igraphLesson

Ethan Thibault

4/23/2018

Today, I am going to talk about the package "igraph". This is a network analysis and visualization package that can be widely used across many different datasets. It can allow you to visualize your data and identify key players within the network, communities, cliques and more. I was hoping to use some of my own data today to show a useful application, but upon doing some of the analysis, it turned out that certain lines of code took 3 hours to complete, so we will use some datasets for igraph from the package "igraphdata".

I will briefly talk about the data I had so you can see an application that won't be in the igraphdata datasets. I am working with a population of beetles that had their RNA extracted and sequenced to both analyze transcription levels as well as perform population genetic analyses on the acquired SNP data. Another package in R called Weighted Correlation Network Analysis (WGCNA) can analyze the expression data and group them into modules. These modules can then be converted to igraph class objects and analyzed for different centrality measures in igraph.

So let's get started!

First, we want to set our working directory, install the packages "igraph" and "igraphdata", and then load their libraries. The package igraphdata, will supply us with multiple premade datasets that we can work with in igraph.

```
setwd("~/Documents/UVM_2018/BIO381")
# install.packages("igraph")
# install.packages("igraphdata")
library(igraphdata)
library(igraph)
```

```
##
## Attaching package: 'igraph'
```

```
## The following objects are masked from 'package:stats':
##
##     decompose, spectrum
```

```
## The following object is masked from 'package:base':
##
##     union
```

Before we start working with the large datasets, I want to introduce you all to some of the basics in creating objects of class igraph and its structure.

```
# Here are some simple commands to make very basic small networks
g1 <- graph(edges = c(1,2,2,3,3,1), n = 3, directed = F)
class(g1)
```

```
## [1] "igraph"
```

```
g1
```

```
## IGRAPH aa64c24 U--- 3 3 --
## + edges from aa64c24:
## [1] 1--2 2--3 1--3
```

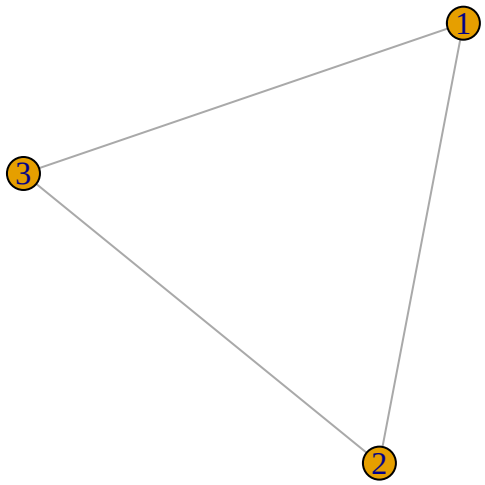
What does this output mean? The first word tells you that it is an igraph object. That is followed by four letters:

- D or U for a directed or undirected graph
- N for a named graph (when vertices have a name attribute)
- W for a weighted graph (when edges have a weight attribute)
- B for a bipartite graph

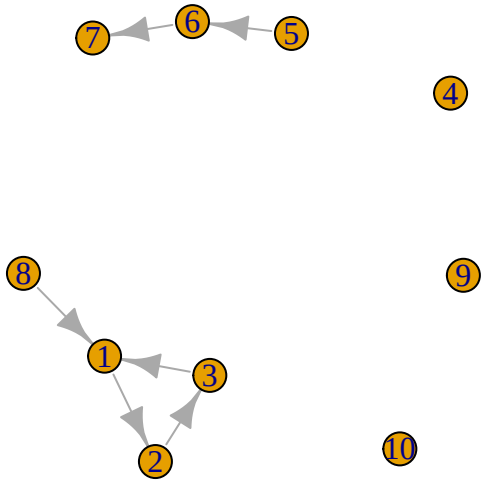
This is followed by the number of vertices and then the number of edges. Finally, it lists all the edges.

Now let's plot it!

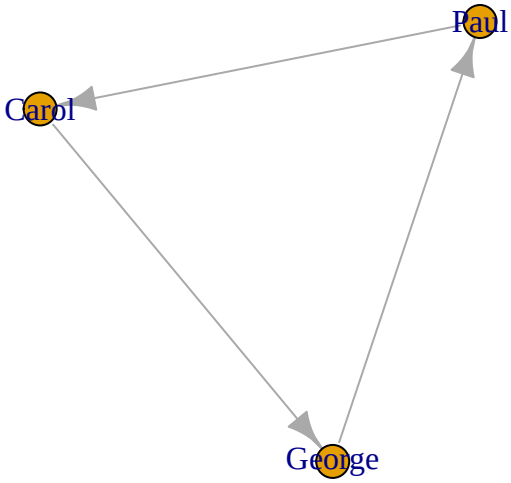
```
plot(g1)
```




```
# Now lets make a directed plot with 10 vertices. Directed is default
g2 <- graph(edges = c(1,2,2,3,3,1,5,6,6,7,8,1), n=10)
plot(g2)
```

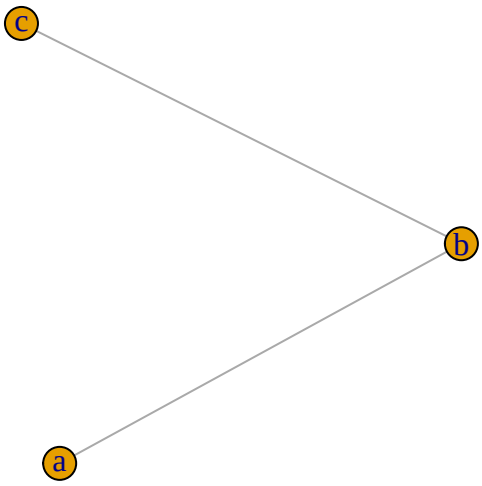



```
# Vertices can also be characters
g3 <- graph(edges = c("Carol", "George","George","Paul","Paul","Carol")) # don't need to identify numbe
plot(g3)
```

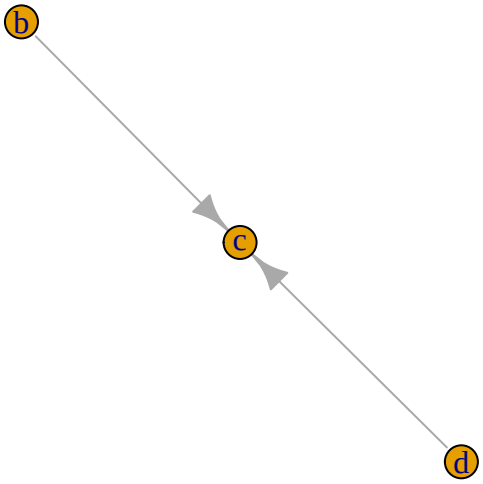



```
# You can also have multiple arrows between two nodes and isolates
g4 <- graph(edges = c("Carol", "George","George","Paul","Paul","Carol"), isolates = c("Jeremy", "Nick",
plot(g4)
```


```
# You can also make a network by indicating edges
plot(graph_from_literal(a--b,b---c)) # number of dashes doesn't matter
```



```
plot(graph_from_literal(b-+c,c+-d))
```



So now we know how to make simple igraph objects, let's access some pieces of the igraph object.

```
# Edges
E(g4)
```

```
## + 3/3 edges from c7bda4a (vertex names):
## [1] Carol ->George George->Paul   Paul  ->Carol
```

```
# Vertices
V(g4)
```

```
## + 6/6 vertices, named, from c7bda4a:
## [1] Carol  George Paul   Jeremy Nick   Sherry
```

```
# Network
g4[]
```

```
## 6 x 6 sparse Matrix of class "dgCMatrix"
##        Carol George Paul Jeremy Nick Sherry
## Carol      .      1    .      .    .      .
## George     .      .    1      .    .      .
## Paul       1      .    .      .    .      .
## Jeremy     .      .    .      .    .      .
## Nick       .      .    .      .    .      .
## Sherry     .      .    .      .    .      .
```

```
# You can access it in a similar way you would a matrix
g4[1,] # first row
```

```
##  Carol George   Paul Jeremy   Nick Sherry
##      0      1      0      0      0      0
```

```
g4[,1] # first column
```

```
##  Carol George   Paul Jeremy   Nick Sherry
##      0      0      1      0      0      0
```

```
# you can add name attributes to the vectors and edges
V(g4)$name # was already assigned when we made the object
```

```
## [1] "Carol"  "George" "Paul"   "Jeremy" "Nick"   "Sherry"
```

```
V(g4)$gender <- c("female", "male","male","male","male","female")
vertex_attr(g4) # view attributes
```

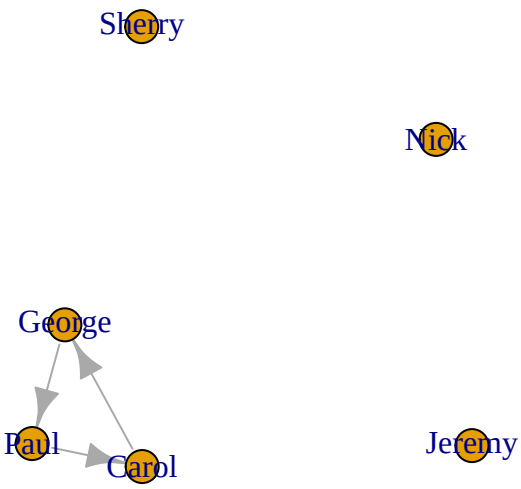
```
## $name
## [1] "Carol"  "George" "Paul"   "Jeremy" "Nick"   "Sherry"
##
## $gender
## [1] "female" "male"   "male"   "male"   "male"   "female"
```

```
E(g4)$type <- "relationships" # adds relationship to all edges
E(g4)$weight <- c(1,2,3) # adds a weight to each edge
edge_attr(g4) # view attributes
```

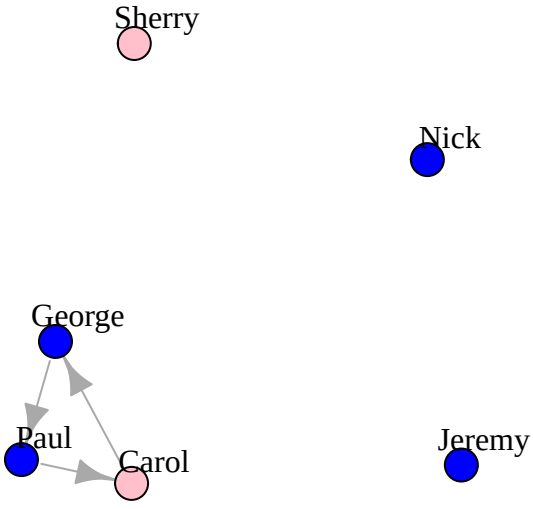
```
## $type
## [1] "relationships" "relationships" "relationships"
##
## $weight
## [1] 1 2 3
```

With the atributes added, let's plot again and see some of the alterations we can make to the plot

