Les_miserables

Max

Tuesday, April 07, 2015

Load Data

Load libraries

```
library(igraph)
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 3.1.1
```

```
library(reshape2)

source('metrics.R')
```

Load Les Miserables graph

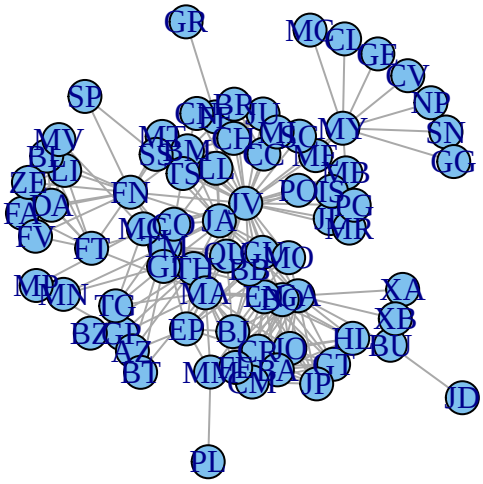
```
load("miserables.Rdata")
```

Let's view graph nodes and general structure

```
V(miserables)
```

```
## Vertex sequence:
##  [1] "MY" "NP" "MB" "ME" "CL" "GE" "MC" "CV" "SN" "GG" "JL" "JV" "MT" "MR"
## [15] "IS" "PG" "FT" "LI" "FA" "BL" "FV" "DA" "ZE" "FN" "TM" "TH" "FF" "JA"
## [29] "BM" "SP" "SS" "SC" "PO" "JU" "CH" "BR" "CN" "CC" "GP" "BZ" "CO" "EP"
## [43] "AZ" "LL" "MI" "GR" "JD" "BU" "GA" "GI" "MN" "MG" "MP" "MV" "TG" "MA"
## [57] "BT" "MM" "BO" "CR" "EN" "GT" "JO" "BA" "CM" "PL" "BB" "GU" "QU" "MO"
## [71] "TS" "XA" "XB" "BJ" "HL" "FE" "JP"
```

```
plot(miserables)
```



Let's see if we have communities here using the Grivan-Newman algorithm. 1st we calculate the edge betweenness, merges, etc...

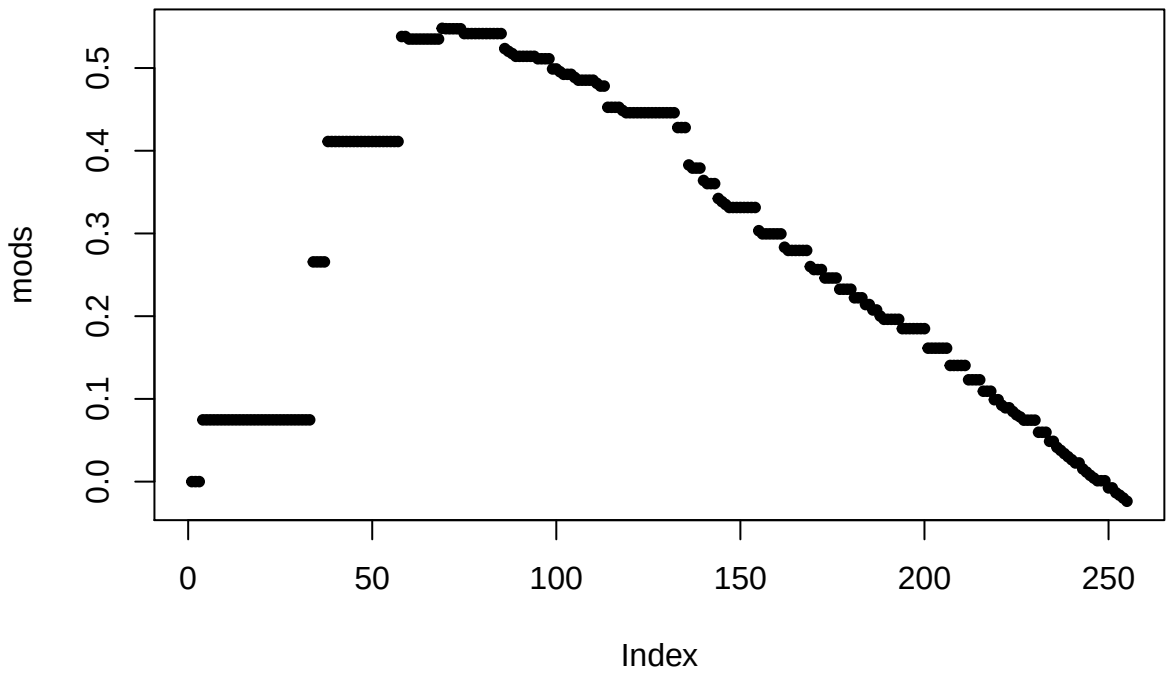
```
ebc <- edge.betweenness.community(miserables, directed=F)
```

