Report

Max

Tuesday, April 28, 2015

In this report I'd like to review what have been done recently.

For starters I've rewritten some code to boost the speed of calculations, including parallel computing.

```
library(doParallel)
```

```
## Loading required package: foreach
## Loading required package: iterators
## Loading required package: parallel
```

```
numWorkers=4
```

Load and preprocessing data

Graph is loaded from the same source, but now I've manually added real characters names instead of two-letters abbriviation. I think it made graph more readable and understandable, and it also how the original paper was made.

```
load("miserables.Rdata")
names <- read.table("names.txt", stringsAsFactors=F)$V1
V(miserables)$name <- names
```

Also I've added fixed layout to ease visual comparison of graph network representation. Nodes locations are calculated based on Fruchterman-Reingold.

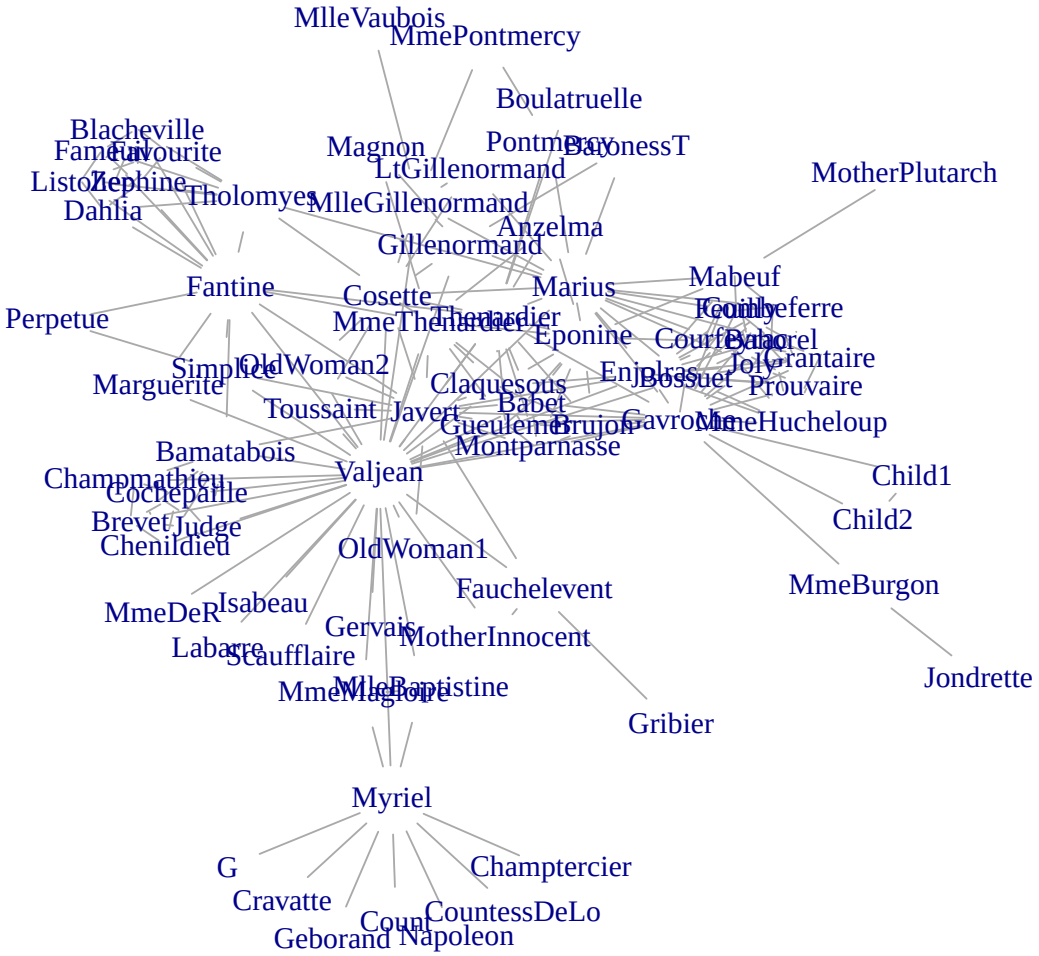
```
mylayout <- layout.fruchterman.reingold(miserables)
miserables$layout <- mylayout
```

And I've also changed color palete, to make graph colors less acidic.

```
palette(brewer.pal(7, "Dark2"))
```

Also now graph only displays names, without shaped.

```
par(mai=c(0,0,0,0))
plot(miserables, vertex.shape="none", margin=0)
```



Clusterization

Initial clustering, which is later used for random marked nodes in semi-supervised classification, is still done with Numan algorithm.