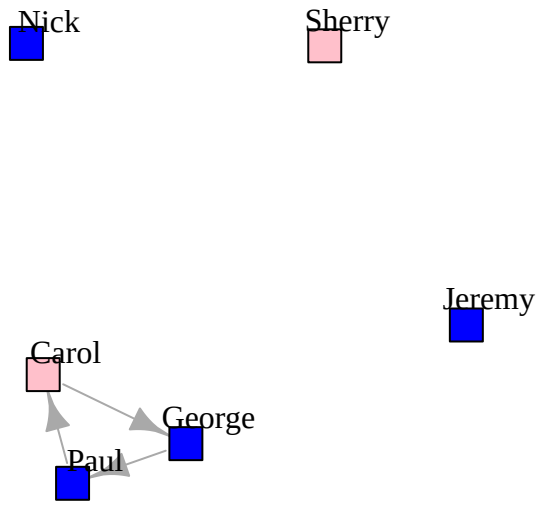
```
plot(g4)
```



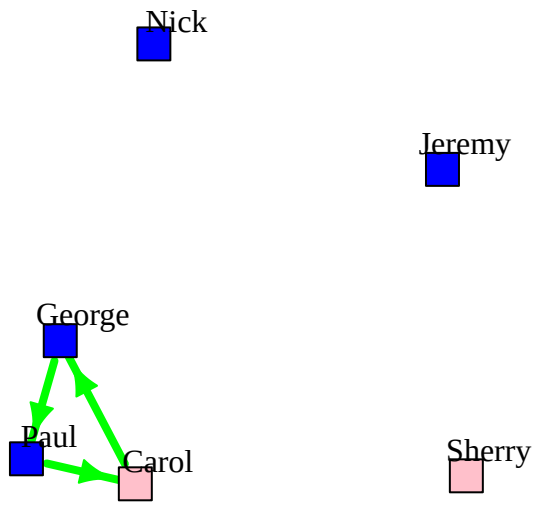

```
plot(g4, edge.arrow.size=1, vertex.label.color="black",vertex.label.dist=2,vertex.color=c("pink","blue"
```



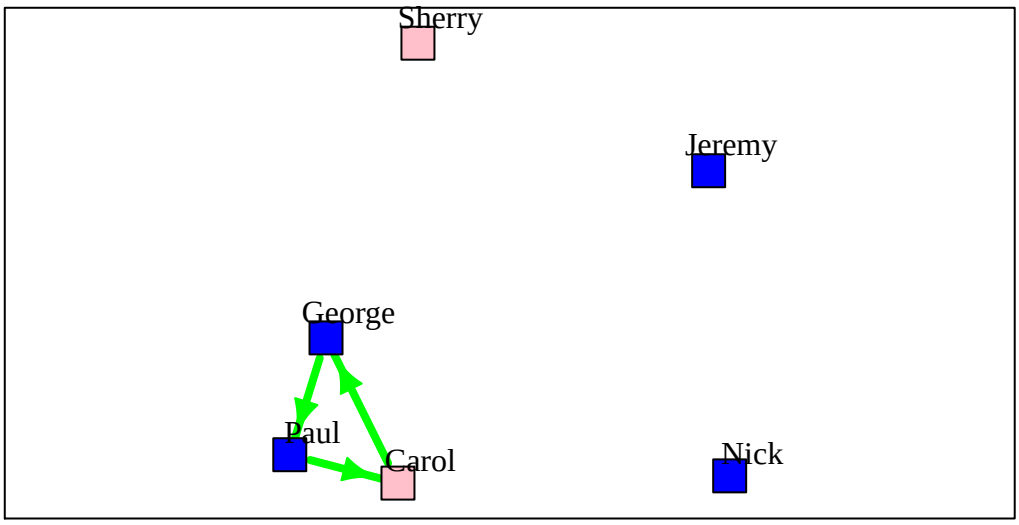
```
# change vertex shape
plot(g4, edge.arrow.size=01, vertex.label.color="black",vertex.label.dist=2,vertex.color=c("pink","blue
```




```
# change edges
plot(g4, edge.arrow.size=1, vertex.label.color="black",vertex.label.dist=2,vertex.color=c("pink","blue"
```

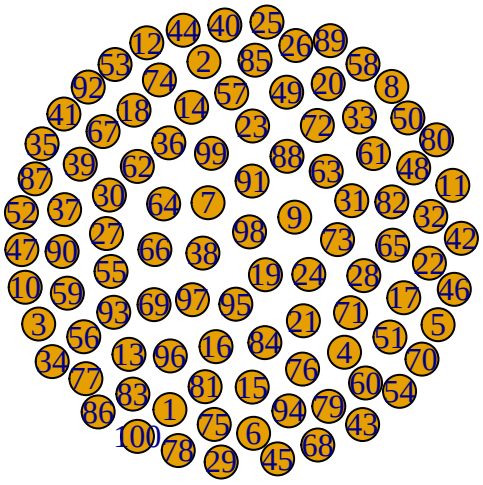



```
# I think this is fun
plot(g4, edge.arrow.size=1, vertex.label.color="black",vertex.label.dist=2,vertex.color=c("pink","blue"
```

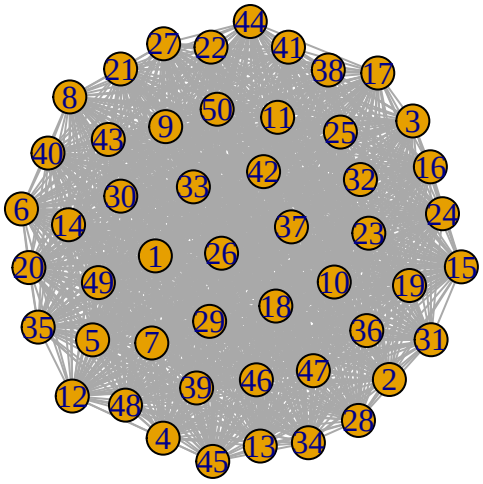


How do you build some larger graphs?

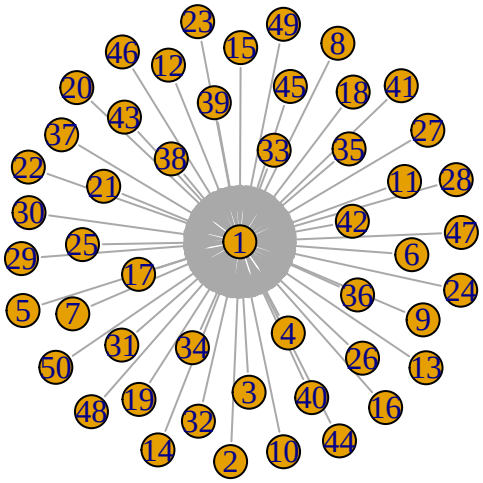
```
# make an empty graph
eg <- make_empty_graph(100)
plot(eg)
```



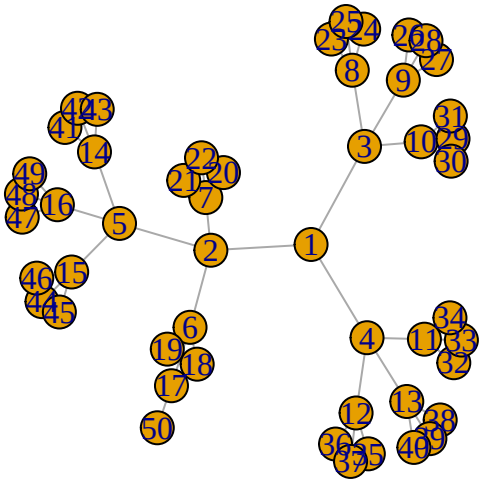
```
# full graph
fg <- make_full_graph(50)
plot(fg)
```



```
# some other formats for graphs
# star graph
st <- make_star(50)
plot(st)
```



```
# Tree graph
tr <- make_tree(50, children = 3, mode = "undirected")
plot(tr)
```



Let's use some real data!!

```
# options in igraphdata
data(package="igraphdata")
# load immuno dataset
data("immuno")
# What is this dataset?
class(immuno)
```

```
## [1] "igraph"
```

```
str(immuno)
```

