```
#run createTheoreticalNetworks.R and createEmpiricalNetworks.r first
#you should have an object called: allNetTheoretical and an object
called allNetEmpirical
#These are graph objects, weighted and directed
#The script creates maps and graphs and write them to graphml files.
These may be viewed in software Gephi.

##create bacbone versions of the graphs
source("R_scripts/backboneExtraction.r")
V(allNetEmpirical)$id<-V(allNetEmpirical)$name
V(allNetTheoretical)$id<-V(allNetTheoretical)$name
freq<-strength(allNetTheoretical,mode = "all")

theoreticalBB<-backboneNetwork(allNetTheoretical,evalFunc = 1, alpha
= 0.001)
V(theoreticalBB)$freq<-strength(allNetTheoretical,mode = "all")
empiricalBB<-backboneNetwork(allNetEmpirical,evalFunc = 1, alpha =
0.001)
V(empiricalBB)$freq<-strength(allNetEmpirical,mode = "all")

####Create a map based on fast-greedy algorithm
fgT<-fastgreedy.community(as.undirected(allNetTheoretical))
fgTBB<-fastgreedy.community(as.undirected(theoreticalBB))
optimal.community(theoreticalBB)
fgE<-fastgreedy.community(as.undirected(allNetEmpirical))
fgEBB<-fastgreedy.community(as.undirected(empiricalBB))

V(theoreticalBB)$fastgreedy<-fgTBB$membership
write.graph(theoreticalBB,"theoretical28052020.graphml",format="grap
hml")

V(empiricalBB)$fastgreedy<-fgEBB$membership
write.graph(empiricalBB,"empirical28052020.graphml",format="graphml"
)

##MAP CREATION
Wij<-function(j,mem,g){
  M<-length(unique(mem))
  m<-which(mem==j)
  dfm<-data.frame()

  for(i in 1:length(m)){
    x<-incident(g,m[i],mode="in")
    a<-ends(g, x,names=F)

    b<-get.edge.attribute(g,name="weight",x)
    c<-mem[a[,1]] #HMMM:..
    dfm<-rbind(dfm,data.frame(a,weight=b,module=c))

  }


  V<-vector()
  for(l in 1:M){
```

```
    V[l]<-sum(dfm$weight[dfm$module==l])
  }

  return(V)

}
makemap<-function(mem,g){
  M<-length(unique(mem))
W<-matrix(NA,ncol=M,nrow=M)
for(j in 1:length(unique(mem))){
  W[,j]<-Wij(j,mem,g) #was W[j,i] - trying out this on June 21st
2020
}
##naming modules based on 10 largest pageranks
pr<-page.rank(g)
modulenames<-vector()
for(k in 1:length(unique(mem))){
  modulenames[k]<-paste(c(k,names(sort(pr$vector[mem==k],decreasing
= T)[1:3])),collapse = ";")
}

internalLinks<-diag(W)
n_words<-as.vector(table(mem))
h<-graph.adjacency(W,mode=c("directed"),weighted = T,diag = F)
V(h)$name<-modulenames #names of the 10 higest pagerank words in
them module
V(h)$id<-modulenames
V(h)$n_words<-n_words #how many words in module
V(h)$internallinks<-internalLinks #how many internal links are in
each module

return(h)
}

mapTheoretical<-makemap(fgTBB$membership,theoreticalBB)
mapEmpirical<-makemap(fgEBB$membership,empiricalBB)


mapTBB<-backboneNetwork(mapTheoretical,0.005,1)
V(mapTBB)$n_words<-V(mapTheoretical)$n_words
mapTBB<-decompose.graph(mapTBB)[[1]]
fgMTBB<-fastgreedy.community(as.undirected(mapTBB))
V(mapTBB)$fastgreedy<-fgMTBB$membership
write.graph(mapTBB,"mapTBB21062020.graphml",format="graphml")

mapEBB<-backboneNetwork(mapEmpirical,0.005,1)
V(mapEBB)$n_words<-V(mapEmpirical)$n_words
mapEBB<-decompose.graph(mapEBB)[[1]]
fgMEBB<-fastgreedy.community(as.undirected(mapEBB))
V(mapEBB)$fastgreedy<-fgMEBB$membership
write.graph(mapEBB,"mapEBB28052020.graphml",format="graphml")

mapTheoreticalrem<-delete.vertices(mapTheoretical,1)
mapTremBB<-backboneNetwork(mapTheoreticalrem,0.005,1)
```

```
V(mapTremBB)$n_words<-V(mapTheoreticalrem)$n_words
mapTremBB<-decompose.graph(mapTremBB)[[1]]
fgMTremBB<-fastgreedy.community(as.undirected(mapTremBB))
V(mapTremBB)$fastgreedy<-fgMTremBB$membership
write.graph(mapTremBB,"mapTBBrem21062020.graphml",format="graphml")

mapEmpiricalrem<-delete.vertices(mapEmpirical,13)
mapEremBB<-backboneNetwork(mapEmpiricalrem,0.005,1)
V(mapEremBB)$n_words<-V(mapEmpiricalrem)$n_words
mapEremBB<-decompose.graph(mapEremBB)[[1]]
fgMEremBB<-fastgreedy.community(as.undirected(mapEremBB))
V(mapEremBB)$fastgreedy<-fgMEremBB$membership
write.graph(mapEremBB,"mapEBB13rem21062020.graphml",format="graphml"
)
```