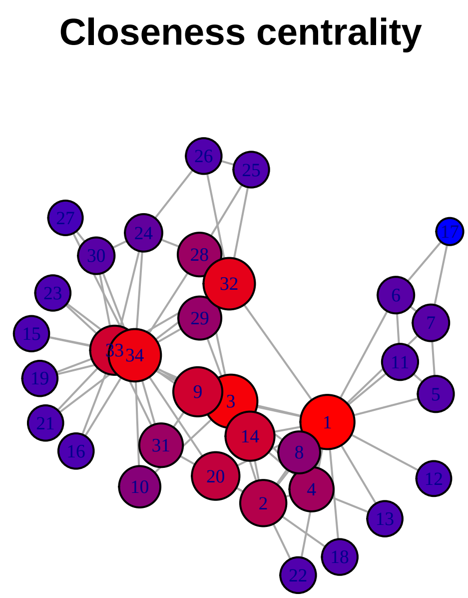
Closeness centrality

Now compute closeness centrality and plot the second graph. Simply use the command `closeness`

```
clos=closeness(z)
#Look at the values:
clos
```

```
##  [1] 0.01724138 0.01470588 0.01694915 0.01408451 0.01149425 0.01162791
##  [7] 0.01162791 0.01333333 0.01562500 0.01315789 0.01149425 0.01111111
## [13] 0.01123596 0.01562500 0.01123596 0.01123596 0.00862069 0.01136364
## [19] 0.01123596 0.01515152 0.01123596 0.01136364 0.01123596 0.01190476
## [25] 0.01136364 0.01136364 0.01098901 0.01388889 0.01369863 0.01162791
## [31] 0.01388889 0.01639344 0.01562500 0.01666667
```

```
# Plot the graph:
closCol = palette(fine)[as.numeric(cut(clos,breaks = fine))]
plot(z,layout = lay, vertex.color=closCol, vertex.size=clos*1500, vertex.label.cex=0.6, main="Closeness
```



Betweness centrality

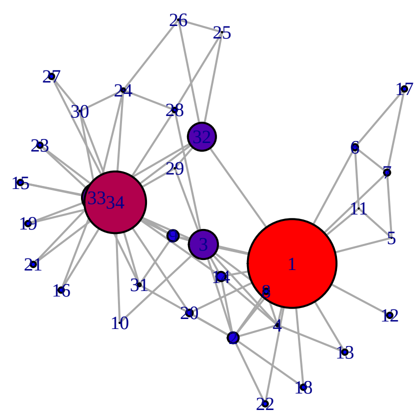
Compute betwenness centrality. Can you guess the command? Right, it's `betweenness`

```
betw <- betweenness(z)
#Look at the values:
betw
```

```
##  [1] 231.0714286  28.4785714  75.8507937   6.2880952   0.3333333
##  [6]  15.8333333  15.8333333   0.0000000  29.5293651   0.4476190
## [11]   0.3333333   0.0000000   0.0000000  24.2158730   0.0000000
## [16]   0.0000000   0.0000000   0.0000000   0.0000000  17.1468254
## [21]   0.0000000   0.0000000   0.0000000   9.3000000   1.1666667
## [26]   2.0277778   0.0000000  11.7920635   0.9476190   1.5428571
## [31]   7.6095238  73.0095238  76.6904762 160.5515873
```

```
#Plot the graph
betwCol = palette(fine)[as.numeric(cut(betw,breaks = fine))]
plot(z,layout = lay, vertex.color=betwCol, vertex.size=betw*0.2, vertex.label.cex=0.6, main="Betwenness
```