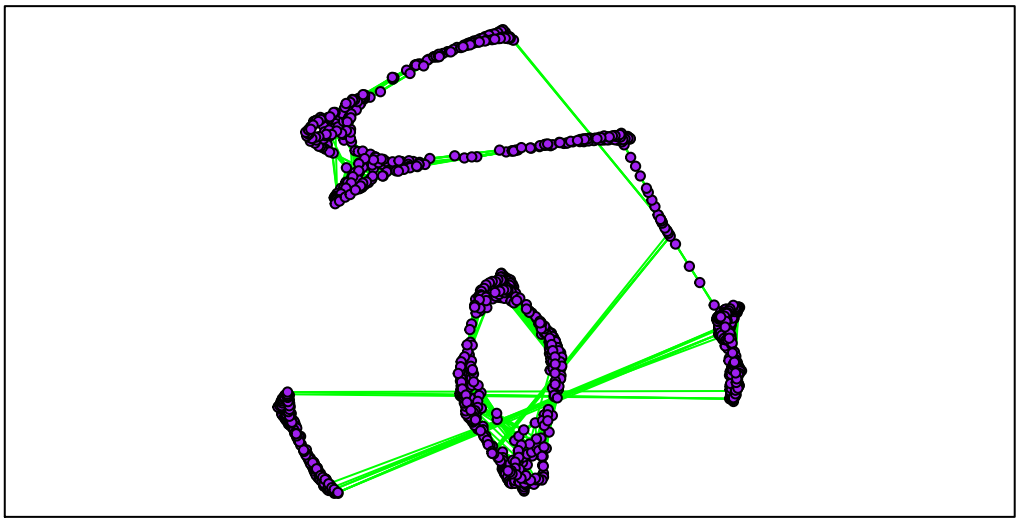
```
## List of 10
##  $ :List of 1
##   ..$ :Class 'igraph.vs'  atomic [1:5] 2 3 94 95 97
##   .. .. ..- attr(*, "env")=<weakref>
##   .. .. ..- attr(*, "graph")= chr "46d8c3fc-dbab-4aaf-a9e3-118630a7ccf1"
##  $ :List of 1
##   ..$ :Class 'igraph.vs'  atomic [1:7] 1 3 4 25 26 27 97
##   .. .. ..- attr(*, "env")=<weakref>
##   .. .. ..- attr(*, "graph")= chr "46d8c3fc-dbab-4aaf-a9e3-118630a7ccf1"
##  $ :List of 1
##   ..$ :Class 'igraph.vs'  atomic [1:9] 1 2 4 5 25 26 97 98 99
##   .. .. ..- attr(*, "env")=<weakref>
##   .. .. ..- attr(*, "graph")= chr "46d8c3fc-dbab-4aaf-a9e3-118630a7ccf1"
##  $ :List of 1
##   ..$ :Class 'igraph.vs'  atomic [1:15] 2 3 5 6 23 24 25 26 88 89 ...
##   .. .. ..- attr(*, "env")=<weakref>
##   .. .. ..- attr(*, "graph")= chr "46d8c3fc-dbab-4aaf-a9e3-118630a7ccf1"
##  $ :List of 1
##   ..$ :Class 'igraph.vs'  atomic [1:9] 3 4 6 7 23 24 25 99 100
##   .. .. ..- attr(*, "env")=<weakref>
##   .. .. ..- attr(*, "graph")= chr "46d8c3fc-dbab-4aaf-a9e3-118630a7ccf1"
##  $ :List of 1
##   ..$ :Class 'igraph.vs'  atomic [1:15] 4 5 7 8 9 21 22 23 24 87 ...
##   .. .. ..- attr(*, "env")=<weakref>
##   .. .. ..- attr(*, "graph")= chr "46d8c3fc-dbab-4aaf-a9e3-118630a7ccf1"
##  $ :List of 1
##   ..$ :Class 'igraph.vs'  atomic [1:9] 5 6 8 9 21 22 23 101 102
```

```
##   .. .. ..- attr(*, "env")=<weakref>
##   .. .. ..- attr(*, "graph")= chr "46d8c3fc-dbab-4aaf-a9e3-118630a7ccf1"
##  $ :List of 1
##   ..$ :Class 'igraph.vs'  atomic [1:8] 6 7 9 10 11 21 22 102
##   .. .. ..- attr(*, "env")=<weakref>
##   .. .. ..- attr(*, "graph")= chr "46d8c3fc-dbab-4aaf-a9e3-118630a7ccf1"
##  $ :List of 1
##   ..$ :Class 'igraph.vs'  atomic [1:8] 6 7 8 10 11 101 102 103
##   .. .. ..- attr(*, "env")=<weakref>
##   .. .. ..- attr(*, "graph")= chr "46d8c3fc-dbab-4aaf-a9e3-118630a7ccf1"
##  $ :List of 1
##   ..$ :Class 'igraph.vs'  atomic [1:8] 8 9 11 12 102 103 104 105
##   .. .. ..- attr(*, "env")=<weakref>
##   .. .. ..- attr(*, "graph")= chr "46d8c3fc-dbab-4aaf-a9e3-118630a7ccf1"
##  - attr(*, "class")= chr "igraph"
```

```
immuno
```

```
## IGRAPH 46d8c3f U--- 1316 6300 -- immunoglobuline network
## + attr: name (g/c), Citation (g/c), Author (g/c)
## + edges from 46d8c3f:
##  [1]  1--  2  1--  3  1-- 94  1-- 95  1-- 97  2--  3  2--  4  2-- 25
##  [9]  2-- 26  2-- 27  2-- 97  3--  4  3--  5  3-- 25  3-- 26  3-- 97
## [17]  3-- 98  3-- 99  4--  5  4--  6  4-- 23  4-- 24  4-- 25  4-- 26
## [25]  4-- 88  4-- 89  4-- 90  4-- 97  4-- 98  4-- 99  4--100  5--  6
## [33]  5--  7  5-- 23  5-- 24  5-- 25  5-- 99  5--100  6--  7  6--  8
## [41]  6--  9  6-- 21  6-- 22  6-- 23  6-- 24  6-- 87  6-- 88  6-- 99
## [49]  6--100  6--101  6--102  7--  8  7--  9  7-- 21  7-- 22  7-- 23
## [57]  7--101  7--102  8--  9  8-- 10  8-- 11  8-- 21  8-- 22  8--102
## + ... omitted several edges
```

```
# what does it look like?
plot(immuno, vertex.label=NA, vertex.size=4,vertex.color="purple", edge.color="green", frame=TRUE)
```



One powerful thing that igraph does is calculate different forms of centrality for you. This can help you identify key players within the network. Let's step through a few.

Degree: the number of ties for a vertex.

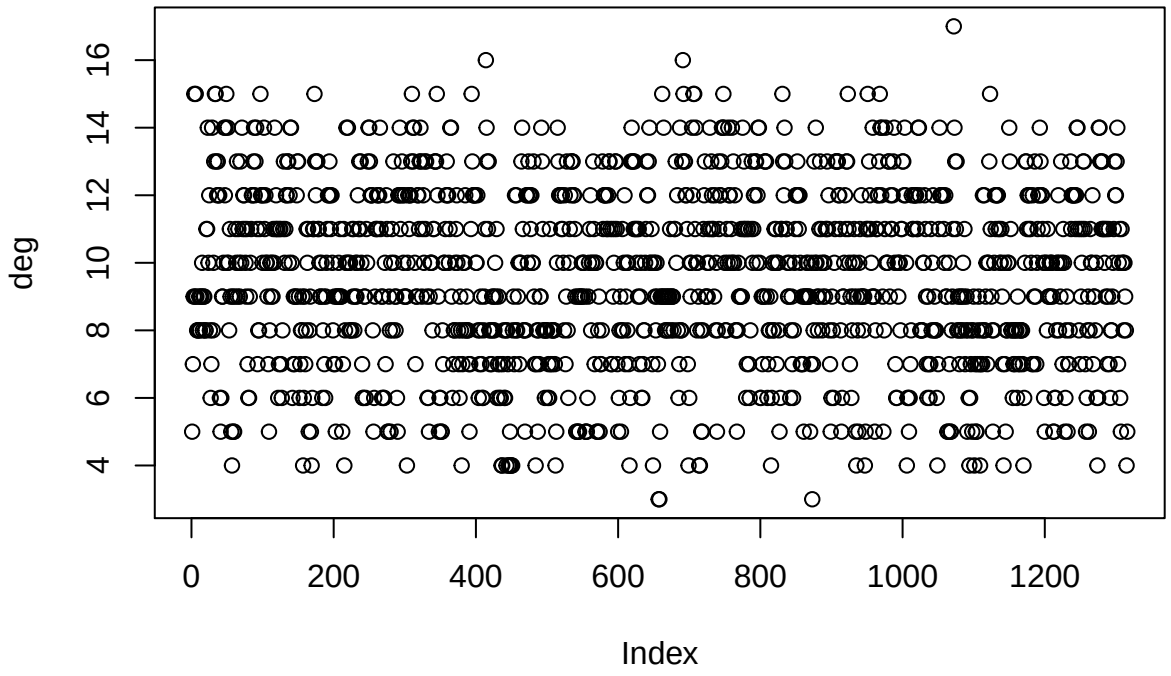
Arguments include graph, the igraph object, v, the ids of the vertices that will be analyzed, mode, out-

degree/in-degree/total-degree for the sum (undirected graphs are defaulted to “all”).

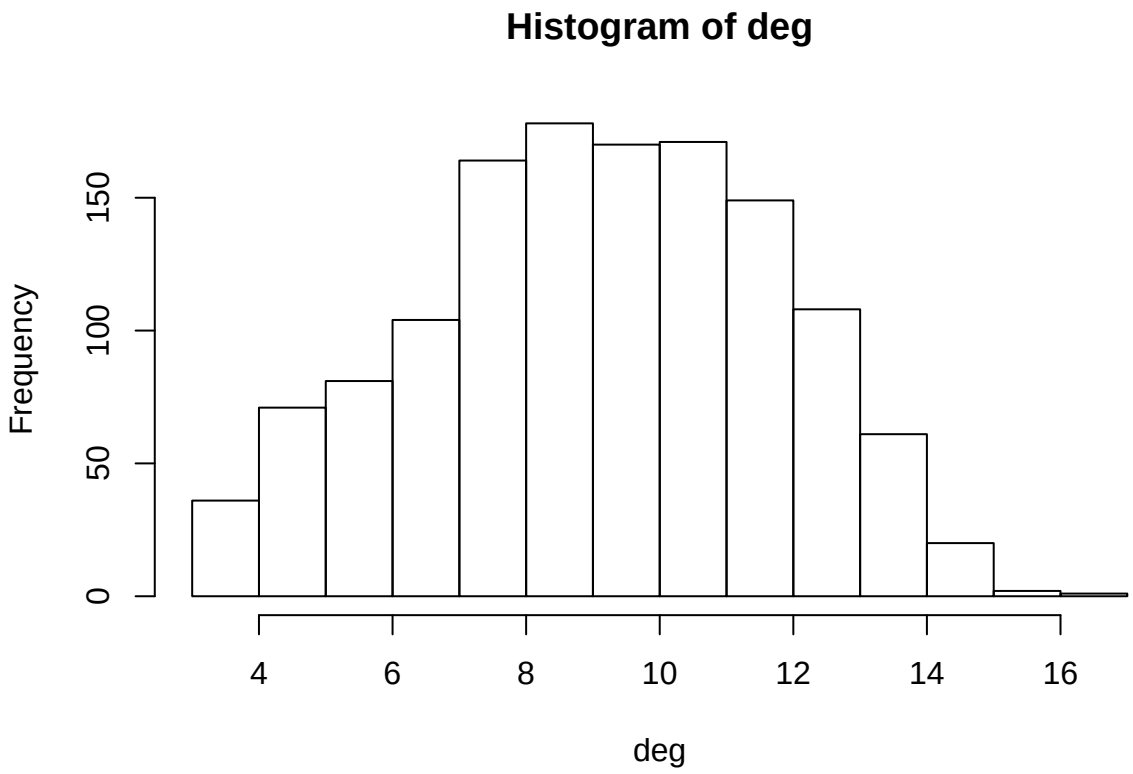
```
# Degree
deg <- degree(immuno, mode="all")
str(deg)
```

```
##  num [1:1316] 5 7 9 15 9 15 9 8 8 8 ...
```

```
plot(deg)
```