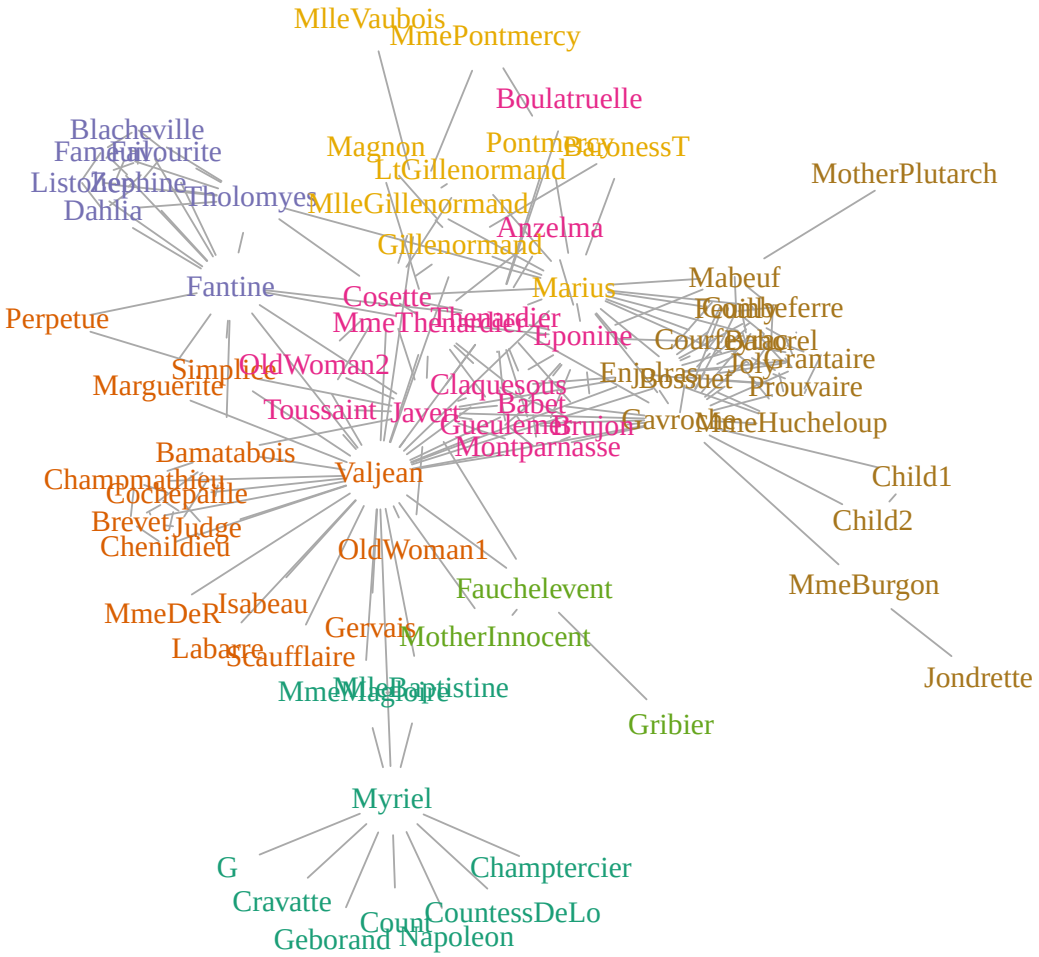
```
ebc <- edge.betweenness.community(miserables, directed=F)

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

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")
```

This is how initial clusterizatin look like

```
V(miserables)$label.color = miserables.clust
par(mai=c(0,0,0,0))
plot(miserables, vertex.shape="none")
```