Betwenness centrality



Eigenvector centrality

Now, eigenvector centrality. Use the command `evcent`

```r
ev <- evcent(z)

# See what's in the output:
ev
```

```
## $vector
##  [1] 0.95213237 0.71233514 0.84955420 0.56561431 0.20347148 0.21288383
##  [7] 0.21288383 0.45789093 0.60906844 0.27499812 0.20347148 0.14156633
## [13] 0.22566382 0.60657439 0.27159396 0.27159396 0.06330461 0.24747879
## [19] 0.27159396 0.39616224 0.27159396 0.24747879 0.27159396 0.40207086
## [25] 0.15280670 0.15857597 0.20242852 0.35749923 0.35107297 0.36147301
## [31] 0.46806481 0.51165649 0.82665886 1.00000000
##
## $value
## [1] 6.725698
##
## $options
## $options$bmat
## [1] "I"
##
## $options$n
```

```
## [1] 34
##
## $options$which
## [1] "LA"
##
## $options$nev
## [1] 1
##
## $options$tol
## [1] 0
##
## $options$ncv
## [1] 0
##
## $options$ldv
## [1] 0
##
## $options$ishift
## [1] 1
##
## $options$maxiter
## [1] 3000
##
## $options$nb
## [1] 1
##
## $options$mode
## [1] 1
##
## $options$start
## [1] 1
##
## $options$sigma
## [1] 0
##
## $options$sigmai
## [1] 0
##
## $options$info
## [1] 0
##
## $options$iter
## [1] 2
##
## $options$nconv
## [1] 1
##
## $options$numop
## [1] 26
##
## $options$numopb
## [1] 0
##
## $options$numreo
```

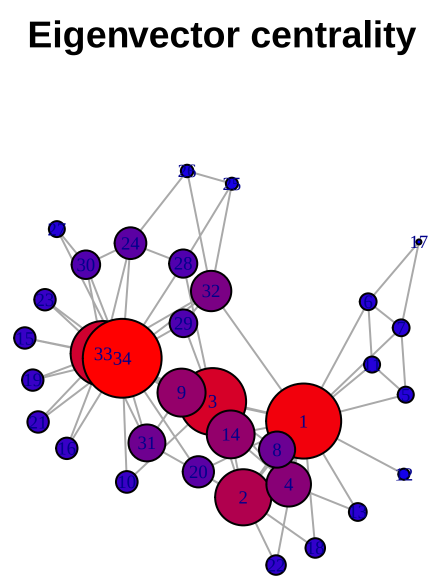
```
## [1] 21
```

The function returns the vector of eigenvector centralities, the eigenvalue corresponding to the eigenvector, and a list of options used for computations. We only need the centrality values, don't we?

```
ev <- evcent(z)$vector
# See what's in the output:
ev
```

```
##  [1] 0.95213237 0.71233514 0.84955420 0.56561431 0.20347148 0.21288383
##  [7] 0.21288383 0.45789093 0.60906844 0.27499812 0.20347148 0.14156633
## [13] 0.22566382 0.60657439 0.27159396 0.27159396 0.06330461 0.24747879
## [19] 0.27159396 0.39616224 0.27159396 0.24747879 0.27159396 0.40207086
## [25] 0.15280670 0.15857597 0.20242852 0.35749923 0.35107297 0.36147301
## [31] 0.46806481 0.51165649 0.82665886 1.00000000
```

```
# Produce the plot:
evCol = palette(fine)[as.numeric(cut(ev,breaks = fine))]
plot(z,layout = lay, vertex.size=ev*40, vertex.color=evCol, vertex.label.cex=0.6, main="Eigenvector cen
```



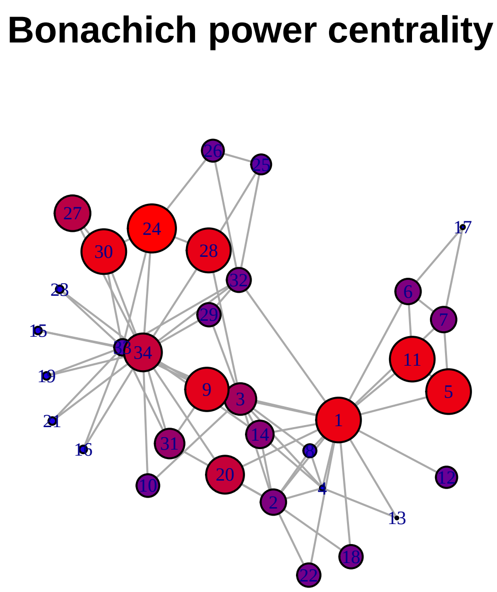
Bonacich power centrality

Bonachich power centrality:

```r
bon <- bonpow(z, rescale=TRUE)
bon
```

```
##  [1] 0.052588997 0.029935275 0.037216828 0.005663430 0.052588997
##  [6] 0.029935275 0.029935275 0.015372168 0.050970874 0.026699029
## [11] 0.052588997 0.025080906 0.003236246 0.032362460 0.008899676
## [16] 0.008899676 0.004854369 0.027508091 0.008899676 0.044498382
## [21] 0.008899676 0.027508091 0.008899676 0.056634304 0.023462783
## [26] 0.025889968 0.042071197 0.051779935 0.027508091 0.052588997
## [31] 0.034789644 0.028317152 0.019417476 0.044498382
```

```r
#Produce the plot
bonCol = palette(fine)[as.numeric(cut(bon,breaks = fine))]
plot(z,layout = lay, vertex.size=bon*400, vertex.color=bonCol, vertex.label.cex=0.6, main="Bonachich po
```