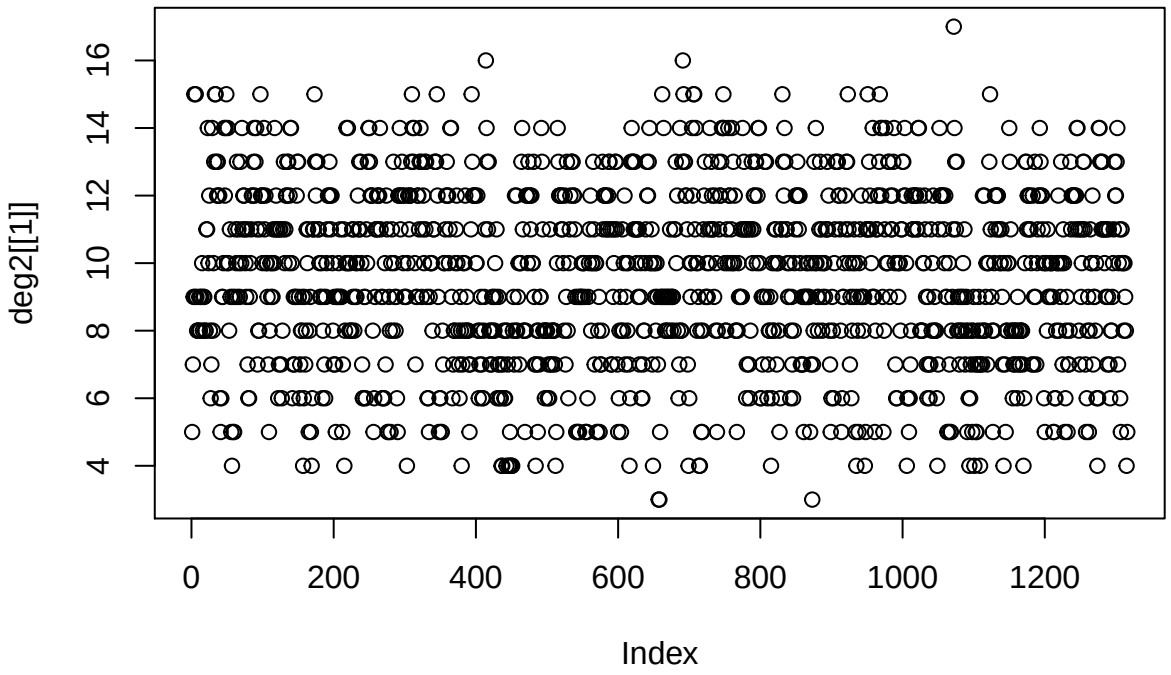

```
hist(deg)
```



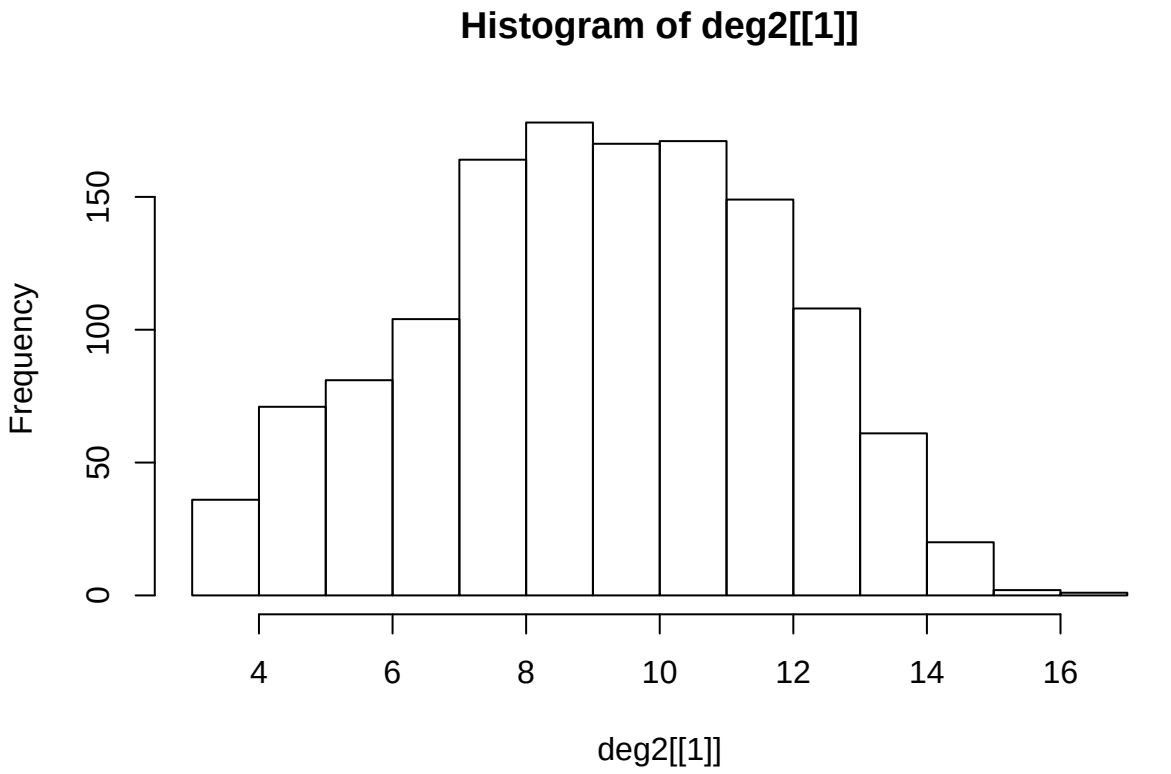

```
# another function for degree
# res: degree value for each vertex
# centralization: the graph level centrality index
# theoretical_max: max theoretical centralization score for a graph of this size
deg2 <- centr_degree(immuno, mode = "all", normalized = T) # normalized if true divides the graph level
str(deg2)
```

```
## List of 3
##  $ res            : num [1:1316] 5 7 9 15 9 15 9 8 8 8 ...
##  $ centralization : num 0.00565
##  $ theoretical_max: num 1730540
```

```
plot(deg2[[1]])
```




```
hist(deg2[[1]])
```



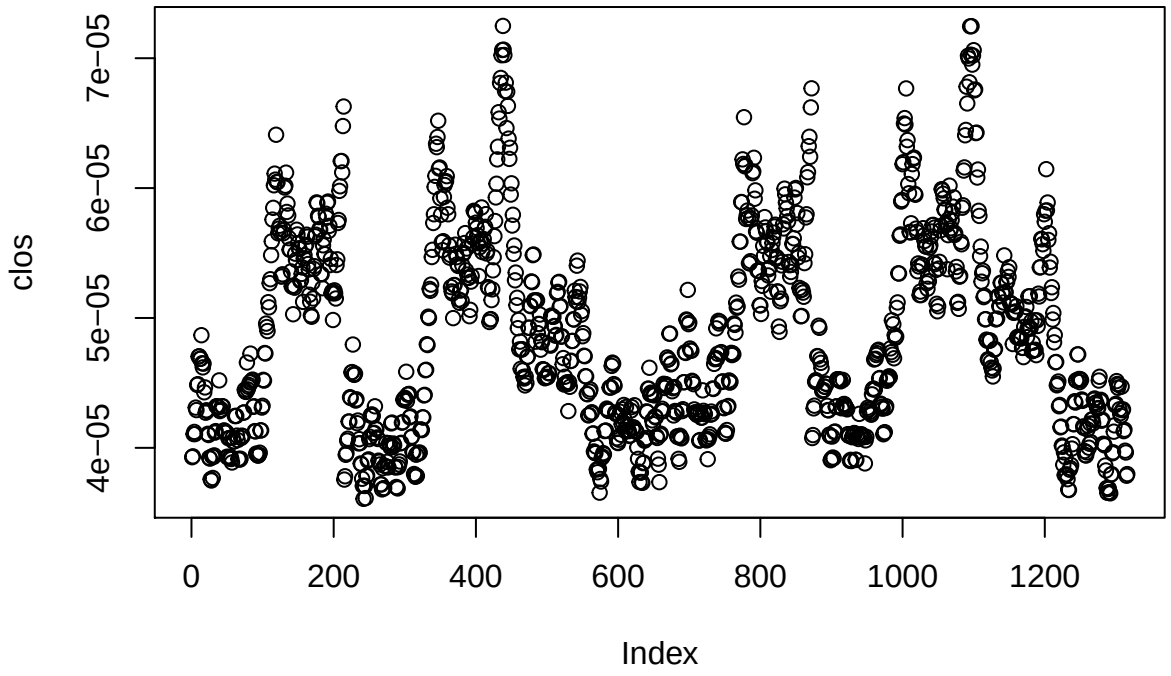
Closeness: centrality based on distance to others in the graph

Arguments now also include weights. In a weighted graph they are automatically included (immuno is not).

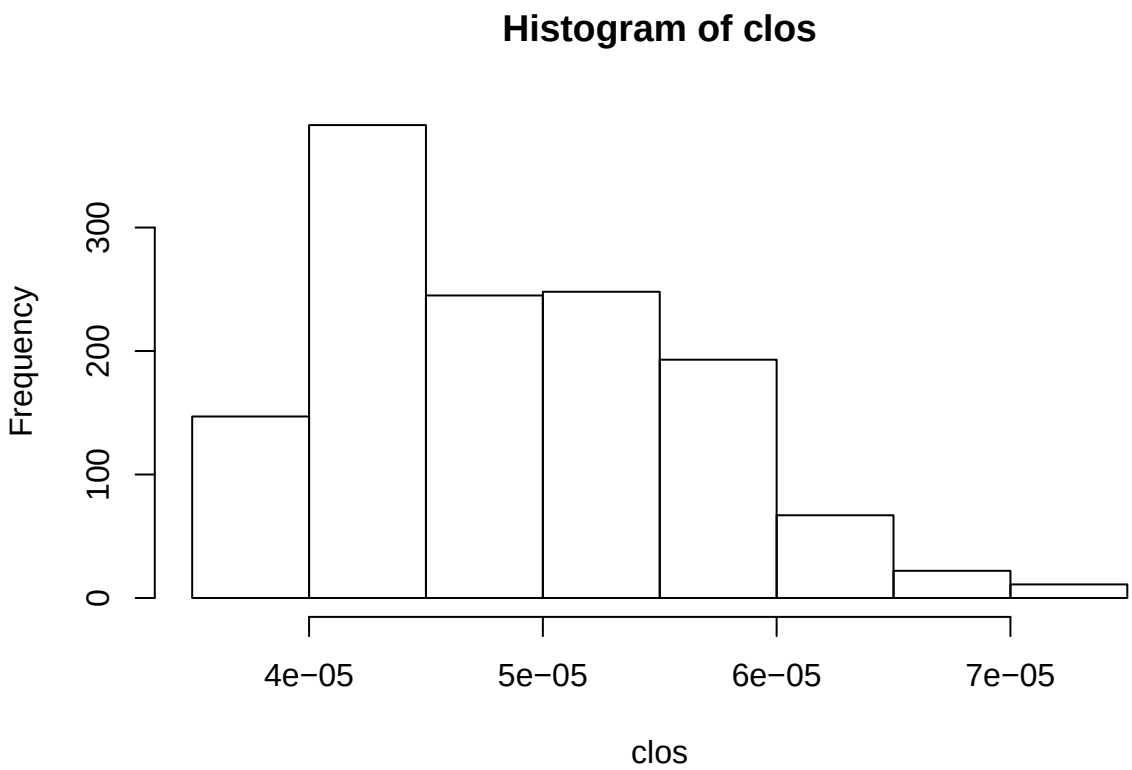
```
# closeness
clos <- closeness(immuno, mode = "all", weights = NA)
str(clos)
```

```
##  num [1:1316] 3.93e-05 3.93e-05 4.11e-05 4.12e-05 4.11e-05 ...
```

```
plot(clos)
```



