Visualization.R

mpereda

Thu Jul 21 20:01:59 2016

```
#3D graphs in R with OpenGL

#Required libraries
rm(list=ls(all=TRUE))
library(igraph)
```

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

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

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

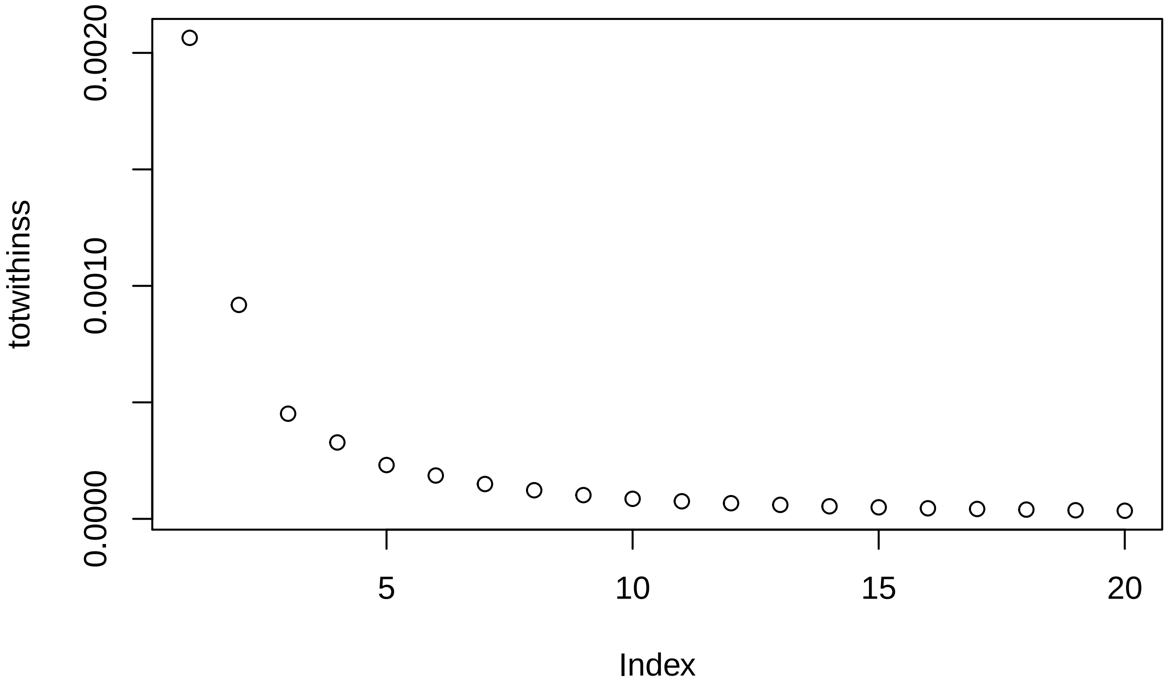
```
library(rgl)
library(xlsx)
```

```
## Loading required package: rJava

## Loading required package: xlsxjars
```

```
# Seed for random numbers generation
set.seed(1)

#Working directory
myWD <- "/Users/mpereda/Dropbox/work-Other projects/Estrada/ejemplos/WorldTrade_Dichot_SymA"
setwd(myWD)

#Load files
myNet <- read.table("WorldTrade_Dichot_SymA.txt", header = FALSE, sep = "")
coordsAndsc <- read.table("WorldTrade_Dichot_SymA-metricsstress-coords-sc.txt", header = FALSE, sep = "
coords <- coordsAndsc[,1:3]
sc<-coordsAndsc[,4]
anglesAndGpq <-read.table("WorldTrade_Dichot_SymA-angles-Gpq.txt", header = FALSE, sep = ",")
angles <- anglesAndGpq[,1]
Gpq <- anglesAndGpq[,2]
An <- read.table("WorldTrade_Dichot_SymA-anglesFull.txt", header = FALSE, sep = ",")
names <- read.table("WorldTrade_Names.txt", header = FALSE, sep = "", stringsAsFactors=FALSE)


#Creation of the network
g <- graph_from_adjacency_matrix(as.matrix(myNet), mode=c("undirected"))
```

```r
degrees<- as.matrix(degree(g))
g <- set.vertex.attribute(g, "sc", value=sc)
g <- set.vertex.attribute(g, "degree", value=degree(g))
g <- set.edge.attribute(g, "communicability_angle", value=angles)
g <- set.edge.attribute(g, "Gpq", value=Gpq)
rbPal <- colorRampPalette(c('paleturquoise1','black'))
Col <- rbPal(10)[as.numeric(cut(E(g)$Gpq,breaks = 10))] #Colors for edges
g <- set.edge.attribute(g, "color", value=Col)
g <- set.vertex.attribute(g, "names", value=names[,2])


# # # # # # # Detecting classical communities with Infomap # # # # # # # # # # # # # # # # # # # # # # #
lec <- cluster_leading_eigen(g)
clusters<-cluster_infomap(g, nb.trials = 100)
g <- set.vertex.attribute(g, "group", value=membership(clusters))
if (length(clusters)!=1){
ColNode <- rainbow(length(clusters))[as.numeric(cut(V(g)$group,breaks = length(clusters)))]
}else{ColNode=rep("#666666",length(sc))}
g <- set.vertex.attribute(g, "color", value=ColNode)


# Para cada cluster, calculo el centroide, la distancia al centroide de cada nodo, y luego el radio med
n_elements <-rep(0, length(clusters))
centroid <-list()
rads <- list()
Avg_radius <-rep(0, length(clusters))
pack_index <-rep(0, length(clusters)) #Pack_index: Compacticidad Avg_radius/n_elements
