

# Social Network Analysis Home Assignment 3

*Shupeyko Nikita*

*due date - 05.06.2016 23:59*

## Contents

<b>Network communities</b>	<b>1</b>
Yahoo Music Network . . . . .	1

## Network communities

### Yahoo Music Network

For this assignment, you have to load the part of Yahoo Music Network. Archive contains network in GML format and corresponding list of artists.

```
yg.f = "music2K.gml"
yg = read.graph(yg.f, format = c("gml"))
```

Edges in this network appear if enough number of users have given ratings to both music bands. Note, that edges are weighted with similarity of the ratings.

#### 1. Preprocessing

This network needs to be preprocessed. Do the steps, described on seminar, to make it look properly. How many vertices and edges were deleted?

```
# Deleting isolated nodes.
yg.isolated.vertices= V(yg)[degree(yg) == 0]
yg.delete.vertices.diff = vcount(yg)
yg = delete.vertices(yg, degree(yg) == 0)
yg.delete.vertices.diff = yg.delete.vertices.diff - vcount(yg)
paste("Deleted vertex count: ", yg.delete.vertices.diff)
```

[1] "Deleted vertex count: 4"

```
# Simplifying the graph's overall structure
# (get rid of loops and multiedges).
yg.deleted.edge.diff = ecount(yg)
yg = simplify(yg)
yg.deleted.edge.diff = yg.deleted.edge.diff - ecount(yg)
paste("Deleted edge count: ", yg.deleted.edge.diff)
```

[1] "Deleted edge count: 0"

```

# Obtaining descriptives.
func_print.graph.descriptives = function(g) {
  g.n = vcount(g)
  g.m = ecounth(g)
  g.dens = graph.density(g)
  g.avgdeg = 2 * g.m / g.n
  g.avgpathlen = average.path.length(g)
  g.diam = diameter(g)
  tbl = rbind(g.n,
              g.m,
              g.dens,
              g.avgdeg,
              g.avgpathlen,
              g.diam)
  rownames(tbl) = c("Node count",
                    "Edge count",
                    "Network Density",
                    "Average Node Degree",
                    "Average Path Length",
                    "Diameter")
  colnames(tbl) = c("Value")
  print(xtable(tbl),
        type = "latex",
        floating = T,
        include.rownames = T,
        latex.environments = "center")
}
func_print.graph.descriptives(yg)

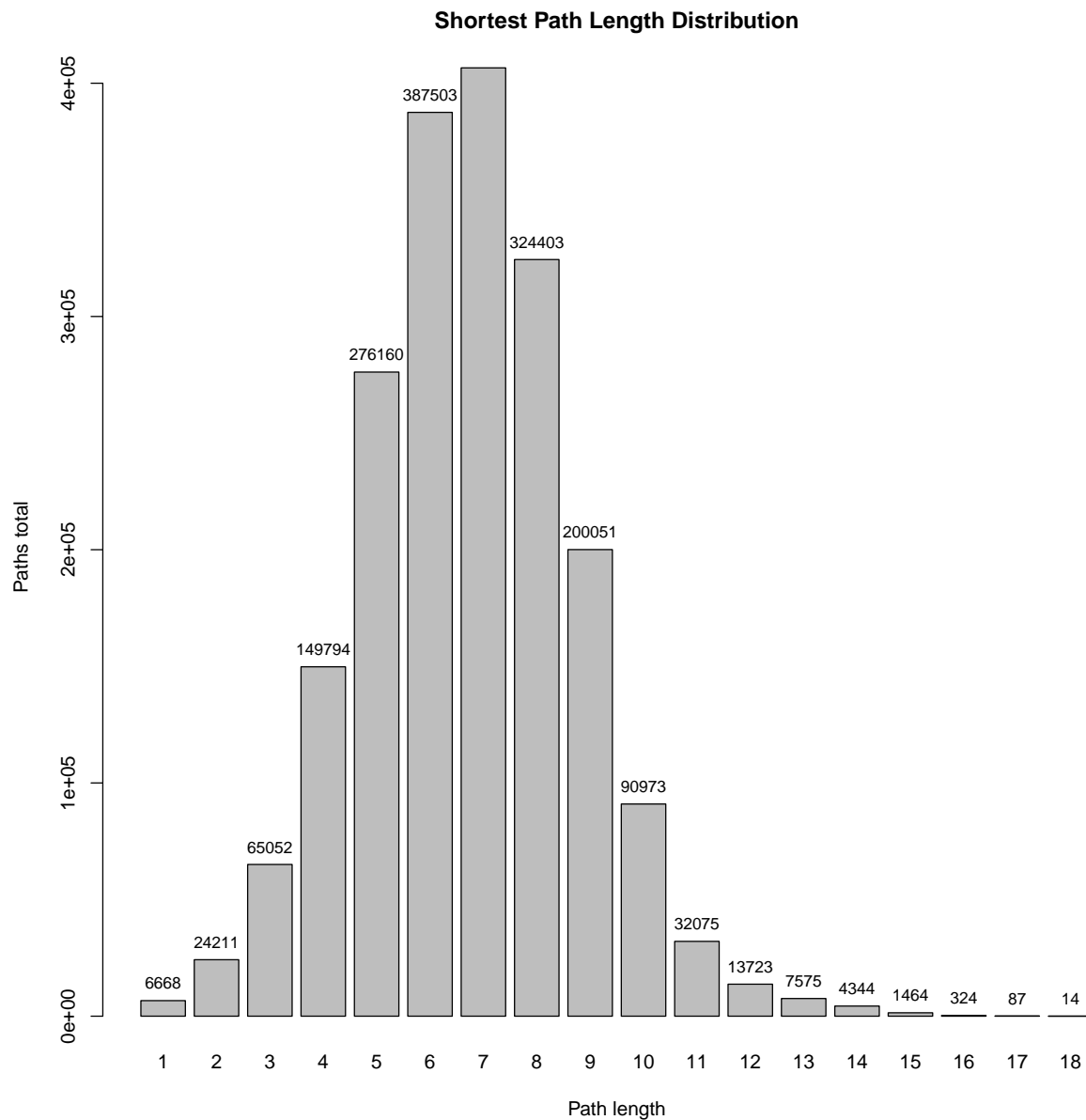
```

