Grey’s Anatomy

HilaReut

16 May 2016

```
library(igraph)
```

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

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

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

```
require(igraph)
```

Grey’s anatomy data

```
edges.data = read.csv('ga_edgelist.csv',header = T)
ga_edges =  graph.data.frame(edges.data,directed = F)
summary(ga_edges)
```

```
## IGRAPH UN-- 32 34 --
## + attr: name (v/c)
```

```
V(ga_edges)$name
```

```
##  [1] "lexi"          "owen"          "sloan"         "torres"
##  [5] "derek"         "karev"         "o'malley"      "yang"
##  [9] "grey"          "chief"         "ellis grey"    "susan grey"
## [13] "bailey"        "izzie"         "altman"        "arizona"
## [17] "colin"         "preston"       "kepner"        "addison"
## [21] "nancy"         "olivia"        "mrs. seabury"  "adele"
## [25] "thatch grey"   "tucker"        "hank"          "denny"
## [29] "finn"          "steve"         "ben"           "avery"
```

```
#Remove self-Loops is exist
ga_edges = simplify(ga_edges)
```

Calculate Betweenness

```
#Calculate betweenness
ga_bet = betweenness(ga_edges)
ga_bet = sort(ga_bet,decreasing = T)
names(ga_bet[1])
```

```
## [1] "sloan"
```

Calculate closeness

```
#Calculate closeness
ga_close = closeness(ga_edges)
ga_close = sort(ga_close, decreasing = T)
names(ga_close[1])
```

```
## [1] "torres"
```

Calculate eigenvector

```
#Calculate eigenvector
ga_eigen = evcent(ga_edges)
ga_eigen = sort(ga_eigen$vector, decreasing = T)
names(ga_eigen[1])
```

```
## [1] "karev"
```

Find commuinty with Girvan-Newman community detection.

```
fc = edge.betweenness.community(ga_edges)
```

Cheack what is the modularity

```
#Cheack what is the modularity
fc$modularity
```

```
##  [1] -0.04584775 -0.01816609  0.01038062  0.03892734  0.06747405
##  [6]  0.09515571  0.12326990  0.14922145  0.17560554  0.20328720
## [11]  0.23053633  0.25821799  0.28633218  0.31358131  0.34169550
## [16]  0.36851211  0.39576125  0.42344291  0.44247405  0.46712803
## [21]  0.49134948  0.50778547  0.52681661  0.54974048  0.57050173
## [26]  0.57742215  0.56098616  0.53416955  0.45804498  0.30449827
```