max_radius <-rep(0, length(clusters))

for (i in 1:length(clusters)){
  centroid[[i]] <- colMeans(coords[V(g)$group==i,])
  rads[[i]] <- sqrt(rowSums((coords[V(g)$group==i,]-centroid[[i]])^2))
  Avg_radius[i] <- mean(rads[[i]])
  n_elements[i] <- nrow(coords[V(g)$group==i,])
  pack_index[i] <- Avg_radius[i] / n_elements[i]
  max_radius[i] <- max(rads[[i]])
}


# Plot

#Node sizes
mns <- 5 #Min node size for plotting
Mns <- 20 #Max node size for plotting
a <- (Mns-mns)/(max(sc)-min(sc))
y0 <- Mns - ((Mns-mns)*max(sc))/(max(sc)-min(sc))

#edge sizes
mes <- 1 #Min edge size for plotting
Mes <- 10 #Max edge size for plotting
ae <- (Mes-mes)/(max(Gpq)-min(Gpq))
y0e <- Mes - ((Mes-mes)*max(Gpq))/(max(Gpq)-min(Gpq))
```

```
rgl.open()# Open a new RGL device
par3d("windowRect"= c(0,0,1300,1300))
rgl.bg(color = "white") # Setup the background color
rglplot(g, layout=as.matrix(coords),vertex.color=V(g)$color, vertex.size= y0 + a*V(g)$sc, vertex.label =
        vertex.label.color="black", vertex.label.dist=1, edge.width= y0e + ae*E(g)$Gpq, edge.color=E(g)$
#legend3d("top", legend ="Infomap communities", cex=1, bty = "n")
rgl.viewpoint(  zoom = .8 )

#Axis en 0,0
rgl.lines(c(-1, 1), c(0, 0), c(0, 0), color = "green")
rgl.lines(c(0, 0), c(-1,1), c(0, 0), color = "green")
rgl.lines(c(0, 0), c(0, 0), c(-1,1), color = "green")

#Export to png
rgl.snapshot( 'infomap_clusters.png', fmt = "png", top = TRUE )



# # # # # # # Detecting communicability communities in the reduced communicability space # # # # # # #

#Select the number of clusters
nclu <- 20
n_k <- 1000 #times to train a nclu-cluster training
totwithinss <- rep(0,nclu)
tol <- 0.05 #minimun improvement in totwithinss (% of no clusters situation) required for selecting the

for (i in 1:nclu){ #1 to nclu clusters to try
  totwithinss1 <- rep(0,n_k)
  for (j in 1:n_k){
    communities <- kmeans(coords,i)
    totwithinss1[j]<-communities$tot.withinss
  }
  totwithinss[i] <- mean(totwithinss1)
}

#Vectors of reduction of totwithinss respect to no clusters situation
reduction <- rep(0,nclu)
for (i in 2:(nclu)){
  (totwithinss[1]-totwithinss[i])/totwithinss[1]
  reduction[i] <- (totwithinss[1]-totwithinss[i])/totwithinss[1] - (totwithinss[1]-totwithinss[i-1])/tot
}

plot(totwithinss)
```




```r
#Number of clusters obtained
num_clusters2 <- which(reduction==reduction[reduction<tol & reduction>0][1])
num_clusters2
```

```
## [1] 5
```

```r
n_times <- 10000
solutions<-list()
totwithinss_solutions <- rep(0,n_times)

for (n_t in 1:n_times){
  communities <- kmeans(coords,num_clusters2)