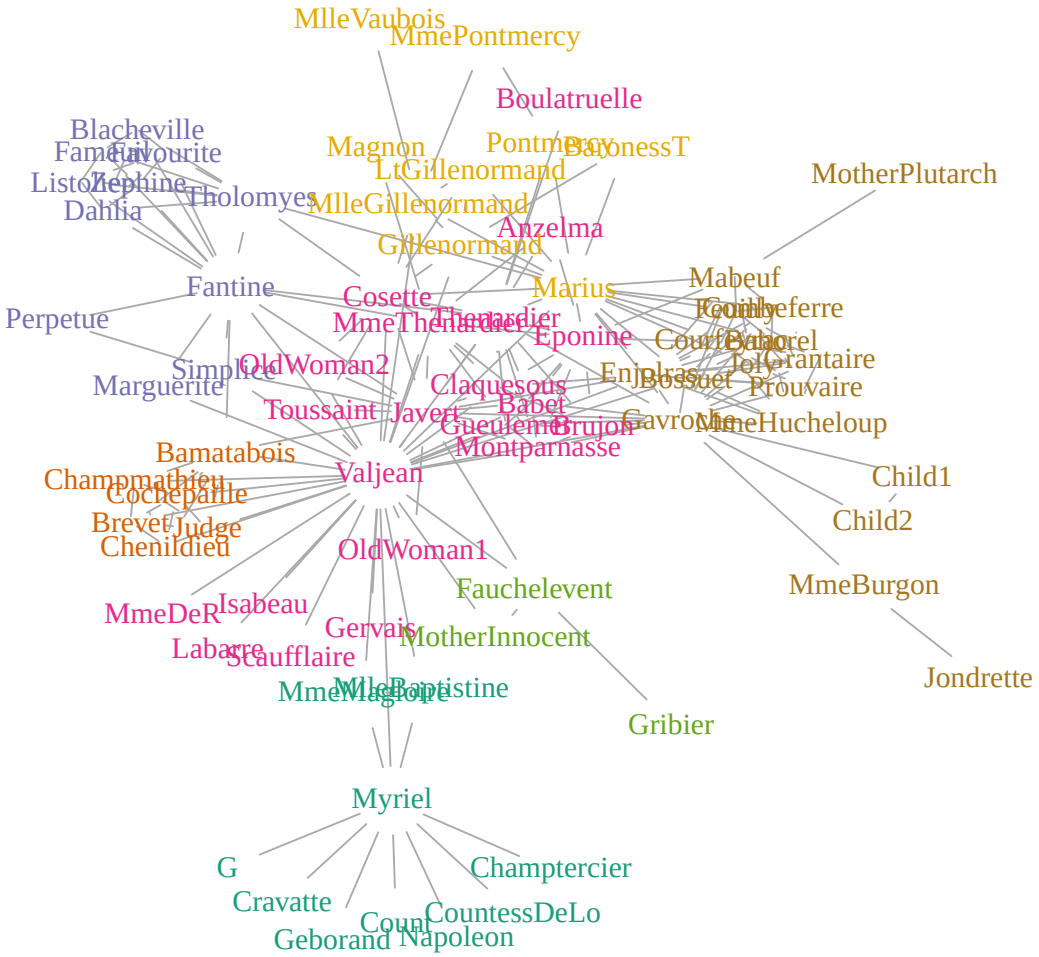
Classification

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

As well as in the original paper classification is done by the following steps:
1. From each cluster one representative is randomly chosen
2. Distance matrix is calculated based on chosen algorithm and marked nodes from step 1
3. With KNN method where k=1 classify all remaining nodes
4. Steos 1-3 are repeated multiple times (*100*) and results are reduced via voting.

The following graph has been classified with logforest and alpha-0.1

```
class = classify.multiple(times = 100, graph = miserables, clusters = df.clust, distance = logforest_dis
V(miserables)$label.color = class
par(mai=c(0,0,0,0))
plot(miserables, vertex.shape="none")
```



Dependency of modularity and error rate on alpha-parameter

Alpha range was widen to [0.01, 0.99] with step 0.01. Research is done for the save distance algorithms:
plain_walk,
walk,
plain_forest,
log_forest,
communicability,
log_communicability

```
alphas <- seq(0.01, 1.0, by=0.01)
dist.vect <- c(plainwalk_dist, walk_dist , plainforest_dist, logforest_dist, communicability_dist, logco
names(dist.vect) <- c("plain_walk", "walk", "plain_forest", "log_forest", "communicability", "log_commu
```

Classification is done by the algorithm described above and done for several distances in parallel

```
cl = makeCluster(numWorkers)
registerDoParallel(cl)
strt <- Sys.time()
#calculate modularity for each alpha in alphas
lres = foreach(i = 1:length(dist.vect), .packages="igraph") %dopar% {
        source("metrics.R")
        mods = vector()
        acc = vector()
        for(a in alphas) {
            class = classify.multiple(times = 100, graph = miserables, clusters = df.clust, distance = d
            mods = c(mods, modularity(miserables, class))
            acc = c(acc, sum(class==miserables.clust)/length(class))
        }
        data.frame(mods, acc)
}
print(Sys.time()-strt)
```

```
## Time difference of 9.853 mins
```