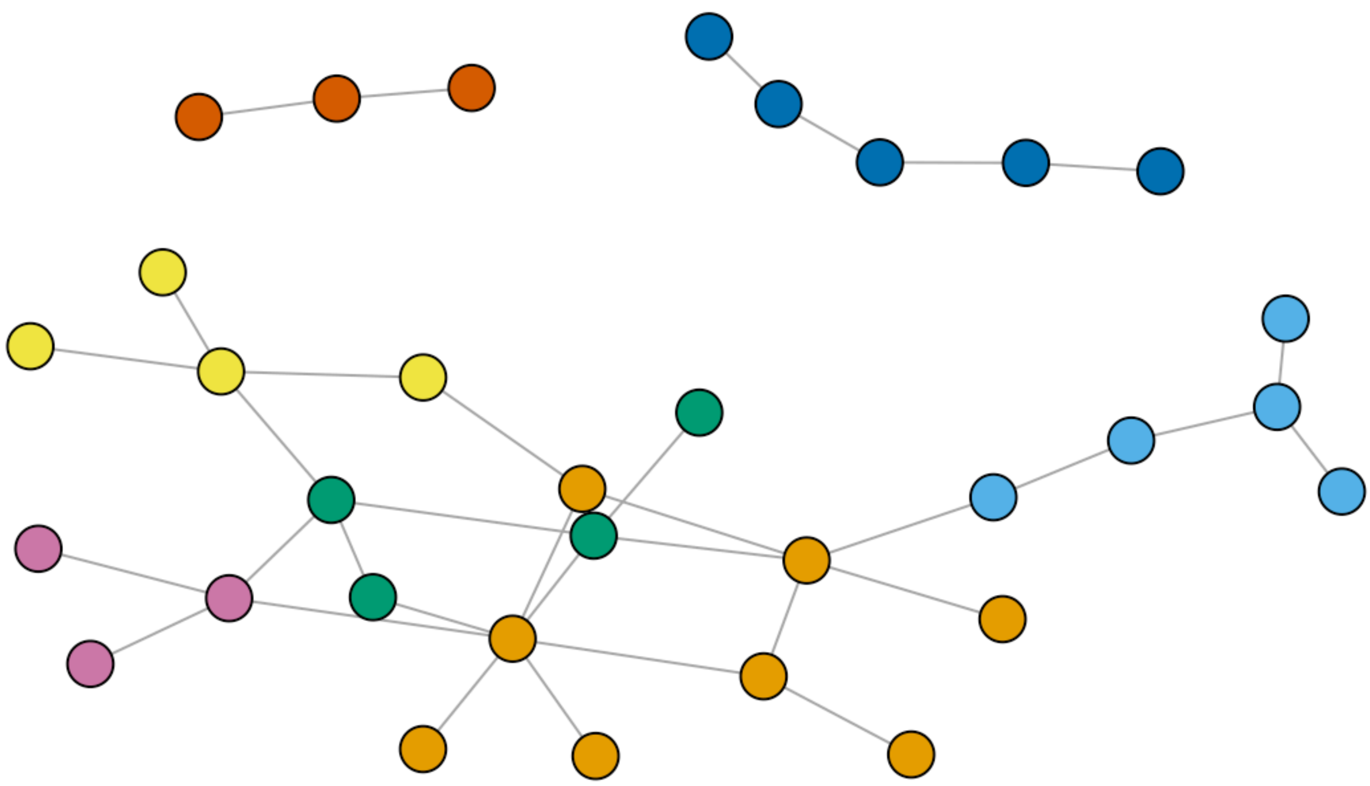
```
#What partition is the best?
max(fc$modularity)
```

```
## [1] 0.5774221
```

```
which.max(fc$modularity)
```

```
## [1] 26
```

```
#Color nodes by partitions
memb = membership(fc)
plot(ga_edges, vertex.size=7, vertex.label=NA,
     vertex.color=memb, asp=FALSE)
```



```
#How many communities received
max(levels(as.factor(memb)))
```

```
## [1] "7"
```

```
#What size of each community
summary(as.factor(memb))
```

```
## 1 2 3 4 5 6 7
## 8 5 4 4 5 3 3
```

Find commuinty with Multi-Level algorithm.

This function implements the multi-level modularity optimization algorithm for finding community structure.

```
ml = multilevel.community(ga_edges)

#Cheack what is the modularity
ml$modularity
```

```
## [1] 0.4762111 0.5804498
```