  solutions[[n_t]]<-communities
  totwithinss_solutions[n_t]<-communities$tot.withinss
}

summary(totwithinss_solutions)
```

```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## 0.0002128 0.0002171 0.0002369 0.0002310 0.0002369 0.0003008
```

```r
best<-which(totwithinss_solutions==min(totwithinss_solutions))

#Store the set of best solutions (with the lowest totwithinss)
my_solutions<-list()

for (i in 1:length(best)){
  my_solutions[[i]]<-solutions[[best[i]]]
}
```

```r
#Do the clusters in all the best solutions have the same size?
num_elements_solutions<-sapply(my_solutions, function(x) sort(x$size))
summary(t(num_elements_solutions)) # veo si todos tienen los mismos elementos
```

```
##        V1           V2            V3            V4            V5
##  Min.   :7   Min.   :12   Min.   :13   Min.   :19   Min.   :29
##  1st Qu.:7   1st Qu.:12   1st Qu.:13   1st Qu.:19   1st Qu.:29
##  Median :7   Median :12   Median :13   Median :19   Median :29
##  Mean   :7   Mean   :12   Mean   :13   Mean   :19   Mean   :29
##  3rd Qu.:7   3rd Qu.:12   3rd Qu.:13   3rd Qu.:19   3rd Qu.:29
##  Max.   :7   Max.   :12   Max.   :13   Max.   :19   Max.   :29
```

```r
SDs<- apply(t(num_elements_solutions), 2, sd)
SDs
```

```
## [1] 0 0 0 0 0
```

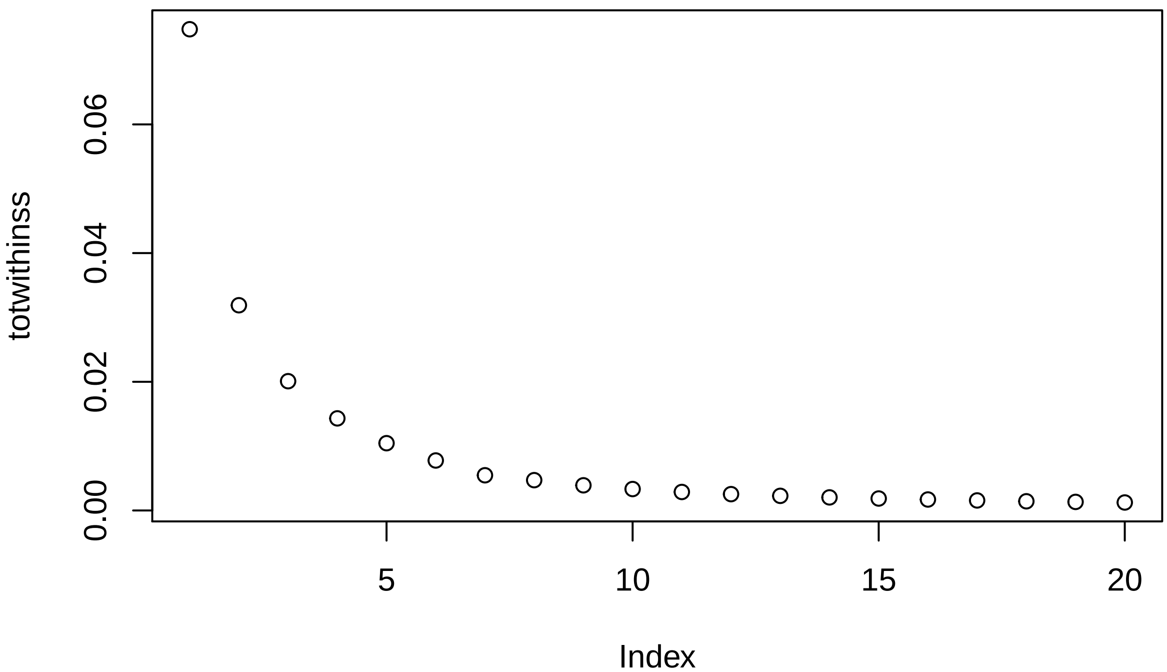
```r
#Since all the solutions have the same totwithinss, I pick one randomly
my_solution<-sample(my_solutions, 1)

#Plot the solution
g <- set.vertex.attribute(g, "group", value=my_solution[[1]]$cluster)
if (length(my_solution[[1]]$size)!=1){
  ColNode <- rainbow(length( my_solution[[1]]$size))[as.numeric(cut(V(g)$group,breaks = length( my_solu
}else{ColNode=rep("#666666",length(sc))}

g <- set.vertex.attribute(g, "color", value=ColNode)

#Node sizes
mns <- 5 #Min node size for plotting
Mns <- 20 #Max node size for plotting
a <- (Mns-mns)/(max(sc)-min(sc))
y0 <- Mns - ((Mns-mns)*max(sc))/(max(sc)-min(sc))

#edge sizes
mes <- 1 #Min edge size for plotting
Mes <- 10 #Max edge size for plotting
ae <- (Mes-mes)/(max(Gpq)-min(Gpq))
y0e <- Mes - ((Mes-mes)*max(Gpq))/(max(Gpq)-min(Gpq))

rgl.open()# Open a new RGL device
par3d("windowRect"= c(0,0,1300,1300))
rgl.bg(color = "white") # Setup the background color
rglplot(g, layout=as.matrix(coords),vertex.color=V(g)$color, vertex.size= y0 + a*V(g)$sc, vertex.label =
        vertex.label.color="black", vertex.label.dist=1, edge.width= y0e + ae*E(g)$Gpq, edge.color=E(g)$
#legend3d("top", legend ="Communicability (reduced) communities", cex=1, bty = "n")
rgl.viewpoint(  zoom = .8 )