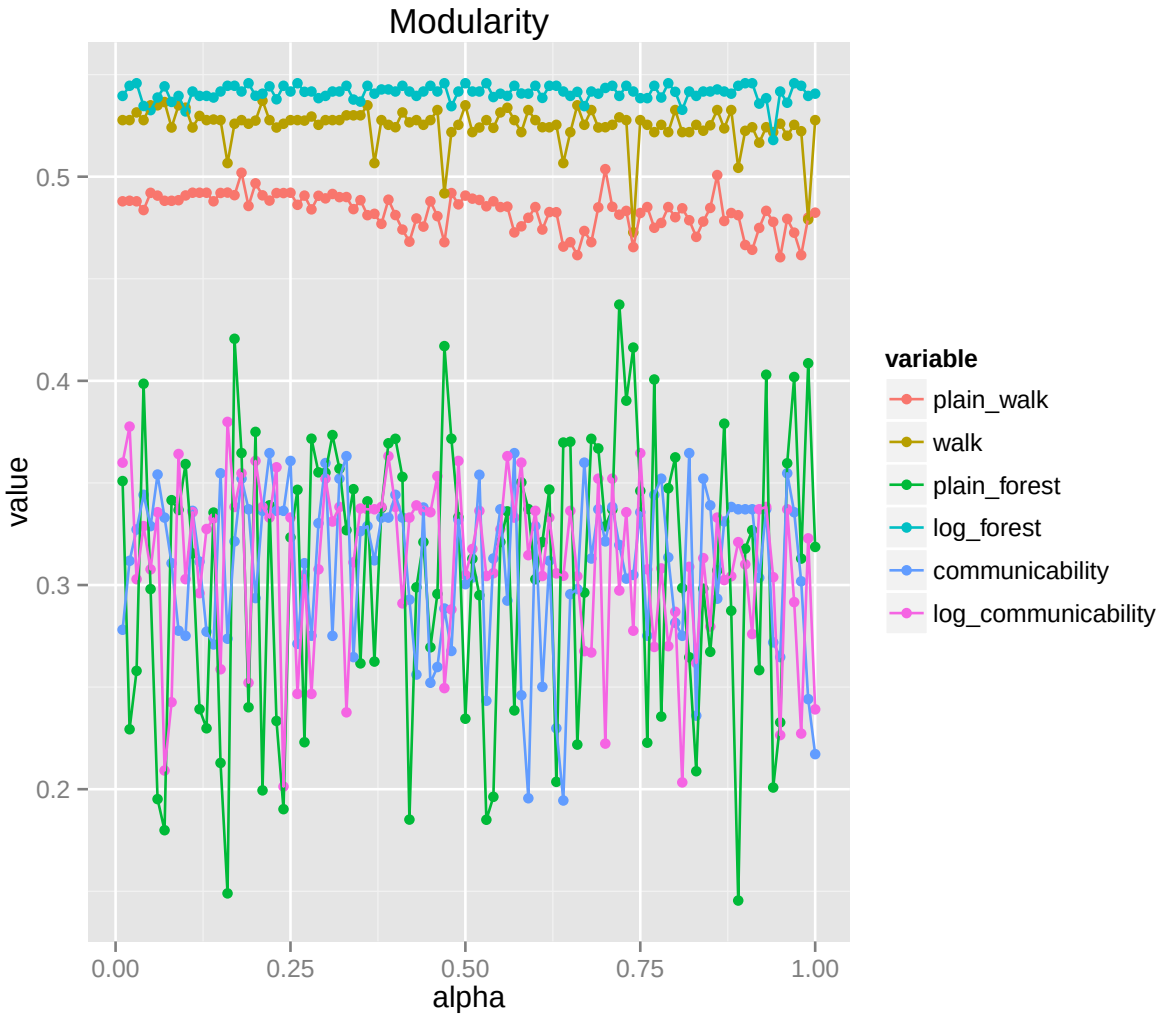
```
stopCluster(cl)

mods.df = data.frame(alpha = alphas)
acc.df = data.frame(alpha = alphas)
for(i in 1:length(dist.vect)) {
    mods.df[names(dist.vect[i])] = lres[[i]]$mods
    acc.df[names(dist.vect[i])] = lres[[i]]$acc
}
```