	Value
Node count	1996.00
Edge count	6668.00
Network Density	0.00
Average Node Degree	6.68
Average Path Length	6.74
Diameter	3.54

```

func_plot.graph.shortestpathsdistribution = function(g) {
  g.plh = path.length.hist(g)$res
  g.plh.tbl = as.table(g.plh)
  names(g.plh.tbl) = 1:length(g.plh.tbl)
  g.plh.bp = barplot(g.plh.tbl,
                     main = "Shortest Path Length Distribution",
                     xlab = "Path length",
                     ylab = "Paths total")
  text(x = g.plh.bp,
       y = g.plh,
       label = g.plh,
       cex = 0.8,
       pos = 3)
}
func_plot.graph.shortestpathsdistribution(yg)

```



## 2. Clustering

Define clusters for this networks using one of the algorithms described on lectures and seminars. Compute density of obtained clusters:

```
# Thanks to its speed on large graphs, the Louvain algorithm is used
# to reveal communities in the given network.
yg.communities = multilevel.community(yg)
yg.communities.count = length(groups(yg.communities))

# yg.communities.as.subgraphs = c()
yg.communities.as.subgraphs = vector("list", yg.communities.count)
for (i in seq_along(yg.communities.as.subgraphs)) {
```

```

    yg.communities.as.subgraphs[[i]] = induced.subgraph(yg, yg.communities$membership == i)
}

yg.community.densities = vector("list", yg.communities.count)
for (i in seq_along(yg.community.densities)) {
  yg.community.densities[[i]] = graph.density(yg.communities.as.subgraphs[[i]])
}
yg.community.names = vector("list", yg.communities.count)
for (i in seq_along(yg.community.names)) {
  yg.community.names[[i]] = paste("Community #", i)
}

tbl = cbind(as.matrix(yg.community.names),
            as.matrix(yg.community.densities))
colnames(tbl) = c("", "Density")
tbl = xtable(tbl)
align(tbl) = "c|r|c"
print(tbl,
      floating = T,
      type = "latex",
      include.rownames = F,
      latex.environments = "center")