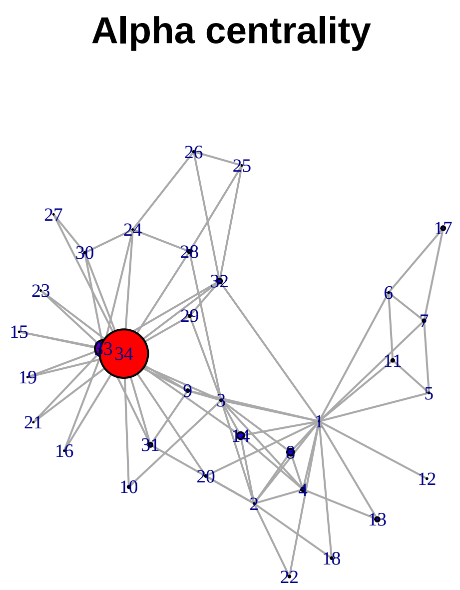
Alpha centrality

Alpha centrality:

```r
alpha <- alpha.centrality(z)
alpha
```

```
##  [1]   1   2   4   8   2   2   6  16   6   5   6   2  10  16   1   1   9
## [18]   4   1   4   1   4   1   1   1   3   1   7   5   3   9  11  40 114
```

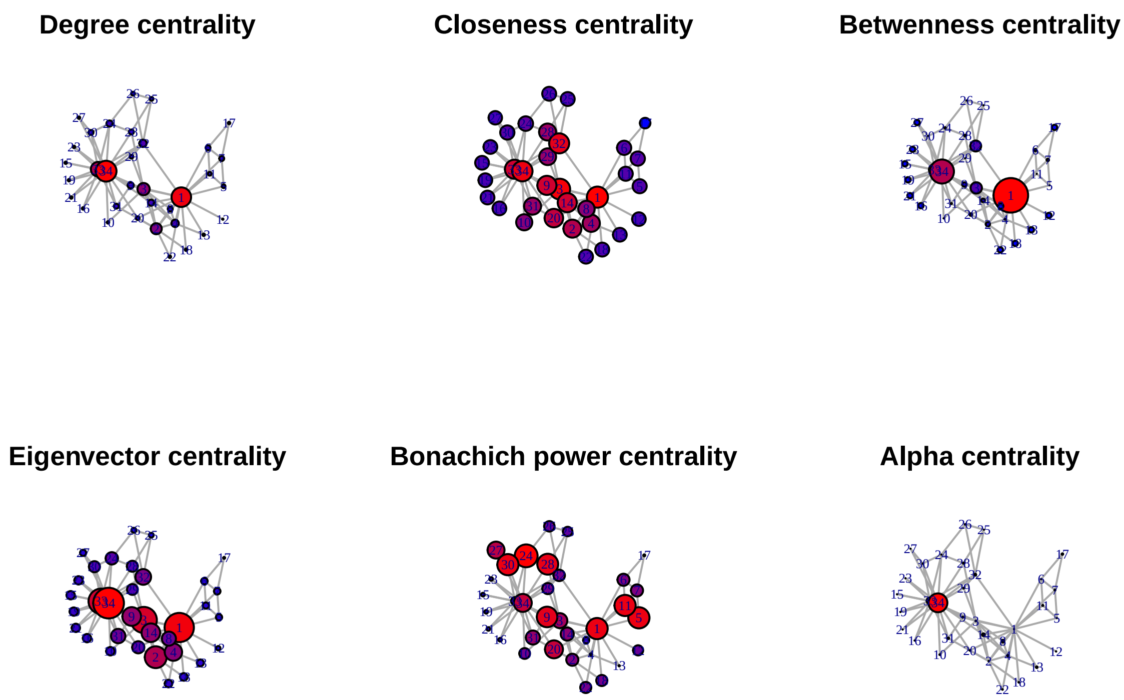
```
#Produce the plot
alphaCol = palette(fine)[as.numeric(cut(alpha,breaks = fine))]
plot(z,layout = lay, vertex.size=alpha*0.2, vertex.color=alphaCol, vertex.label.cex=0.6, main="Alpha cen
```