Now we have the merges/splits and we need to calculate the modularity for each merge for this we'll use a function that for each edge removed will create a second graph, check for its membership and use that membership to calculate the modularity

```
mods <- sapply(0:ecount(miserables), function(i){
    g <- delete.edges(miserables, ebc$removed.edges[seq(length=i)])
    cl <- clusters(g)$membership
    modularity(miserables,cl)
})
```

We can now plot all modularities

```
plot(mods, pch=20)
```

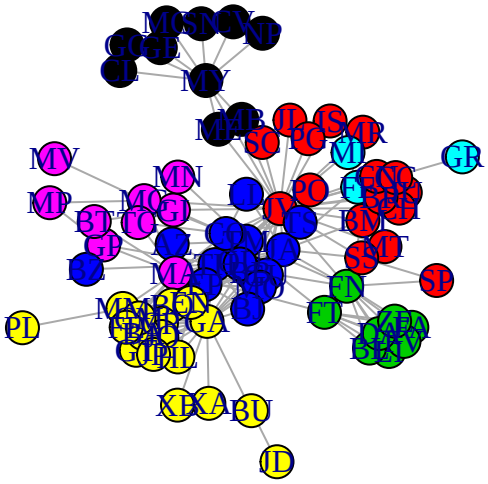


Save clusterization with max modularity

```
miserables.separated = delete.edges(miserables, ebc$removed.edges[seq(length=which.max(mods)-1)])
miserables.clust = clusters(miserables.separated)$membership
#clusters in df
df.clust = data.frame(V(miserables)$name,miserables.clust)
colnames(df.clust) = c("V","clust")
```

View graph colored by cluster

```
V(miserables)$color = miserables.clust
miserables$layout <- layout.fruchterman.reingold
plot(miserables)
```



Classification

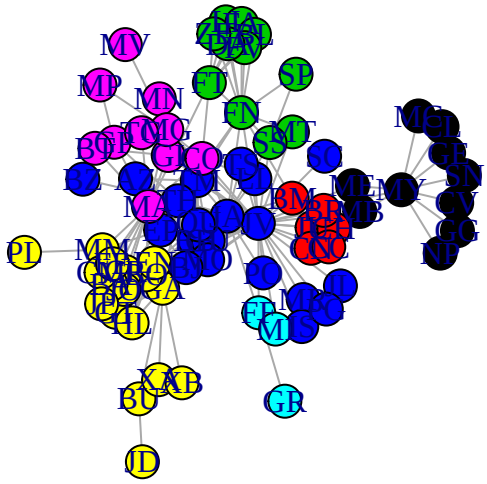
```
source('classification.R')
```

Let's classify with **logforest** distance with **alpha** parameter **0.01**

```
class = classify.multiple(times = 100, graph = miserables, distance = logforest_dist, alpha = 0.01)
```

And plot miserables graph with classification coloring

```
V(miserables)$color = class
plot(miserables)
```



Modularity and error alpha-resistance

Let's check resistance of distances in respect to modularity and error rate

```
alphas <- seq(0.01, 0.91, by=0.01)
dist.vect <- c(plainwalk_dist, walk_dist , plainforest_dist, logforest_dist, communicability_dist, logc
names(dist.vect) <- c("plain_walk", "walk", "plain_forest", "log_forest", "communicability", "log_commu
```