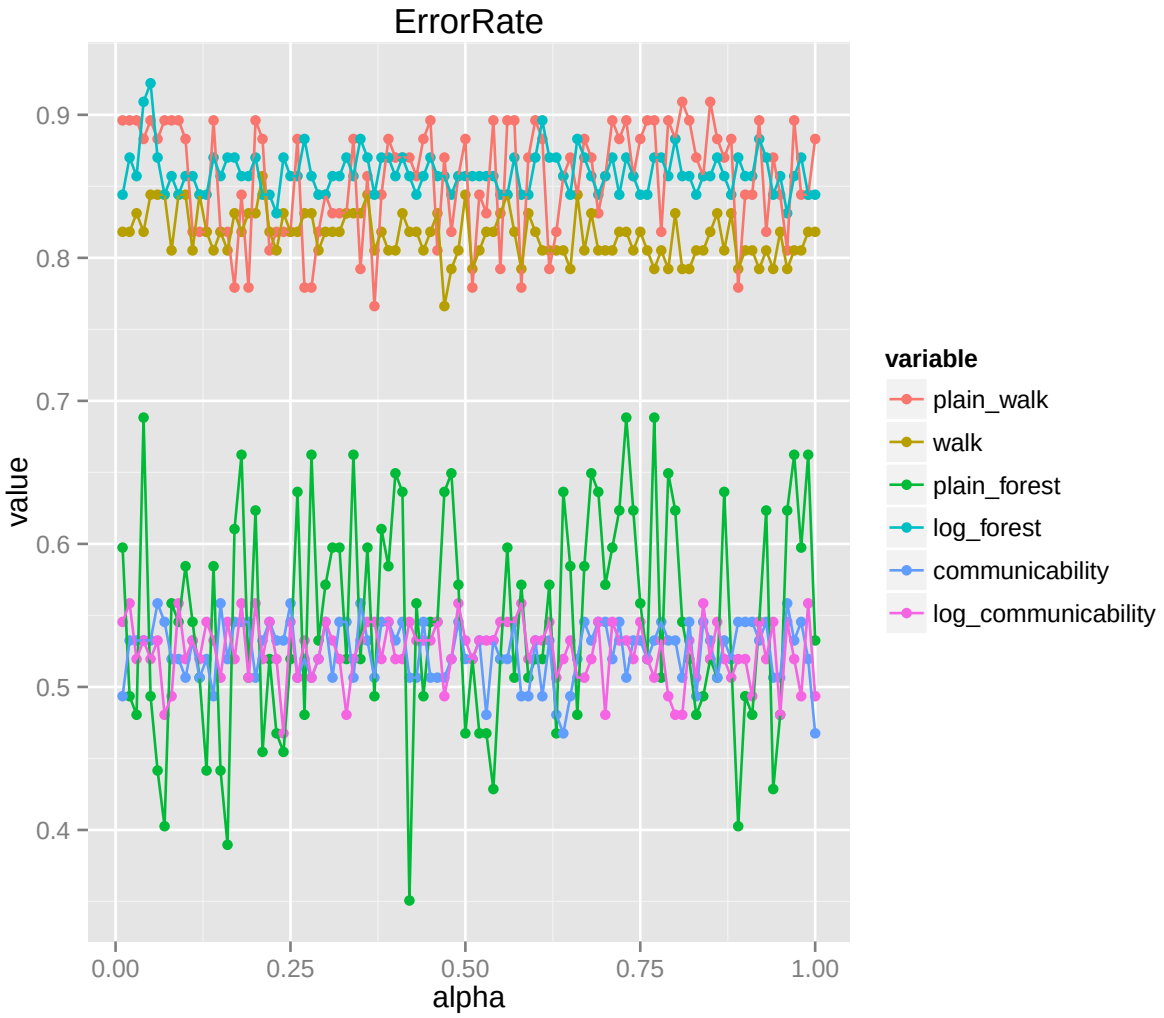
Results

This is already familiar graphics.

```
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)) +
    ggtitle("Modularity")
plot(g.mods)
```




```
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)) +
                  ggtitle("ErrorRate")
plot(g.acc)
```



As it appears some distances are more consistent with results and has better results than others. But I can't see any dependence on alpha-parameter, which is supported by the next research.

Research with fixed parameter

To prove that high diversity has nothing to do with alpha parameter the same reaseach was conducted, but with fixed alpha (0.01) for many times(100).

```
a = 0.01

cl = makeCluster(numWorkers)
registerDoParallel(cl)
strt <- Sys.time()
#calculate modularity and accuracy
lres = foreach(i = 1:length(dist.vect), .packages="igraph") %dopar% {
    source("metrics.R")
    mods = vector()
    acc = vector()
    for(j in 1:100) {
```

```
        class = classify.multiple(times = 100, graph = miserables, clusters = df.clust, distance = dist
        mods = c(mods, modularity(miserables, class))
        acc = c(acc, sum(class==miserables.clust)/length(class))
    }
    data.frame(mods, acc)
}
print(Sys.time()-strt)
```

```
## Time difference of 8.906 mins
```

```
stopCluster(cl)

fix.mods.df = data.frame(matrix(NA, nrow=100, ncol=0))
fix.acc.df = data.frame(matrix(NA, nrow=100, ncol=0))
for(i in 1:length(dist.vect)) {
    fix.mods.df[names(dist.vect[i])] = lres[[i]]$mods
    fix.acc.df[names(dist.vect[i])] = lres[[i]]$acc
}
```