```

V1	Density
Community # 1	0.54
Community # 2	0.13
Community # 3	0.03
Community # 4	0.10
Community # 5	0.07
Community # 6	0.10
Community # 7	0.05
Community # 8	0.17
Community # 9	0.04
Community # 10	0.18
Community # 11	0.04
Community # 12	0.05
Community # 13	0.06
Community # 14	0.09
Community # 15	0.25
Community # 16	0.05
Community # 17	0.35
Community # 18	0.10
Community # 19	0.06
Community # 20	0.04
Community # 21	0.07
Community # 22	0.11
Community # 23	0.64
Community # 24	0.10
Community # 25	0.14
Community # 26	0.11
Community # 27	0.14
Community # 28	0.11

Compute the ratio of inner clusters connections to outer ones:

```
yg.m = ecount(yg)
yg.community.inoutrratios = vector("list", yg.communities.count)
for (i in seq_along(yg.community.inoutrratios)) {
  c = yg.communities.as.subgraphs[[i]]
  c.m = ecount(c)
  yg.community.inoutrratios[[i]] = c.m / ( yg.m - c.m )
}

tbl = cbind(as.matrix(yg.community.names),
            as.matrix(yg.community.inoutrratios))
colnames(tbl) = c("", "Inner-Outer Connections Ratio")
tbl = xtable(tbl)
digits(tbl) = c(0, 0, 4)
align(tbl) = "c|r|c"
print(tbl,
      floating = T,
      type = "latex",
      include.rownames = F,
      latex.environments = "center")
```

V1	Inner-Outer Connections Ratio
Community # 1	0.0063
Community # 2	0.0114
Community # 3	0.0630
Community # 4	0.0229
Community # 5	0.0257
Community # 6	0.0285
Community # 7	0.0341
Community # 8	0.0071
Community # 9	0.0710
Community # 10	0.0162
Community # 11	0.0593
Community # 12	0.0539
Community # 13	0.0308
Community # 14	0.0367
Community # 15	0.0106
Community # 16	0.0420
Community # 17	0.0080
Community # 18	0.0369
Community # 19	0.0398
Community # 20	0.0689
Community # 21	0.0507
Community # 22	0.0320
Community # 23	0.0027
Community # 24	0.0229
Community # 25	0.0197
Community # 26	0.0380
Community # 27	0.0165
Community # 28	0.0362