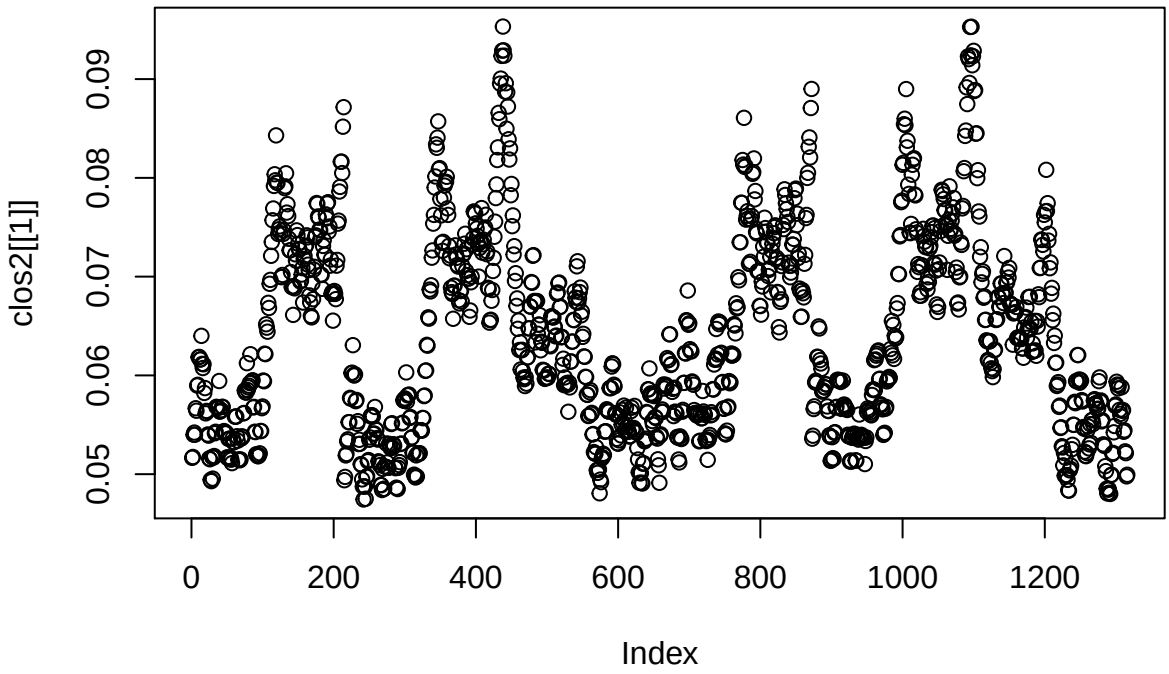

```
hist(clos)
```



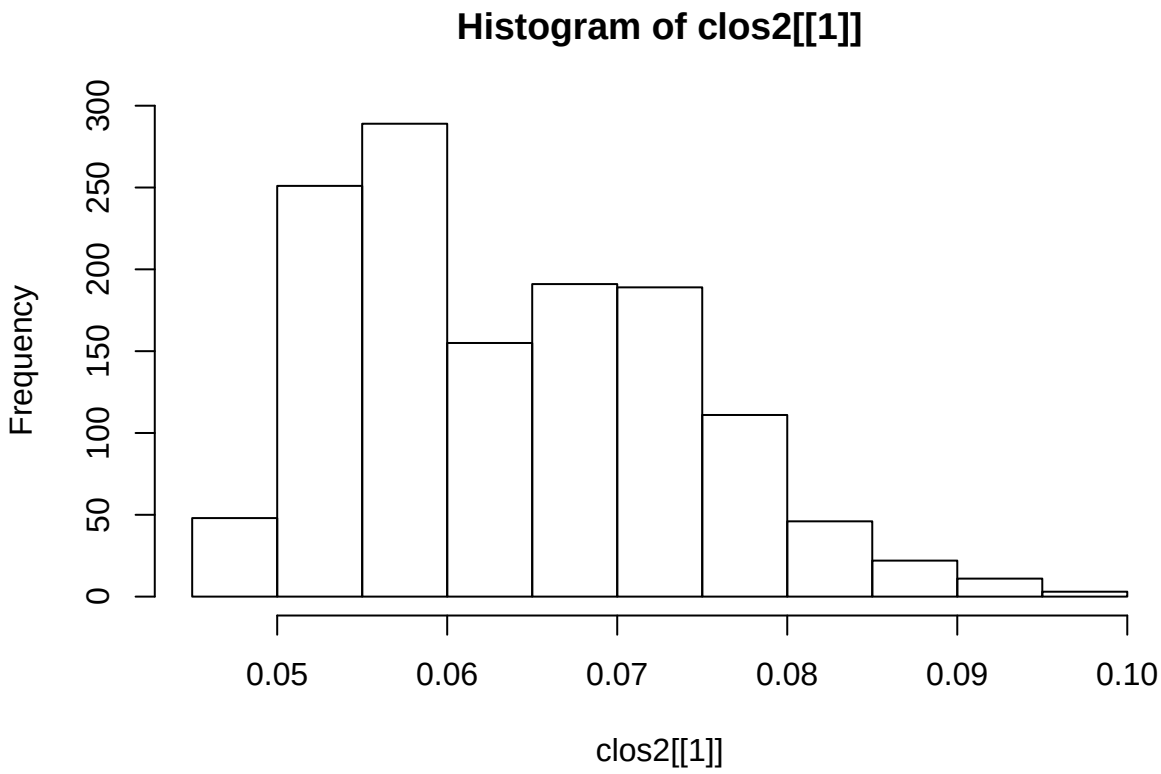

```
# another function for closeness
clos2 <- centr_clo(immuno, normalized=T)
str(clos2)
```

```
## List of 3
##  $ res            : num [1:1316] 0.0517 0.0517 0.054 0.0542 0.054 ...
##  $ centralization : num 0.0629
##  $ theoretical_max: num 657
```

```
plot(clos2[[1]])
```




```
hist(clos2[[1]])
```

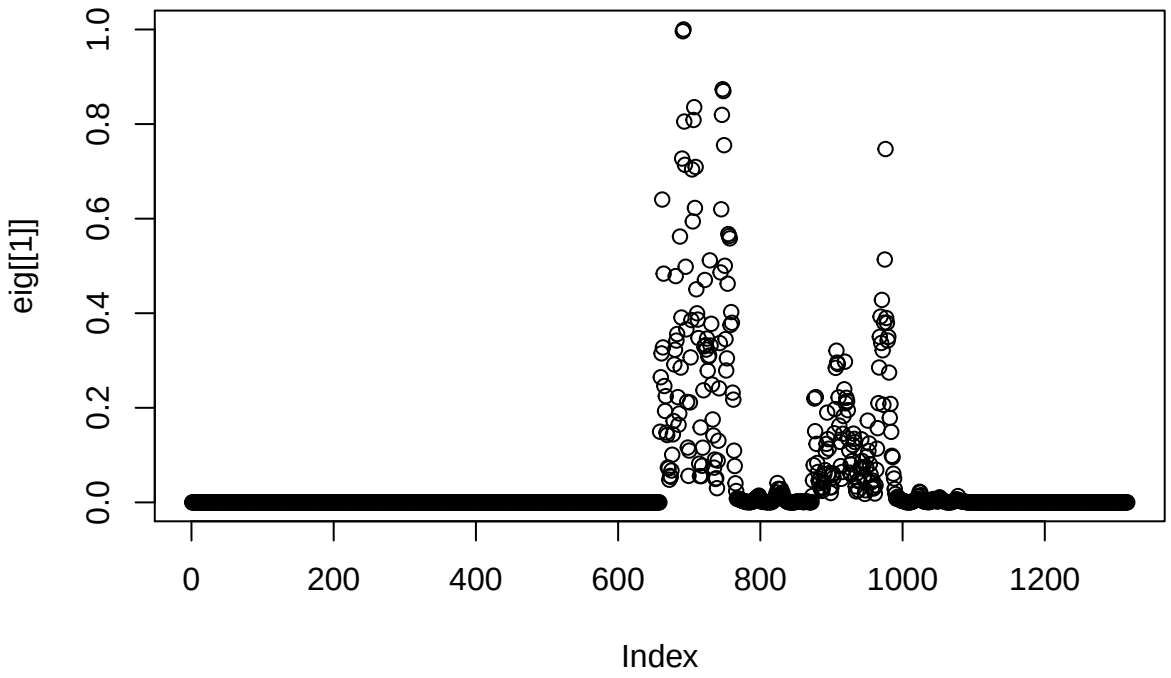


Eigenvectors: centrality proportional to the sum of connection centralities. In a little easier words, it is the centrality of a vertex proportional to the sum of the centrality values of those it is connected to.

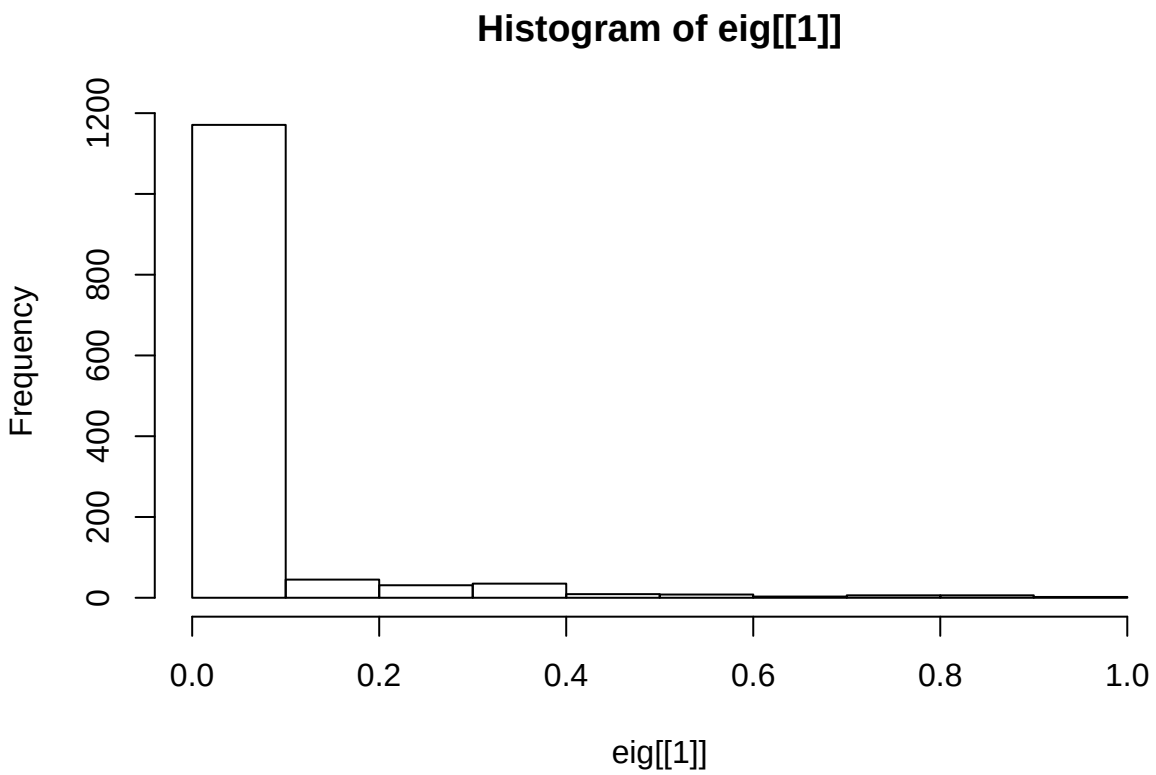
```
# Eigenvectors
eig <- eigen_centrality(immuno, directed=F, weights=NA)
str(eig) # eigenvalue per vertex, eigenvalue of eigenvector, options
```

```
## List of 3
##  $ vector : num [1:1316] 4.06e-12 6.38e-12 9.64e-12 1.95e-11 1.12e-11 ...
##  $ value  : num 11.6
##  $ options:List of 20
##   ..$ bmat   : chr "I"
##   ..$ n      : int 1316
##   ..$ which  : chr "LA"
##   ..$ nev    : int 1
##   ..$ tol    : num 0
##   ..$ ncv    : int 0
##   ..$ ldv    : int 0
##   ..$ ishift : int 1
##   ..$ maxiter: int 1000
##   ..$ nb     : int 1
##   ..$ mode   : int 1
##   ..$ start  : int 1
##   ..$ sigma  : num 0
##   ..$ sigmai : num 0
##   ..$ info   : int 0
##   ..$ iter   : int 8
##   ..$ nconv  : int 1
##   ..$ numop  : int 90
##   ..$ numopb : int 0
##   ..$ numreo : int 81
```

```
plot(eig[[1]])
```



```r
hist(eig[[1]])
```