Calculate modularity and error rate for each alpha

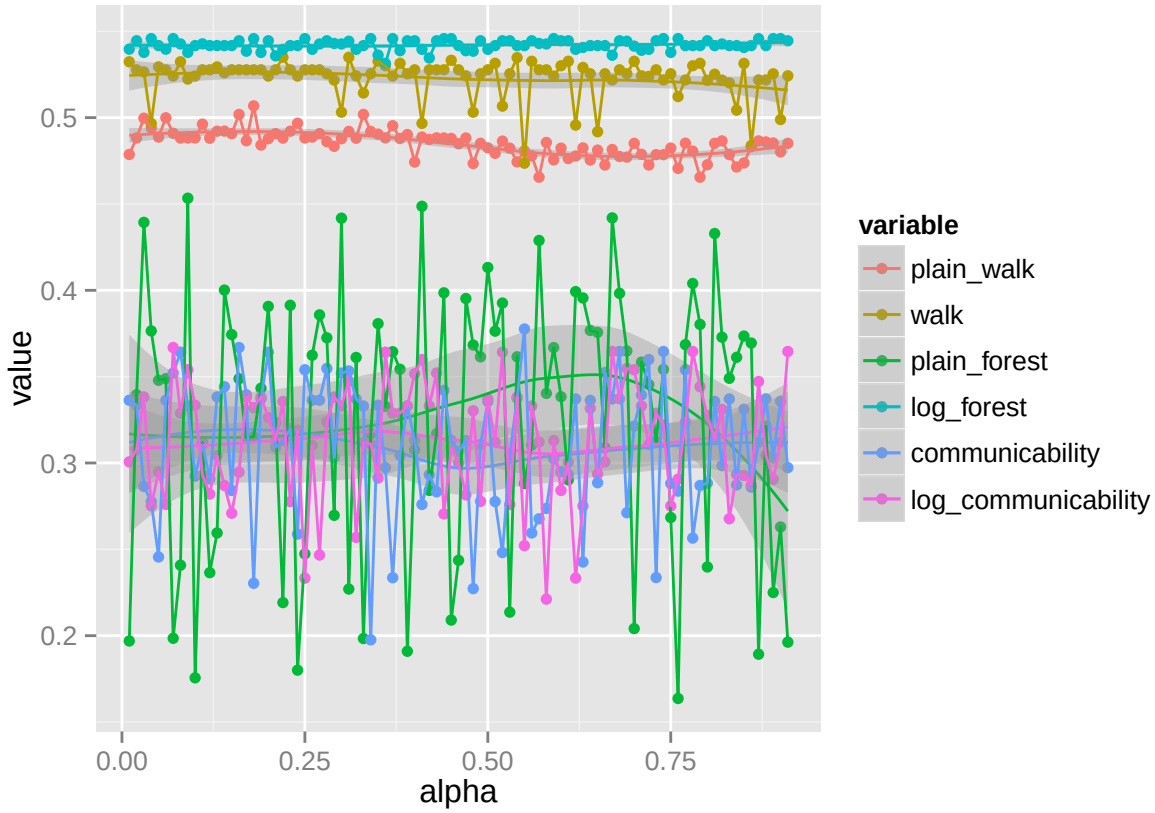
```
mods.df = data.frame(alpha = alphas)
acc.df = data.frame(alpha = alphas)

for(i in 1:length(dist.vect)) {
    mods = vector()
    acc = vector()
    for(a in alphas) {
        class = classify.multiple(times = 100, graph = miserables, distance = dist.vect[[i]], alpha = a
        mods = c(mods, modularity(miserables, class))
        acc = c(acc, sum(class==miserables.clust)/length(class))
    }
    mods.df[names(dist.vect[i])] = mods
    acc.df[names(dist.vect[i])] = acc
}
```

View dependency of alpha and modularity

```
mods.melted.df <- melt(mods.df, id=c("alpha"))
g.mods<-ggplot(mods.melted.df) +
    geom_point(aes(alpha, value, colour=variable)) +
    geom_smooth(aes(alpha, value, colour=variable)) +
    geom_line(aes(alpha, value, colour=variable))
plot(g.mods)
```