#Ejes en 0,0
rgl.lines(c(-1, 1), c(0, 0), c(0, 0), color = "green")
rgl.lines(c(0, 0), c(-1,1), c(0, 0), color = "green")
```

```r
rgl.lines(c(0, 0), c(0, 0), c(-1,1), color = "green")

#Export to png
rgl.snapshot( 'communicability_reduced.png', fmt = "png", top = TRUE )

my_clusters <- data.frame(names=names[,2],my_solution[[1]]$cluster)
my_clusters <- my_clusters[order(my_clusters[2]),]

#Uncoment for saving clusters to Excel file
write.xlsx(my_clusters, paste(myWD,"/my_clusters.xlsx",sep=""))


# Para cada cluster, calculo el centroide, la distancia al centroide de cada nodo, y luego el radio med
n_elements2 <-rep(0, num_clusters2)
centroid2 <-list()
rads2 <- list()
Avg_radius2 <-rep(0, num_clusters2)
pack_index2 <-rep(0, num_clusters2) #Pack_index: Compacticidad Avg_radius/n_elements
max_radius2 <-rep(0, num_clusters2)

for (i in 1:num_clusters2){
  centroid2[[i]] <- colMeans(coords[V(g)$group==i,])
  rads2[[i]] <- sqrt(rowSums((coords[V(g)$group==i,]-centroid2[[i]])^2))
  Avg_radius2[i] <- mean(rads2[[i]])
  n_elements2[i] <- nrow(coords[V(g)$group==i,])
  pack_index2[i] <- Avg_radius2[i] / n_elements2[i]
  max_radius2[i] <- max(rads2[[i]])
}


# # # # # # # Detecting communicability communities in the full communicability space # # # # # # # # #

#Select the number of clusters
nclu <- 20
n_k <- 1000 #times to train a nclu-cluster training
totwithinss <- rep(0,nclu)
tol <- 0.05 #minimun improvement in totwithinss required

for (i in 1:nclu){ #1 to nclu clusters to try
  totwithinss1 <- rep(0,n_k)
  for (j in 1:n_k){
    communities <- kmeans(An,i)
    totwithinss1[j]<-communities$tot.withinss
  }
  totwithinss[i] <- mean(totwithinss1)
}

#Vectors of reduction of totwithinss respect to no clusters situation
reduction <- rep(0,nclu)
for (i in 2:(nclu)){
  (totwithinss[1]-totwithinss[i])/totwithinss[1]
  reduction[i] <- (totwithinss[1]-totwithinss[i])/totwithinss[1] - (totwithinss[1]-totwithinss[i-1])/to
}
```

```r
#Number of clusters obtained
num_clusters3 <- which(reduction==reduction[reduction<tol & reduction>0][1])
num_clusters3
```

```
## [1] 6
```

```r
plot(totwithinss)
```




```r
n_times <- 10000
solutions<-list()
totwithinss_solutions <- rep(0,n_times)

for (n_t in 1:n_times){
  communities <- kmeans(coords,num_clusters3)
  solutions[[n_t]]<-communities
  totwithinss_solutions[n_t]<-communities$tot.withinss
}

summary(totwithinss_solutions)
```

```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## 0.0001588 0.0001597 0.0001997 0.0001862 0.0002118 0.0002445
```

```r