Compare measures

Now, let's plot all graphs together. Remember how to produce several plots on the same graph?

```
# We will plot 6 graphs in 2 rows and 3 columns:
op <- par(mfrow = c(2, 3))
#Remember we assigned a name to each graph?
plot(z, layout=lay, vertex.color=degCol, vertex.size=deg*1.5, vertex.label.cex=0.6, main="Degree centra
plot(z,layout = lay, vertex.color=closCol, vertex.size=clos*1500, vertex.label.cex=0.6, main="Closeness
plot(z,layout = lay, vertex.color=betwCol, vertex.size=betw*0.2, vertex.label.cex=0.6, main="Betwenness
plot(z,layout = lay, vertex.size=ev*40, vertex.color=evCol, vertex.label.cex=0.6, main="Eigenvector cen
plot(z,layout = lay, vertex.size=bon*500, vertex.color=bonCol, vertex.label.cex=0.6, main="Bonachich pow
plot(z,layout = lay, vertex.size=alpha*0.2, vertex.color=alphaCol, vertex.label.cex=0.6, main="Alpha cen
```




```r
par(op)
```

Print degrees with maximal values for each ranking:

```r
which.max(deg)
```

```
## [1] 34
```

```r
which.max(clos)
```

```
## [1] 1
```

```r
which.max(betw)
```

```
## [1] 1
```

```r
which.max(ev)
```

```
## [1] 34
```

```r
which.max(bon)
```

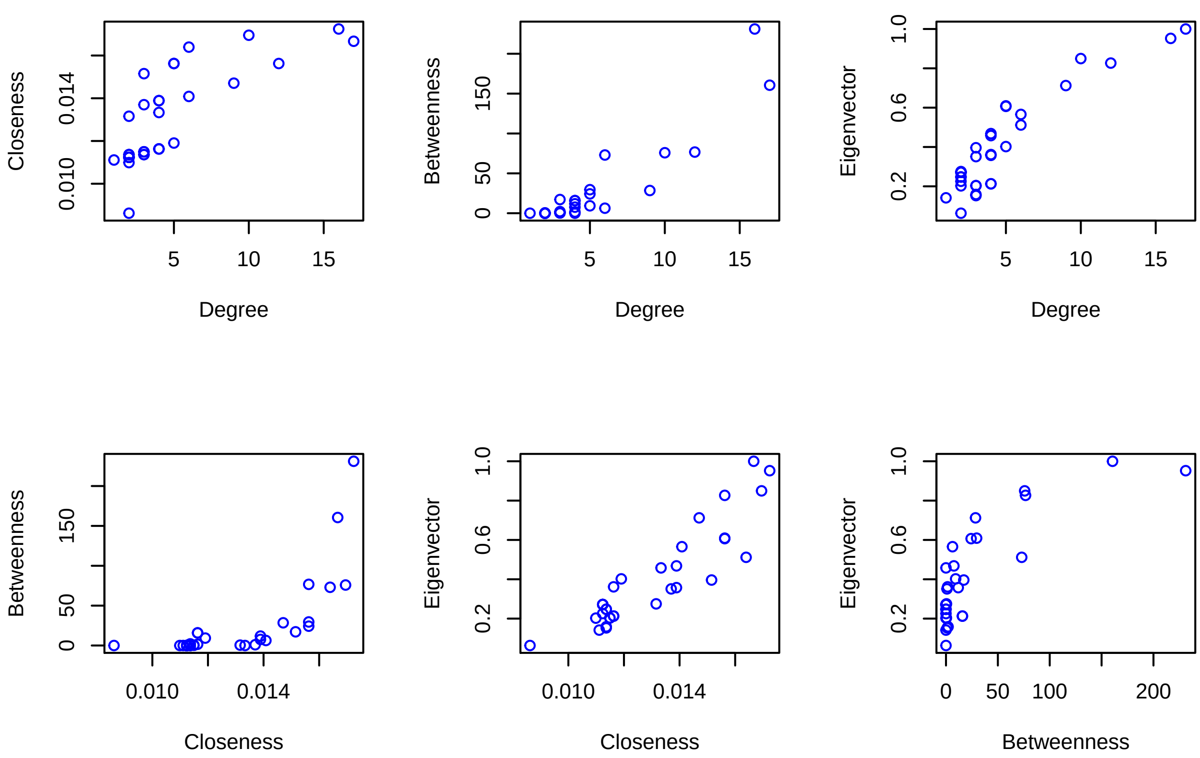
```
## [1] 24
```

```r
which.max(alpha)
```

```
## [1] 34
```