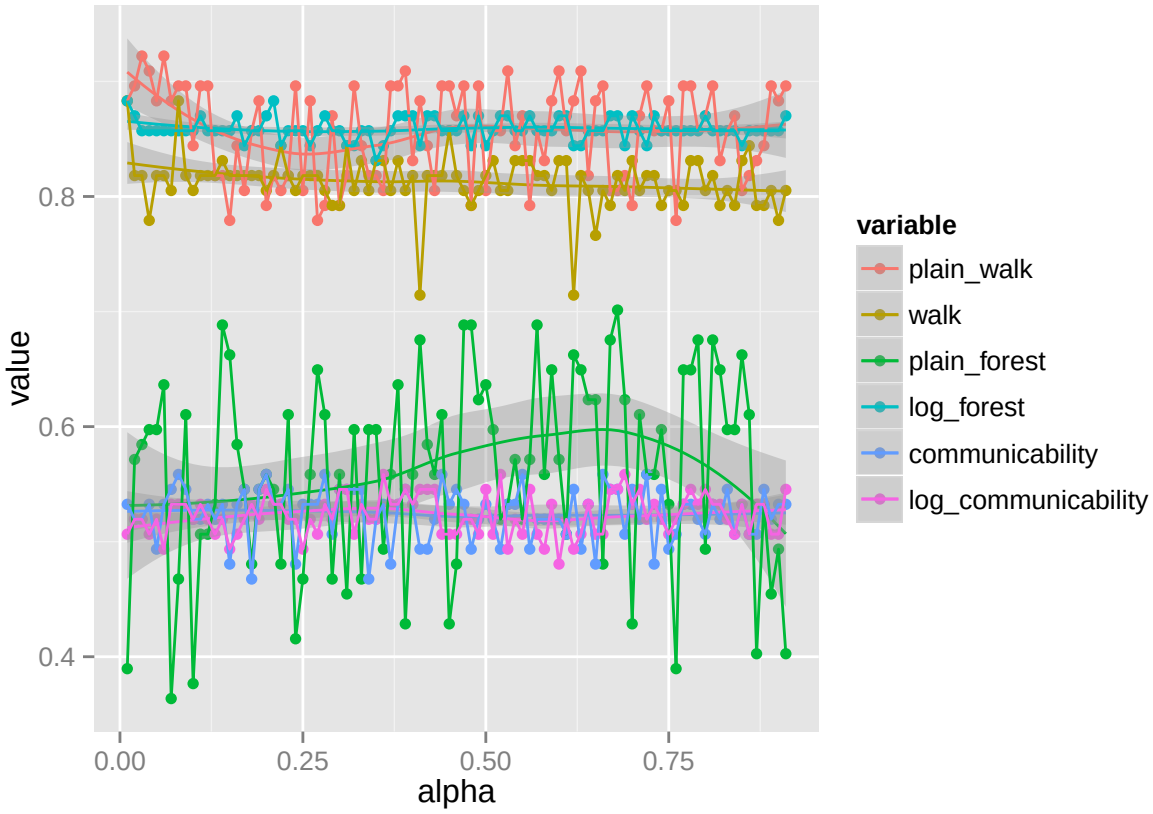
```
## geom_smooth: method="auto" and size of largest group is <1000, so using loess. Use 'method = x' to ch
```



View dependency of alpha and error

```
acc.melted.df <- melt(acc.df, id=c("alpha"))
g.acc<-ggplot(acc.melted.df) +
                geom_point(aes(alpha, value, colour=variable)) +
                geom_smooth(aes(alpha, value, colour=variable)) +
                geom_line(aes(alpha, value, colour=variable))
plot(g.acc)
```

```
## geom_smooth: method="auto" and size of largest group is <1000, so using loess. Use 'method = x' to ch
```


value
0.8
0.6
0.4
0.00
0.25
0.50
0.75
alpha
variable
plain_walk
walk
plain_forest
log_forest
communicability
log_communicability