best<-which(totwithinss_solutions==min(totwithinss_solutions))

my_solutions<-list()

for (i in 1:length(best)){
  my_solutions[[i]]<-solutions[[best[i]]]
```

```r
}

#Do the clusters in all the best solutions have the same size?
num_elements_solutions<-sapply(my_solutions, function(x) sort(x$size))
summary(t(num_elements_solutions)) # veo si todos tienen los mismos elementos
```

```
##        V1           V2            V3            V4            V5
##  Min.   :6   Min.   :12   Min.   :12   Min.   :13   Min.   :14
##  1st Qu.:6   1st Qu.:12   1st Qu.:12   1st Qu.:13   1st Qu.:14
##  Median :6   Median :12   Median :12   Median :13   Median :14
##  Mean   :6   Mean   :12   Mean   :12   Mean   :13   Mean   :14
##  3rd Qu.:6   3rd Qu.:12   3rd Qu.:12   3rd Qu.:13   3rd Qu.:14
##  Max.   :6   Max.   :12   Max.   :12   Max.   :13   Max.   :14
##        V6
##  Min.   :23
##  1st Qu.:23
##  Median :23
##  Mean   :23
##  3rd Qu.:23
##  Max.   :23
```

```r
SDs<- apply(t(num_elements_solutions), 2, sd)
SDs
```

```
## [1] 0 0 0 0 0 0
```

```r
#Since all the solutions have the same totwithinss, I pick one randomly
my_solution<-sample(my_solutions, 1)




#Plot the solution

g <- set.vertex.attribute(g, "group", value=my_solution[[1]]$cluster)
if (length(my_solution[[1]]$size)!=1){
  ColNode <- rainbow(length( my_solution[[1]]$size))[as.numeric(cut(V(g)$group,breaks = length( my_solu
}else{ColNode=rep("#666666",length(sc))}
g <- set.vertex.attribute(g, "color", value=ColNode)

#Node sizes
mns <- 5 #Min node size for plotting
Mns <- 20 #Max node size for plotting
a <- (Mns-mns)/(max(sc)-min(sc))
y0 <- Mns - ((Mns-mns)*max(sc))/(max(sc)-min(sc))

#edge sizes
mes <- 1 #Min edge size for plotting
Mes <- 10 #Max edge size for plotting
ae <- (Mes-mes)/(max(Gpq)-min(Gpq))
y0e <- Mes - ((Mes-mes)*max(Gpq))/(max(Gpq)-min(Gpq))
```

```r
rgl.open()# Open a new RGL device
par3d("windowRect"= c(0,0,1300,1300))
rgl.bg(color = "white") # Setup the background color
rglplot(g, layout=as.matrix(coords),vertex.color=V(g)$color, vertex.size= y0 + a*V(g)$sc, vertex.label =
        vertex.label.color="black", vertex.label.dist=1, edge.width= y0e + ae*E(g)$Gpq, edge.color=E(g)$
#legend3d("top", legend ="Communicability (full) communities", cex=1, bty = "n")
rgl.viewpoint(  zoom = .8 )

#Ejes en 0,0
rgl.lines(c(-1, 1), c(0, 0), c(0, 0), color = "green")
rgl.lines(c(0, 0), c(-1,1), c(0, 0), color = "green")
rgl.lines(c(0, 0), c(0, 0), c(-1,1), color = "green")

#Export to png
rgl.snapshot( 'communicability_full.png', fmt = "png", top = TRUE )