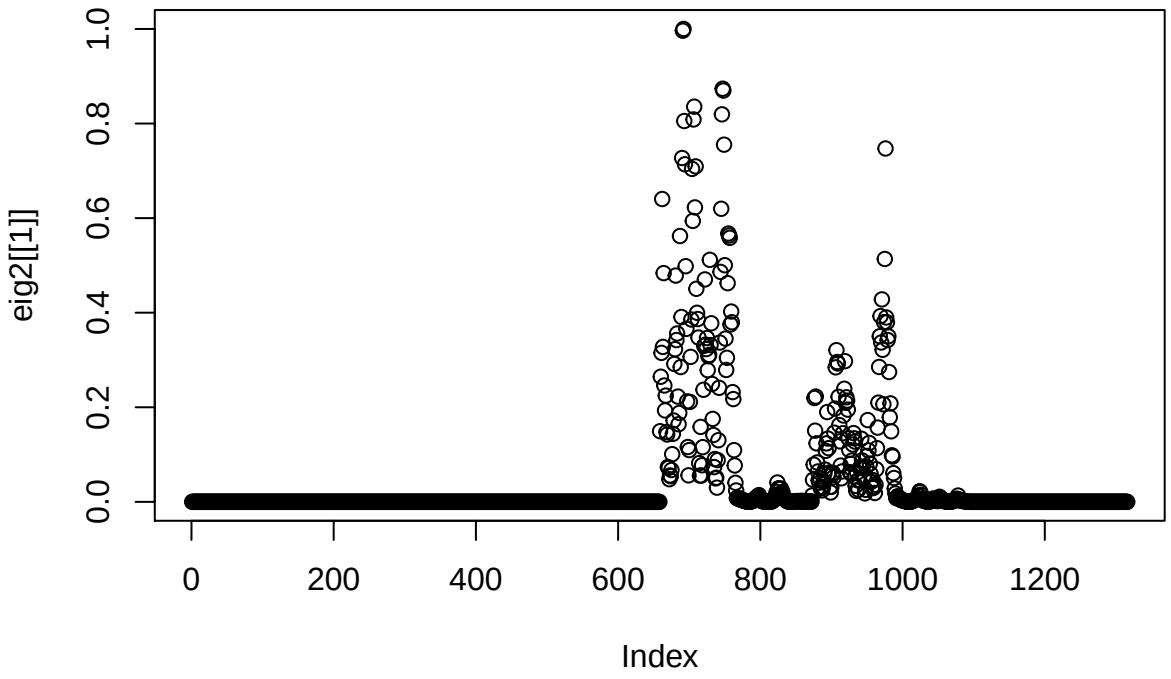



```r
# another function for eigenvectors
eig2 <- centr_eigen(immuno, directed=T, normalized=T)
str(eig2)
```

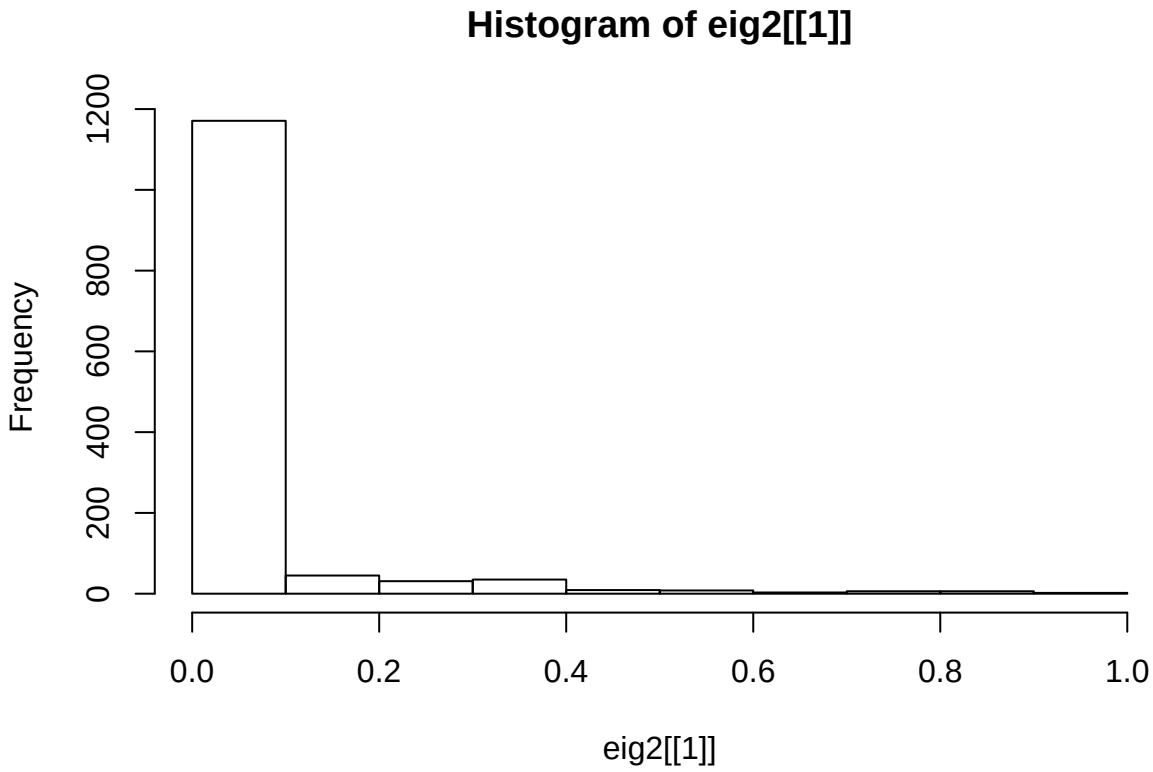
```
## List of 5
##  $ vector         : num [1:1316] 4.08e-12 6.41e-12 9.68e-12 1.96e-11 1.13e-11 ...
##  $ value          : num 11.6
##  $ options        :List of 20
##   ..$ bmat   : chr "I"
##   ..$ n      : int 1316
##   ..$ which  : chr "LA"
##   ..$ nev    : int 1
##   ..$ tol    : num 0
##   ..$ ncv    : int 0
##   ..$ ldv    : int 0
##   ..$ ishift : int 1
##   ..$ maxiter: int 1000
##   ..$ nb     : int 1
##   ..$ mode   : int 1
##   ..$ start  : int 1
##   ..$ sigma  : num 0
##   ..$ sigmai : num 0
##   ..$ info   : int 0
##   ..$ iter   : int 8
##   ..$ nconv  : int 1
##   ..$ numop  : int 90
##   ..$ numopb : int 0
##   ..$ numreo : int 81
```

```
##  $ centralization : num 0.961
##  $ theoretical_max: num 1314
```

```
plot(eig2[[1]])
```




```
hist(eig2[[1]])
```



Betweenness: Centrality based on broker position

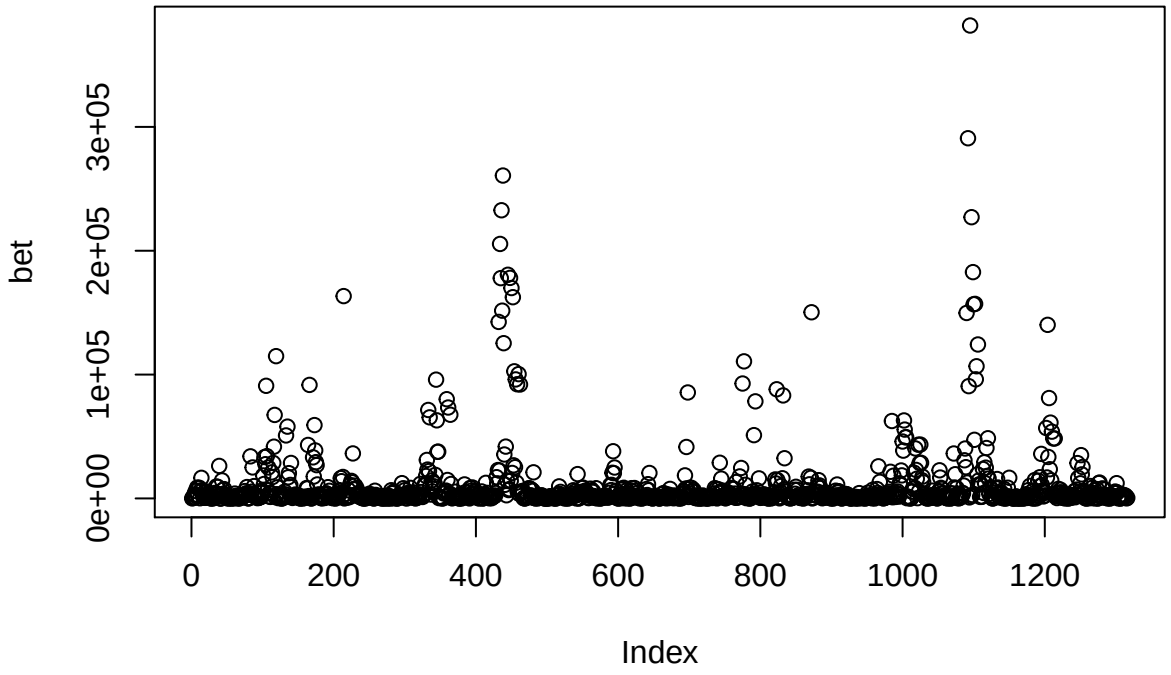
A broker position is a position that is highly connected to groups that are not necessarily connected to each

other otherwise.