Now we will plot degree metrics against each other. We will only plot degree centrality, closeness, betweeness and eigenvector centralities. You can plot the remaining at home.

```r
op <- par(mfrow = c(2, 3))
plot(deg, clos, xlab="Degree", ylab="Closeness", col="blue")
plot(deg, betw, xlab="Degree", ylab="Betweenness", col="blue")
plot(deg, ev, xlab="Degree", ylab="Eigenvector", col="blue")
plot(clos, betw, xlab="Closeness", ylab="Betweenness", col="blue")
plot(clos, ev, xlab="Closeness", ylab="Eigenvector", col="blue")
plot(betw, ev, xlab="Betweenness", ylab="Eigenvector", col="blue")
```



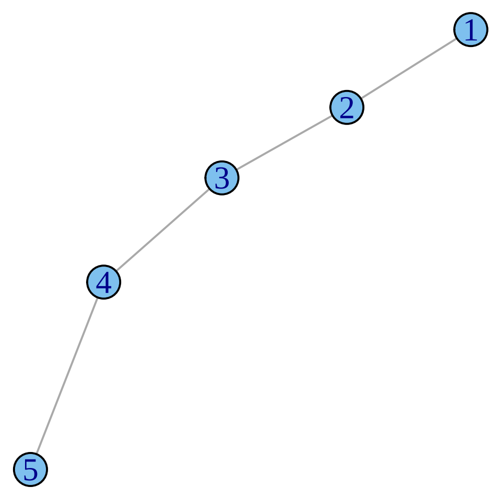

```r
par(op)
```

Very simple examples

Centrality metrics for a path

We will create a path of 5 nodes, plot it, compute and output centrality metrics for the nodes:

```
p <- graph.tree(5, children=1, mode="undirected")
plot(p)
```



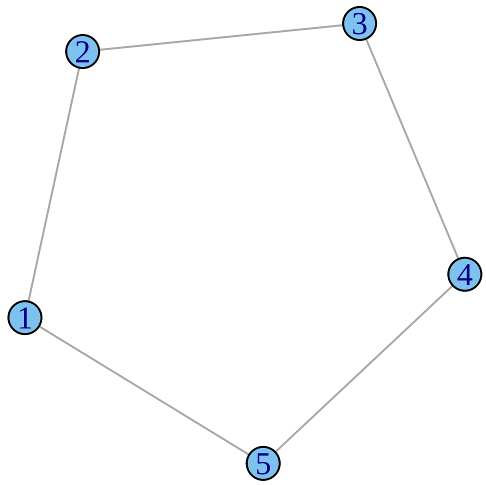

```
ptable <- cbind(degree(p), round(closeness(p), 3), betweenness(p), round(evcent(p)$vector, 3))
titles <- c("Degree", "Closeness", "Betweenness", "Eigenvector")
colnames(ptable) <- titles
ptable
```

```
##      Degree Closeness Betweenness Eigenvector
## [1,]      1     0.100           0       0.500
## [2,]      2     0.143           3       0.866
## [3,]      2     0.167           4       1.000
## [4,]      2     0.143           3       0.866
## [5,]      1     0.100           0       0.500
```

Centrality metrics for a cyrcle

Now we will create a cyrcle of 5 nodes, plot it, compute and output centrality metrics for the nodes:

```r
cyr <- graph.ring(5)
plot(cyr)
```