my_clusters <- data.frame(names=names[,2],my_solution[[1]]$cluster)
my_clusters <- my_clusters[order(my_clusters[2]),]

write.xlsx(my_clusters, paste(myWD,"/my_clusters_full.xlsx",sep=""))

# Para cada cluster, calculo el centroide, la distancia al centroide de cada nodo, y luego el radio med
n_elements3 <-rep(0, num_clusters3)
centroid3 <-list()
rads3 <- list()
Avg_radius3 <-rep(0, num_clusters3)
pack_index3 <-rep(0, num_clusters3) #Pack_index: Compacticidad Avg_radius/n_elements
max_radius3 <-rep(0, num_clusters3)

for (i in 1:num_clusters3){
  centroid3[[i]] <- colMeans(coords[V(g)$group==i,])
  rads3[[i]] <- sqrt(rowSums((coords[V(g)$group==i,]-centroid3[[i]])^2))
  Avg_radius3[i] <- mean(rads3[[i]])
  n_elements3[i] <- nrow(coords[V(g)$group==i,])
  pack_index3[i] <- Avg_radius3[i] / n_elements3[i]
  max_radius3[i] <- max(rads3[[i]])
}



# # # # # # # Comparison # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # #

#Infomap
length(clusters)
```

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

```r
Avg_radius
```

```
## [1] 0.004392951
```

```r
mean(Avg_radius)
```

```
## [1] 0.004392951
```

```r
n_elements
```

```
## [1] 80
```

```r
pack_index #Pack_index: Avg_radius/n_elements
```

```
## [1] 5.491189e-05
```

```r
mean(pack_index)
```

```
## [1] 5.491189e-05
```

```r
max_radius
```

```
## [1] 0.01211865
```

```r
#Communicability reduced
num_clusters2
```

```
## [1] 5
```

```r
Avg_radius2
```

```
## [1] 0.005128112 0.003594130 0.009721367 0.006439820 0.002391861
```

```r
mean(Avg_radius2)
```

```
## [1] 0.005455058
```

```r
n_elements2
```

```
## [1] 13 19  7 12 29
```

```r
pack_index2 #Pack_index: Compacticidad Avg_radius/n_elements
```

```
## [1] 3.944701e-04 1.891648e-04 1.388767e-03 5.366517e-04 8.247795e-05
```

```r
mean(pack_index2)
```

```
## [1] 0.0005183062
```

```r
max_radius2
```

```
## [1] 0.008689813 0.006152763 0.017174269 0.008216948 0.004045419
```

```r
#Communicability full
num_clusters3
```

```
## [1] 6
```

```r
Avg_radius3
```

```
## [1] 0.002612369 0.006294708 0.004609075 0.001651044 0.007072672 0.013078445
```

```r
mean(Avg_radius3)
```

```
## [1] 0.005886386
```

```r
n_elements3
```

```
## [1] 23 12 14 13 12  6
```

```r
pack_index3 #Pack_index: Compacticidad Avg_radius/n_elements
```

```
## [1] 0.0001135813 0.0005245590 0.0003292197 0.0001270034 0.0005893893
## [6] 0.0021797409
```

```r
mean(pack_index3)
```

```
## [1] 0.0006439156
```

```r
max_radius3
```

```
## [1] 0.004531512 0.008452125 0.006520937 0.003180517 0.009447945 0.016672674
```