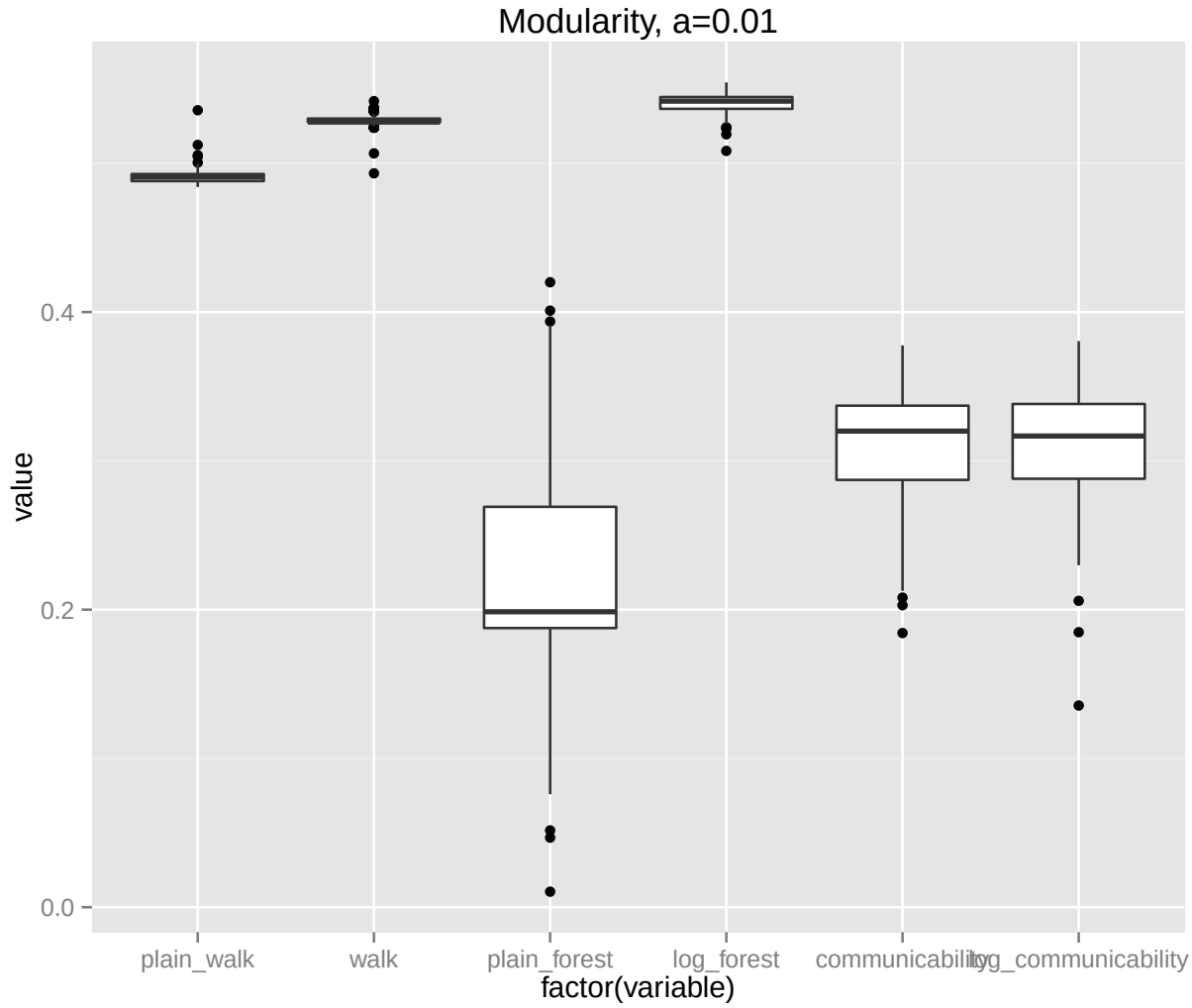
Results

Results are shown on boxplot graphs, where the end of whiskers and borders of rectangles represent quartiles. 50% of all results lay in the area of rectangles. Also the dots represents min and max values.

```
fix.mods.melted.df <- melt(fix.mods.df)
```

```
## No id variables; using all as measure variables
```

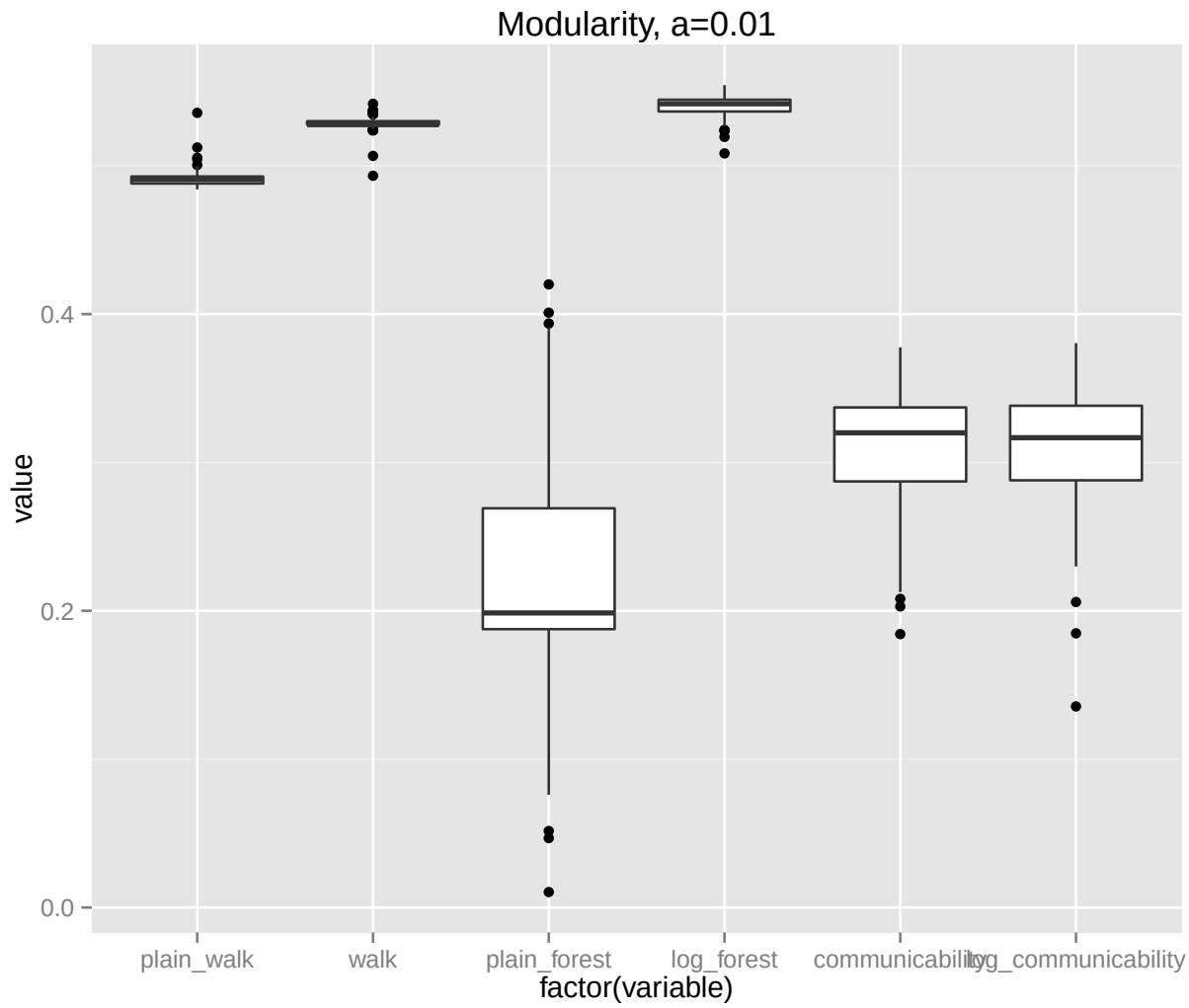
```
g.fix.mod <- ggplot(fix.mods.melted.df, aes(factor(variable),value)) +
                    geom_boxplot() + ggtitle("Modularity, a=0.01")
plot(g.fix.mod)
```




```
fix.acc.melted.df <- melt(fix.acc.df)
```

```
## No id variables; using all as measure variables
```

```
g.fix.acc <- ggplot(fix.acc.melted.df, aes(factor(variable),value)) +
    geom_boxplot() + ggtitle("ErrorRate, a=0.01")
plot(g.fix.mod)
```