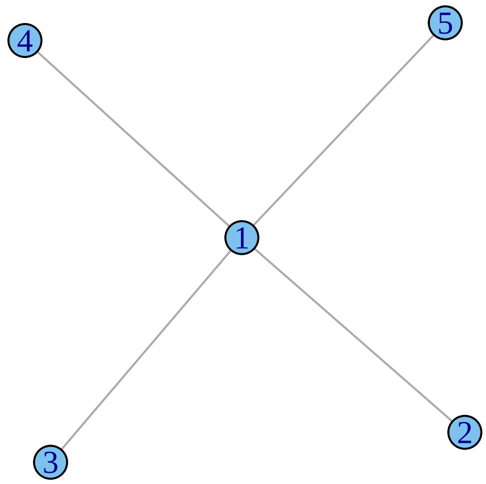

```r
cyrtable <- cbind(degree(cyr), round(closeness(cyr), 3), betweenness(cyr), round(evcent(cyr)$vector, 3))
colnames(cyrtable) <- titles
cyrtable
```

```
##      Degree Closeness Betweenness Eigenvector
## [1,]      2     0.167           1           1
## [2,]      2     0.167           1           1
## [3,]      2     0.167           1           1
## [4,]      2     0.167           1           1
## [5,]      2     0.167           1           1
```

Centrality metrics for a star

Finally, we will do the same things for a star of 5 nodes:

```r
star <- graph.star(5, mode="undirected")
plot(star)
```




```
startable <- cbind(degree(star), round(closeness(star), 3), betweenness(star), round(evcent(star)$vecto
colnames(startable) <- titles
startable
```

```
##      Degree Closeness Betweenness Eigenvector
## [1,]      4     0.250           6         1.0
## [2,]      1     0.143           0         0.5
## [3,]      1     0.143           0         0.5
## [4,]      1     0.143           0         0.5
## [5,]      1     0.143           0         0.5
```

Political blogs

Please download the network from here. This is a directed network of hyperlinks between weblogs on US politics, recorded in 2005. Each node has an attribute 'value' correspondent to its political side: 0 - liberal, 1 - conservative.

Load the data

```
PB <- read.graph(file = 'polblogs.gml', format = 'gml')
vcount(PB)
```

```
## [1] 1490
```

```r
ecount(PB)
```

```
## [1] 19090
```

```r
attr <- vertex.attributes(graph = PB)
```

And you can plot it if you wish (it takes some time..)

```r
plot(PB, vertex.label = NA, vertex.size = 3.5,
     layout = layout.fruchterman.reingold,
     edge.arrow.size = 0.2)
```

Compute degree centrality, PageRank, Hubs and Authorities

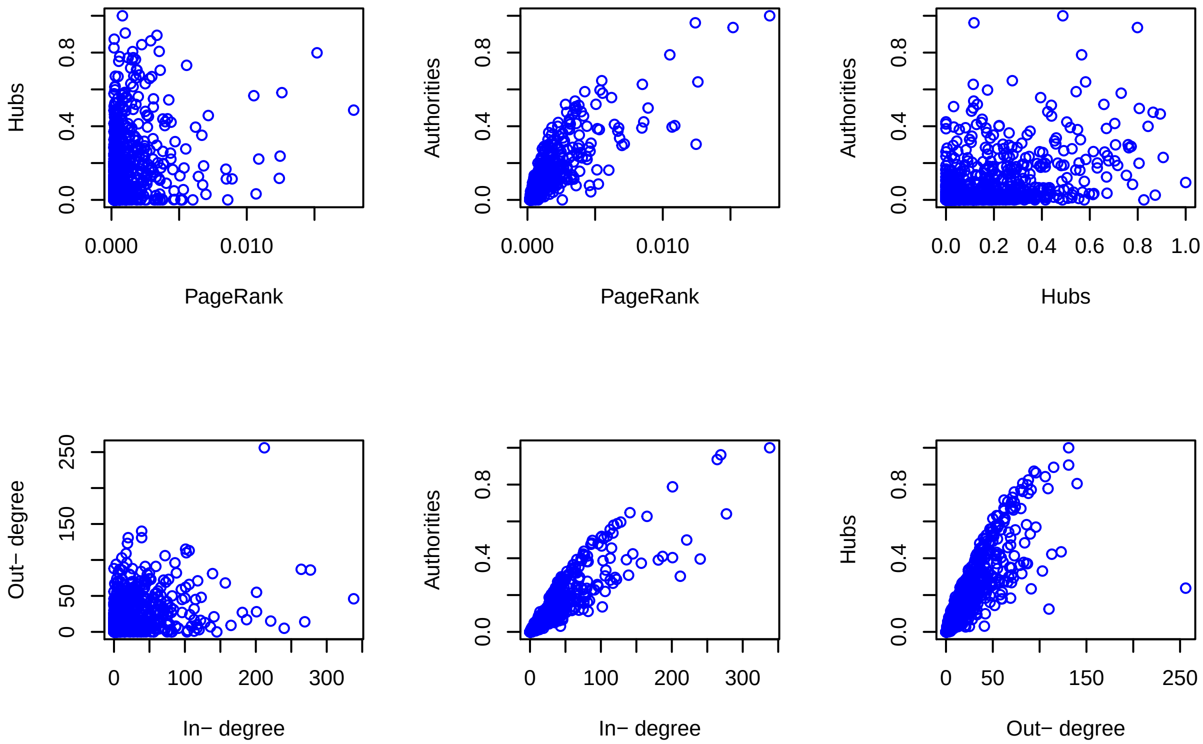
Now we will compute in- and out- degree centralities, PageRank, Hubs and Authorities for this network. Yet again, we will use igraph functions:

```r
#Incoming degrees:
indegPB=degree(PB, mode="in")

#Outgoing degrees:
outdegPB=degree(PB, mode="out")

#PageRank:
prPB=page.rank(PB)$vector
## Note that page.rank function returns a vector of values, an eigenvalue and computational options. We

#Hubs:
hPB=hub.score(PB)$vector

#Authorities:
authPB=authority.score(PB)$vector
```