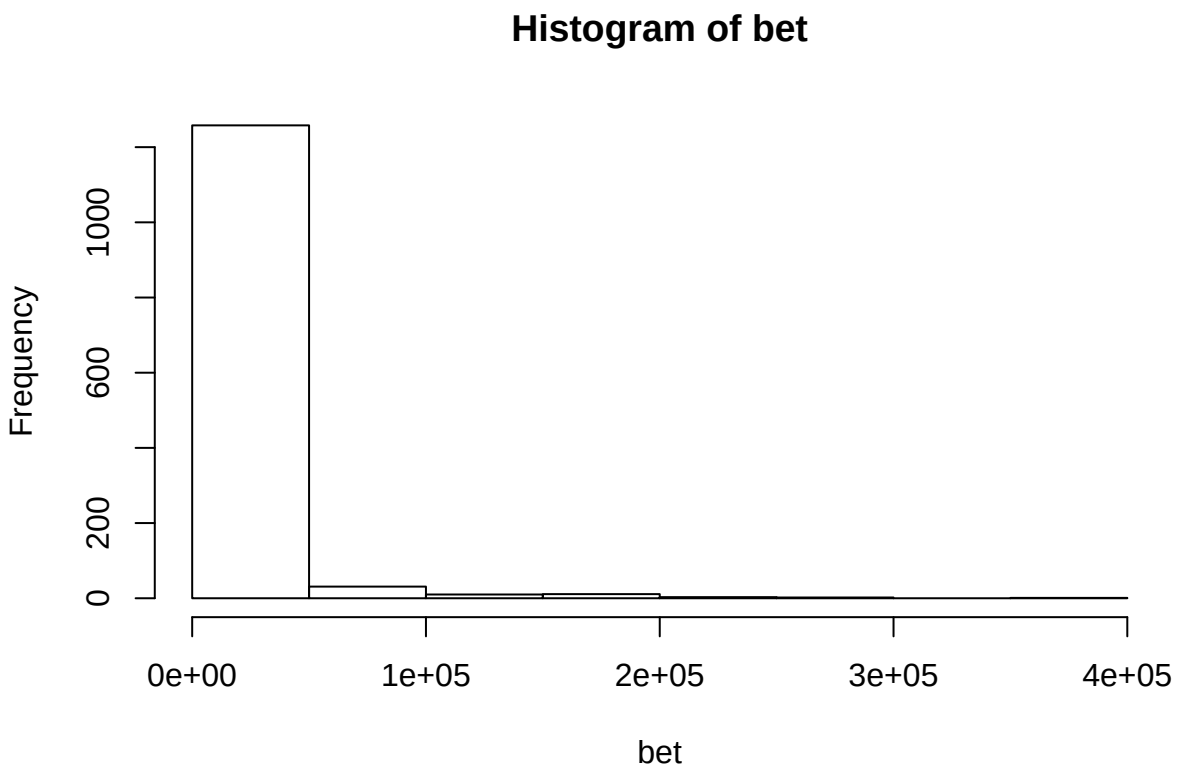
```
# Betweenness
bet <- betweenness(immuno, directed=F, weights=NA)
str(bet)
```

```
##  num [1:1316] 42.5 309.8 793.4 3050.7 662.2 ...
```

```
plot(bet)
```



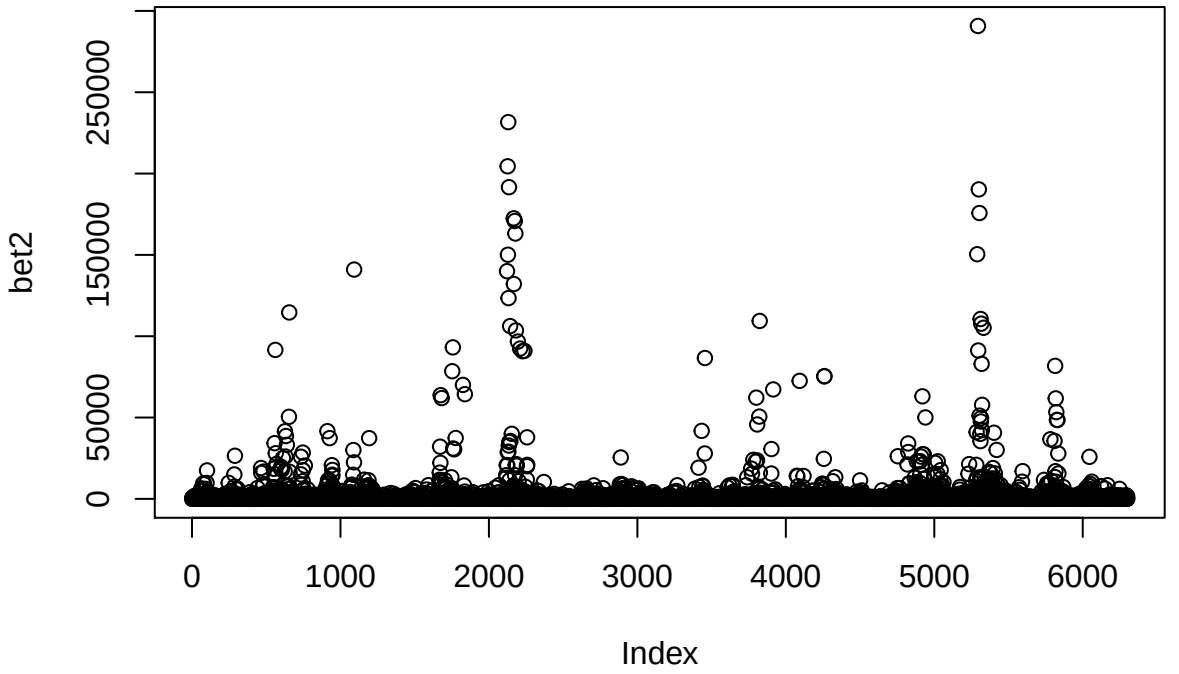

```
hist(bet)
```



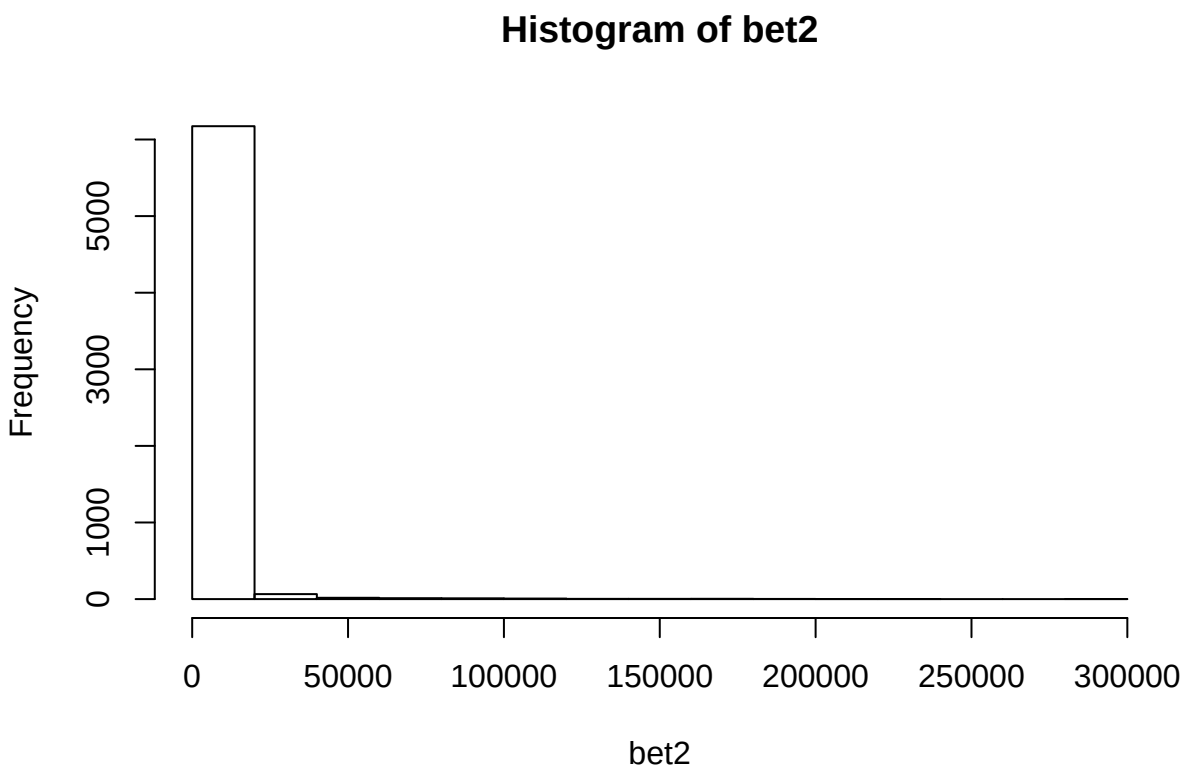

```
# you can also calculate this for edges
bet2 <- edge_betweenness(immuno, directed=F, weights=NA)
str(bet2)
```

```
##  num [1:6300] 35.5 576.4 33 90.4 664.7 ...
```

```
plot(bet2)
```