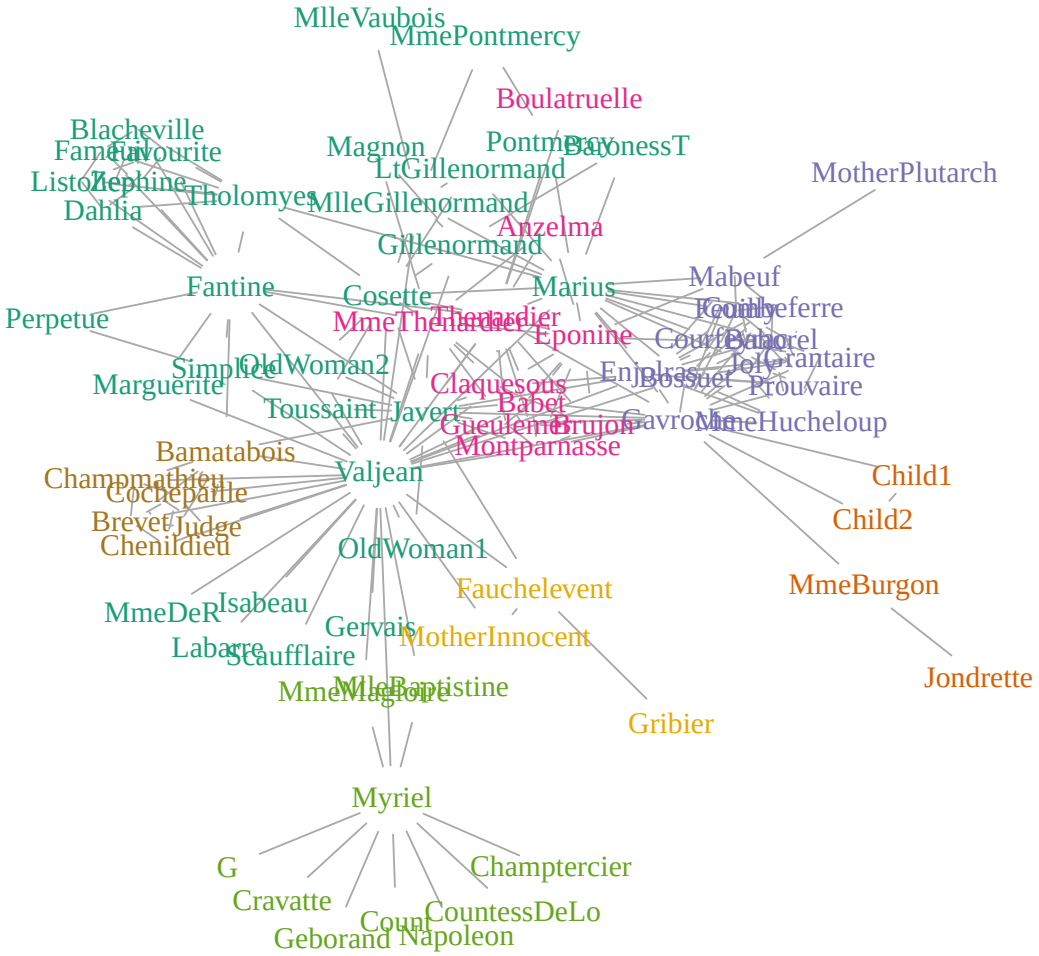
As we see, even with fixed parameter results spreaded the same way they are in previous graphs.