Output the results

First, let's plot these measures agains each other. We will only plot several pairs, you can do the remaining at home.

```r
op <- par(mfrow = c(2, 3))
plot(prPB, hPB, xlab="PageRank", ylab="Hubs", col="blue")
plot(prPB, authPB, xlab="PageRank", ylab="Authorities", col="blue")
plot(hPB, authPB, xlab="Hubs", ylab="Authorities", col="blue")
plot(indegPB, outdegPB, xlab="In- degree", ylab="Out- degree", col="blue")
plot(indegPB, authPB, xlab="In- degree", ylab="Authorities", col="blue")
plot(outdegPB, hPB, xlab="Out- degree", ylab="Hubs", col="blue")
```




```
par(op)
```

Now we will print top ten names in each ranking:

```
#For in- degrees:
indegnamesPB=which(indegPB>sort(indegPB)[vcount(PB)-10])

#For out- degrees:
outdegnamesPB=which(outdegPB>sort(outdegPB)[vcount(PB)-10])

#For PageRank:
prnamesPB=which(prPB>sort(prPB)[vcount(PB)-10])

#For Hubs:
hnamesPB=which(hPB>sort(hPB)[vcount(PB)-10])

#For Authorities:
authnamesPB=which(authPB>sort(authPB)[vcount(PB)-10])

##Create a matrix to output:
topnamesPB=cbind(indegnamesPB, authnamesPB, prnamesPB, outdegnamesPB, hnamesPB)
#Assign column names:
colnames(topnamesPB) <- c("In- degree", "Authorities", "PageRank", "Out- degree", "Hubs")
topnamesPB
```

```
##         In- degree Authorities PageRank Out- degree Hubs
```

```
## [1,]        55          55        55          144   55
## [2,]       155         155       155          363   56
## [3,]       641         180       641          387   99
## [4,]       729         323       729          454  144
## [5,]       855         493       798          512  363
## [6,]       963         641       855          524  387
## [7,]      1051         642       963          855  454
## [8,]      1153         729      1051          880  512
## [9,]      1245         756      1153         1000  618
## [10,]     1437        1051      1245         1101  644
```

We want a nice table in HTML, don't we? Let's use an `xtable` package. Please install it now, load the library and use `xtable` function:

```
library(xtable)
toptablePB <- xtable(topnamesPB)
print(toptablePB, floating=FALSE, type="latex")
```

% latex table generated in R 3.1.3 by xtable 1.7-4 package % Thu May 14 11:38:59 2015

	In- degree	Authorities	PageRank	Out- degree	Hubs
1	55	55	55	144	55
2	155	155	155	363	56
3	641	180	641	387	99
4	729	323	729	454	144
5	855	493	798	512	363
6	963	641	855	524	387
7	1051	642	963	855	454
8	1153	729	1051	880	512
9	1245	756	1153	1000	618
10	1437	1051	1245	1101	644