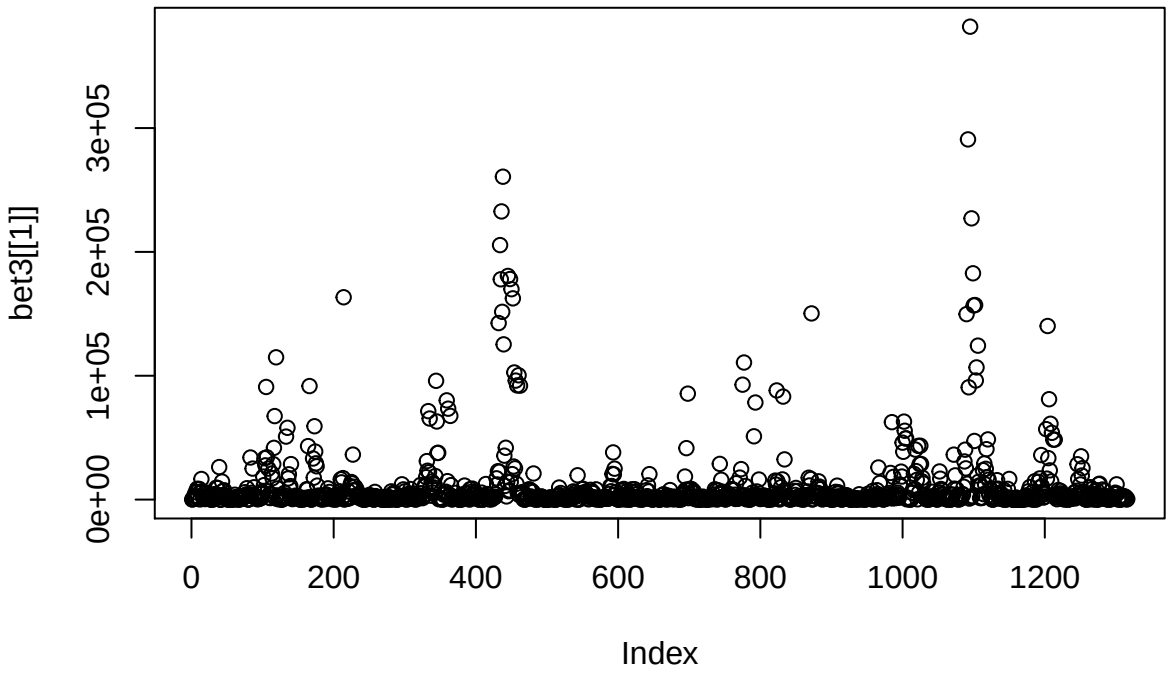
```r
hist(bet2)
```



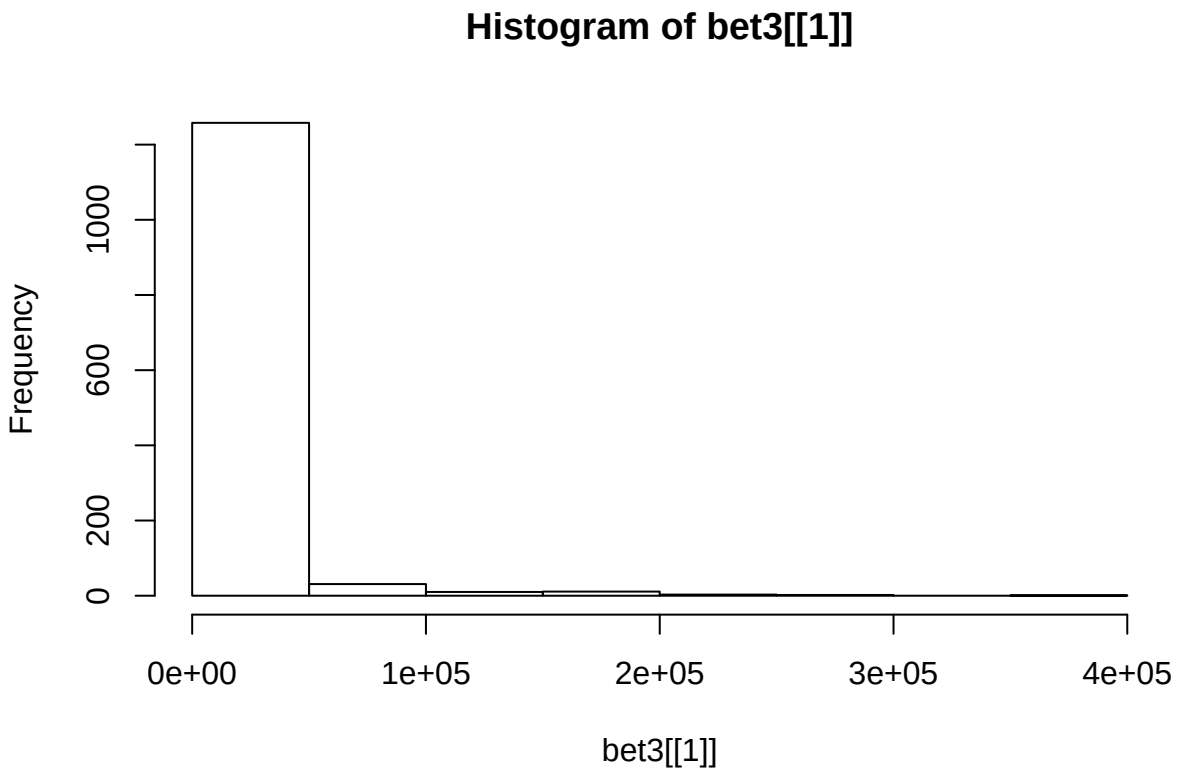

```r
# another function for betweenness
bet3 <- centr_betw(immuno, directed=F, normalized=T)
str(bet3)
```

```
## List of 3
##  $ res            : num [1:1316] 42.5 309.8 793.4 3050.7 662.2 ...
##  $ centralization : num 0.431
##  $ theoretical_max: num 1.14e+09
```

```r
plot(bet3[[1]])
```




```
hist(bet3[[1]])
```



igraph can also identify hub and authority genes.

Hubs are vertices with many outgoing links while authorities are vertices with many incoming links.

```
# hubs
hs <- hub_score(immuno, weights=NA)
str(hs)
```

```
## List of 3
##  $ vector : num [1:1316] 4.10e-12 6.44e-12 9.73e-12 1.96e-11 1.13e-11 ...
##  $ value  : num 134
##  $ options:List of 20
##   ..$ bmat   : chr "I"
##   ..$ n      : int 1316
##   ..$ which  : chr "LM"
##   ..$ nev    : int 1
##   ..$ tol    : num 0
##   ..$ ncv    : int 0
##   ..$ ldv    : int 0
##   ..$ ishift : int 1
##   ..$ maxiter: int 1000
##   ..$ nb     : int 1
##   ..$ mode   : int 1
##   ..$ start  : int 1
##   ..$ sigma  : num 0
##   ..$ sigmai : num 0
##   ..$ info   : int 0
##   ..$ iter   : int 4
##   ..$ nconv  : int 1
##   ..$ numop  : int 50
##   ..$ numopb : int 0
##   ..$ numreo : int 50
```

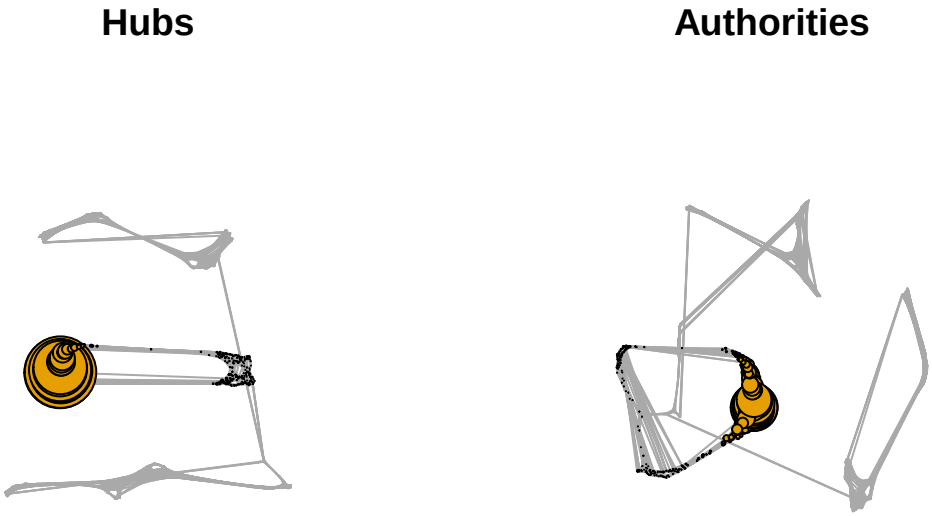
```
# get just the hub_scores
hs <- hub_score(immuno)$vector
head(hs)
```

```
## [1] 4.099616e-12 6.438937e-12 9.729339e-12 1.964246e-11 1.130695e-11
## [6] 1.907280e-11
```

```
# authority
as <- authority_score(immuno, weights=NA)$vector
head(as)
```

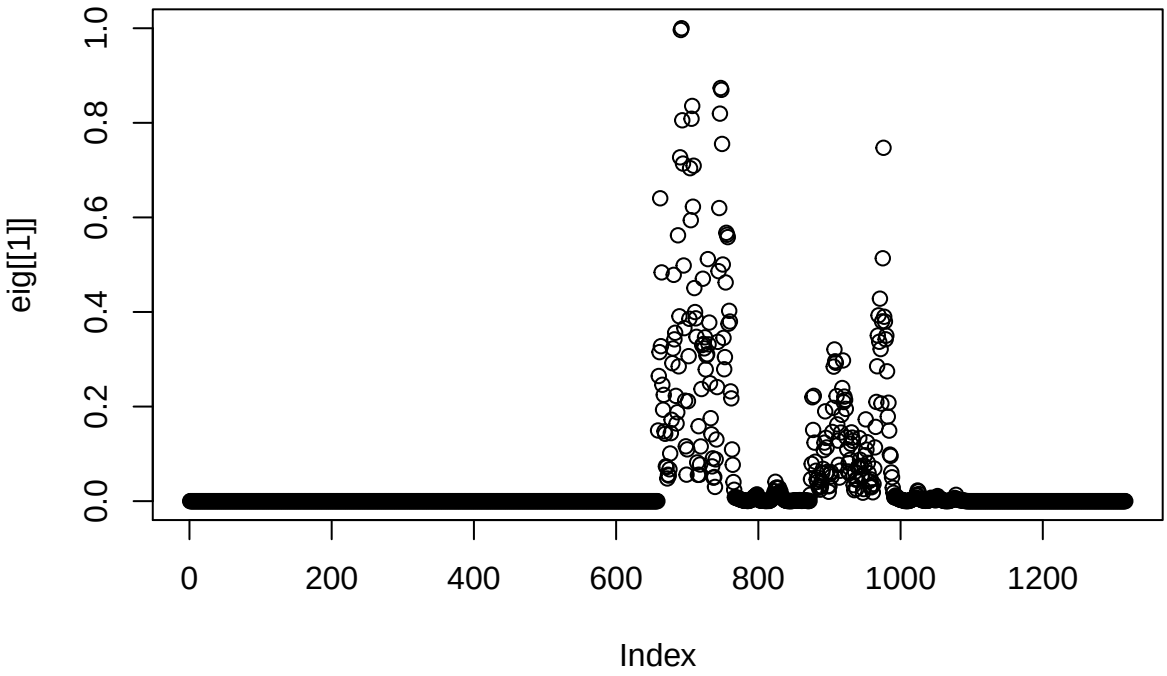
```
## [1] 4.099616e-12 6.438937e-12 9.729339e-12 1.964246e-11 1.130695e-11
## [6] 1.907280e-11
```

```
# plot these two
par(mfrow=c(1,2))
 plot(immuno, vertex.size=hs*50, main="Hubs", vertex.label=NA)
 plot(immuno, vertex.size=as*30, main="Authorities",vertex.label=NA)
```



So what do you do with these values? Well, what if you want to know which vertices have the high centrality scores? Let's work with the eigenvectors

```
# bring up the plot again
plot(eig[[1]])
```




```
# use a cut off of 0.2
sigeigindex <- which(eig[[1]] >= 0.2)
sigeigvalue <- eig[[1]][eig[[1]] >= 0.2]
df <- cbind(sigeigindex, sigeigvalue)
head(df)
```

```
##      sigeigindex sigeigvalue
## [1,]         660   0.2646300
## [2,]         661   0.3151167
## [3,]         662   0.6404080
## [4,]         663   0.3276921
```

```
## [5,]         664   0.4837146
## [6,]         665   0.2460551
```

```r
class(df)
```

```
## [1] "matrix"
```