### 3. Visualization & interpretation

Visualize five of the most dense clusters. Use names of artists as node labels on the graph.

```
yg.artists = scan("artists.txt", what = character(), sep = "\n")
yg.artists = yg.artists[-yg.isolated.vertices]
V(yg)$label = yg.artists

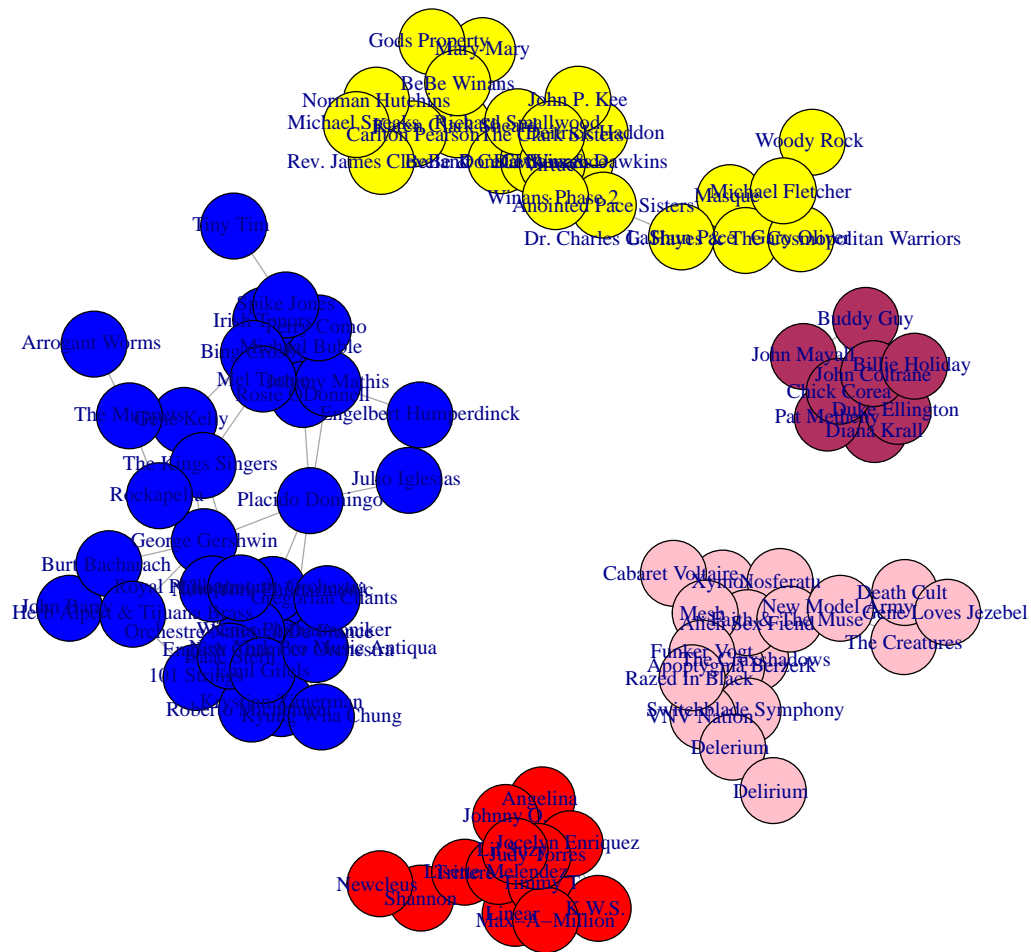
func_top.N.indecies = function(l, N) {
  l = unlist(l)
  N = as.integer(N)
  which( l > sort(l)[ length(l)-N ] )
}

yg.communities.top.5.density.indecies = func_top.N.indecies(yg.community.densities, 5)
yg.communities.top.5.density = yg.communities[yg.communities.top.5.density.indecies]
yg.vertices.in.communities.top.5.density = V(yg)[yg.communities$membership %in% yg.communities.top.5.density.indecies]

# palette()
crp = colorRampPalette(c("red", "blue", "yellow", "pink", "maroon"))(5)
for (v.i in yg.vertices.in.communities.top.5.density) {
  V(yg)[v.i]$color = crp[which( yg.communities.top.5.density.indecies == yg.communities$membership[v.i] )]
  V(yg)[v.i]$label = yg.artists[v.i]
}

yg.subgraph.vertices.in.communities.top.5.density = induced.subgraph(yg, yg.vertices.in.communities.top.5.density)

# V(yg.subgraph.vertices.in.communities.top.5.density)$color = 7
plot(yg.subgraph.vertices.in.communities.top.5.density)
```



(Extra task) Try to interpret (using Wikipedia or other resources) obtained clusters.

Artist clusters colored by genre:

*Sky Blue* — **Pop/Folk**;

*Blue* — **Jazz/Blues**;

*Dark Green* — **Gospel**;

*Orange* — **Electro/Hip-hop/Dance**

*Yellow* — **Rock/Punk-rock**