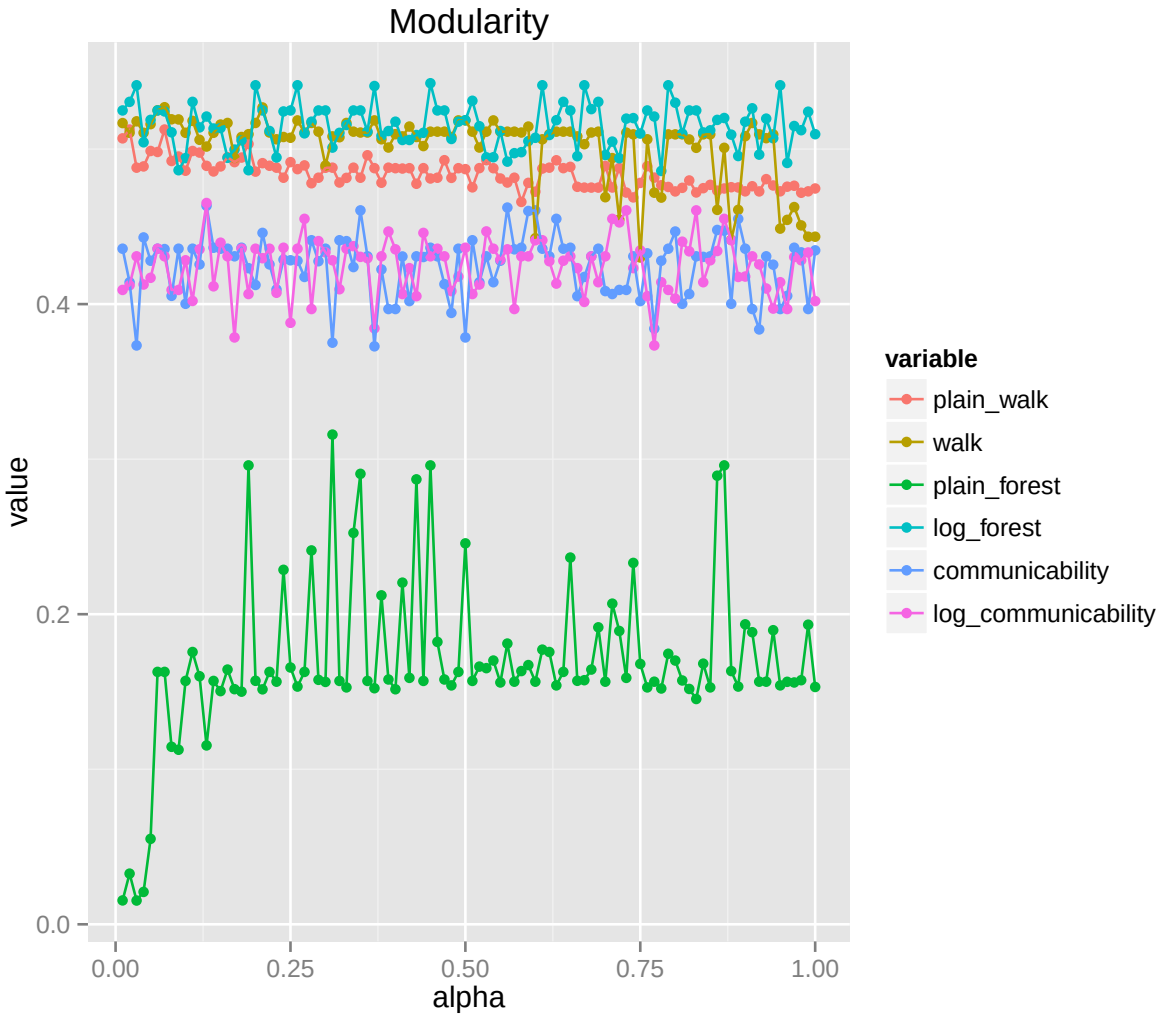
Other modifications

It's worth to mention that the same research was also conducted with another initial clustering algorithm instead Newman one. Walktrap algorithm, which is based on random walk was also tested.

```
wc <- walktrap.community(miserables)
miserables.clust = wc$membership
df.clust = data.frame(V(miserables)$name,miserables.clust)
colnames(df.clust) = c("V","clust")
```

```
V(miserables)$label.color = miserables.clust
par(mai=c(0,0,0,0))
plot(miserables, vertex.shape="none")
```


MlleVaubois
MmePontmercy
Boulatruelle
Blacheville
Fameuil
Favourite
Magnon
Pontmercy
BaronessT
LtGillenormand
MotherPlutarch
Listolier
Zephine
Tholomyes
MlleGillenormand
Dahlia
Anzelma
Gillenormand
Mabeuf
Fantine
Cosette
Marius
Perpetue
MmeThenardier
Thenardier
Eponine
Simplice
OldWoman2
Marguerite
Claquesous
Babet
Toussaint
Javert
Gueulemer
Brujon
Montparnasse
Bamatabois
Valjean
Champmathieu
Cochepaille
Child1
Child2
Brevet
Judge
Chenildieu
OldWoman1
MmeBurgon
Fauchelevent
MmeDeR
Isabeau
Gervais
MotherInnocent
Labarre
Stafflaire
Jondrette
MmeMagloire
MlleBaptistine
Gribier
Myriel
G
Champtercier
Cravatte
Count
CountessDeLo
Geborand
Napoleon


Modularity
value
0.0
0.2
0.4
0.00
0.25
0.50
0.75
1.00
alpha
variable
plain_walk
walk
plain_forest
log_forest
communicability
log_communicability


Error Rate
value
0.4
0.6
0.8
variable
plain_walk
walk
plain_forest
log_forest
communicability
log_communicability

Communicability distance as well as its logarithmic version now show much better results.

Questions

It's difficult to interprete above results comparing them to results of original work. It seems that our distances are resistant to alpha parameter change. And it doesn't matter what alpha value to pick. And at the same time results of communicability distance depends on the nature of clusters origin.