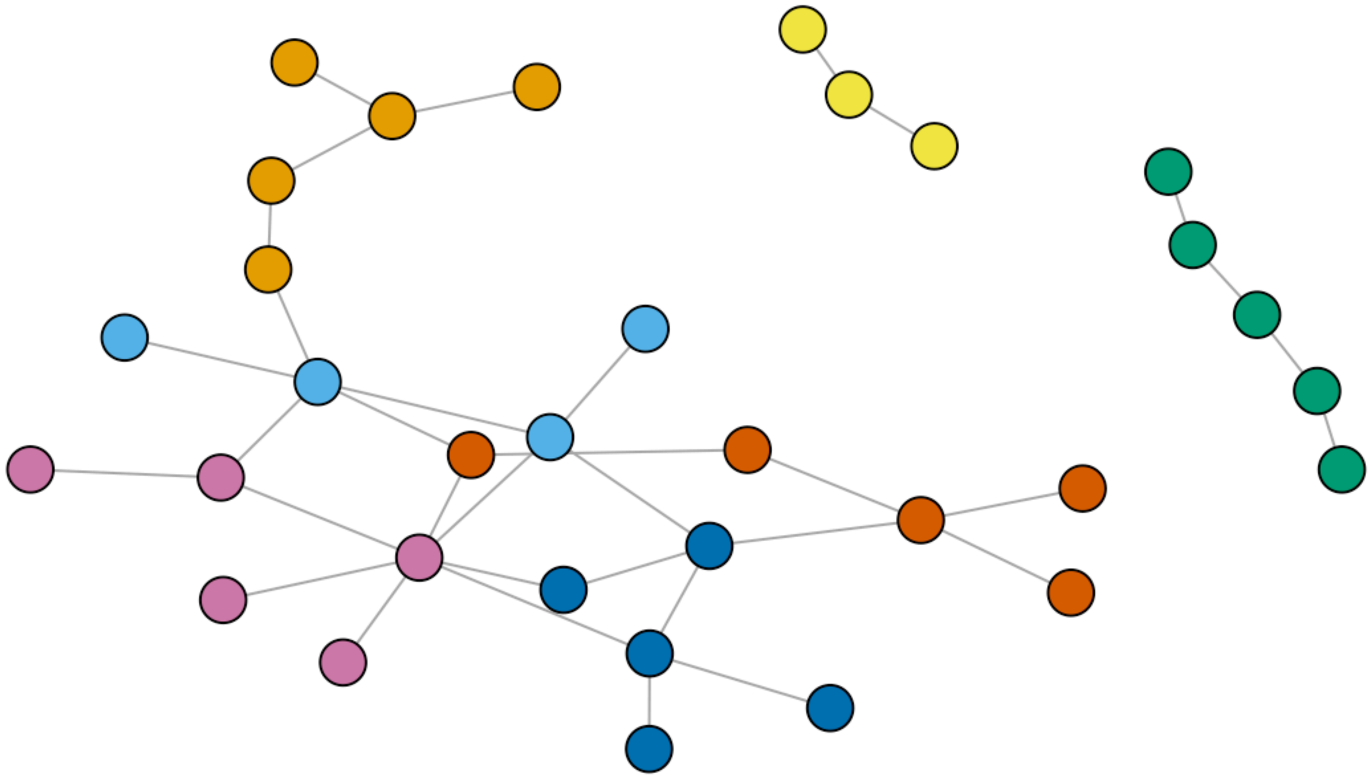
```
#What partition is the best?
max(ml$modularity)
```

```
## [1] 0.5804498
```

```
which.max(ml$modularity)
```

```
## [1] 2
```

```
#Color nodes by partitions
memb = membership(ml)
plot(ga_edges, vertex.size=7, vertex.label=NA,
     vertex.color=memb, asp=FALSE)
```



```
#How many communities received
max(levels(as.factor(memb)))
```

```
## [1] "7"
```

```
#What size of each community
summary(as.factor(memb))
```

```
## 1 2 3 4 5 6 7
## 5 4 5 3 5 5 5
```

Find communty with propagating labels algorithm

This is a fast, nearly linear time algorithm for detecting community structure in networks. In works by labeling the vertices with unique labels and then updating the labels by majority voting in the neighborhood of the vertex.

```r
#Find communinty with propagating labels algorithm
#This is a fast, nearly linear time algorithm for detecting community structure in networks.
#In works by labeling the vertices with unique labels and then updating the labels by majority voting in the neighborhood of the vertex.
pl = label.propagation.community(ga_edges)

#Cheack what is the modularity
pl$modularity
```

```
## [1] 0.533737
```

```r
#What partition is the best?
max(pl$modularity)
```

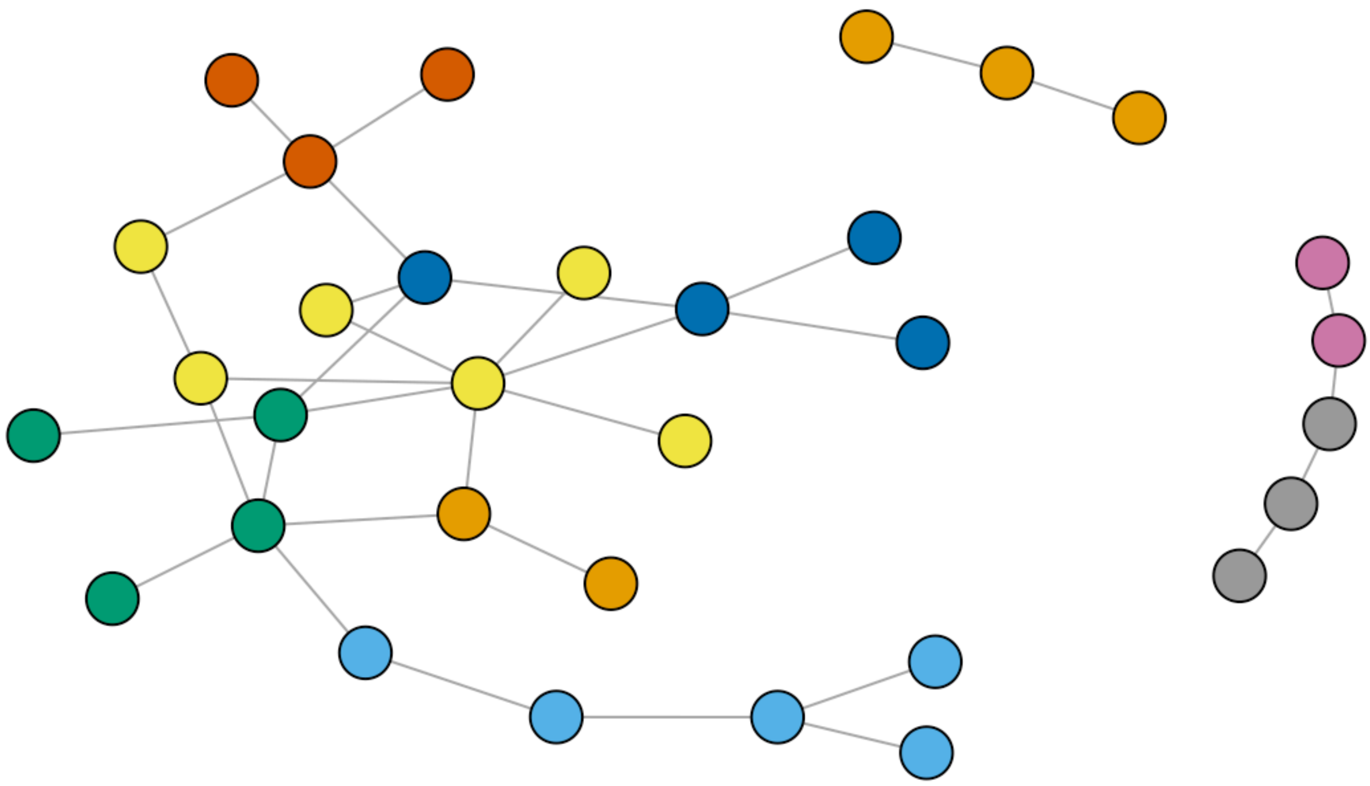
```
## [1] 0.533737
```

```r
which.max(pl$modularity)
```

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

```r
#Color nodes by partitions
memb = membership(pl)

plot(ga_edges, vertex.size=8, vertex.label=NA,
     vertex.color=memb, asp=FALSE)
```



```
#How many communities received
max(levels(as.factor(memb)))
```

```
## [1] "9"
```

```
#What size of each commuinty
summary(as.factor(memb))
```

```
## 1 2 3 4 5 6 7 8 9
## 2 5 4 6 4 3 2 3 3
```