

Students network

```
library("randnet")
library("sbm")
library("infotheo")
library("entropy")
library("blockmodels")
library("tictoc")
library("kernlab")
library("fossil")
library("gtools")
library(igraph)
library(mclustcomp)

remove(list=ls())

the_wd="/Users/enricoladdomada/Enrico/Tesi_finale" # or the name of the folder where data are stored

setwd(paste(the_wd,"Students_network",sep="/"))

set.seed(1)
source("various_functions.R")

printcompletecommunities=FALSE # if TRUE, prints the list of all students in each community of each metanetwork

y=read.csv("network_directed.csv",header=TRUE,row.names=1,sep=";") # friendship network data as recorded in the original file

sum(colSums(y))

## [1] NA

n=dim(y)[1]
for (i in 1:n){
  for (j in 1:n) {if (is.na(y[i,j])) { y[i,j]=0}} }

indicenomi=read.csv("index.csv",header=TRUE,row.names=1,sep=";") # id, gender and section of the students

# note: the id is in the column named "Sigla" of the original file

rownames(y)=indicenomi$Sigla # giving to the network matrix the good label
colnames(y)=indicenomi$Sigla # giving to the network matrix the good label

y_simmetric=matrix(nrow=n,ncol=n) # y is the directed message network, we compute the undirected network
for (i in 1:n) {
  for (j in 1:n){
```

```

y_simmetric[i,j]=as.numeric(y[i,j]==1 | y[j,i]==1) }}

sum(colSums(y_simmetric))/2

## [1] 1820

z=vectomat(as.integer(as.factor(indicenomi$Sezione)),11) # z contains the linguistic sections of the st
rownames(z)=indicenomi$Sigla
colnames(z)=levels(as.factor(indicenomi$Sezione)) # labelling the linguistic sections

# table of students in terms of gender and sections
atable=table(indicenomi$Sesso,indicenomi$Sezione)

# adding the total column to previous table
atable2=cbind(atable,rowSums(atable))
colnames(atable2)=c(colnames(atable),"Total")

# adding the total row to previous table
atable3=rbind(atable2,colSums(atable2))
rownames(atable3)=c(rownames(atable2),"Total")
print(atable3)

##          DE  ENA  ENB  FI  FRA  FRB  FRC  IT  NL  PT  SW  Total
## F           6   11   7   2   13    5   8   5  11   9   7    84
## M          13    9   7   7   7   11   5  13   9  19  12   112
## Total      19   20  14   9   20   16  13  18  20  28  19   196

# computing the edge probability matrix between linguistic sections

all_friends_matrix=matrix(1,nrow=dim(y)[1],ncol=dim(y)[1])
diag(all_friends_matrix)=rep(0,n)
seztable=t(z)%*%as.matrix(y_simmetric)%*%z
omatrix=t(z)%*%as.matrix(all_friends_matrix)%*%z
diag(omatrix)=diag(omatrix)/2
diag(seztable)=diag(seztable)/2

# the edge probability matrix between linguistic sections
estimated_theta=seztable*(1/omatrix)
print(estimated_theta,digits=2)

##          DE    ENA    ENB    FI    FRA    FRB    FRC    IT    NL    PT    SW
## DE  0.579 0.0447 0.049 0.0409 0.013 0.0197 0.028 0.0468 0.050 0.032 0.055
## ENA 0.045 0.4842 0.275 0.0444 0.040 0.0156 0.023 0.0056 0.070 0.018 0.042
## ENB 0.049 0.2750 0.451 0.0556 0.068 0.0491 0.060 0.0516 0.061 0.051 0.068
## FI  0.041 0.0444 0.056 0.4167 0.028 0.0069 0.026 0.0062 0.100 0.024 0.158
## FRA 0.013 0.0400 0.068 0.0278 0.400 0.1250 0.208 0.0306 0.033 0.054 0.034
## FRB 0.020 0.0156 0.049 0.0069 0.125 0.2333 0.139 0.0035 0.031 0.022 0.036
## FRC 0.028 0.0231 0.060 0.0256 0.208 0.1394 0.513 0.0385 0.062 0.027 0.081
## IT  0.047 0.0056 0.052 0.0062 0.031 0.0035 0.038 0.8039 0.039 0.018 0.029
## NL  0.050 0.0700 0.061 0.1000 0.033 0.0312 0.062 0.0389 0.474 0.046 0.092
## PT  0.032 0.0179 0.051 0.0238 0.054 0.0223 0.027 0.0179 0.046 0.651 0.021
## SW  0.055 0.0421 0.068 0.1579 0.034 0.0362 0.081 0.0292 0.092 0.021 0.678

```

```

z_mergingsubsection=matrix(NA,n,8) # merging the 3 French sections and the 2 english sections in one si
colnames(z_mergingsubsection)=c("DE","EN","FI","FR","IT","NL","PT","SW")
z_mergingsubsection[, "DE"]=z[, "DE"]
z_mergingsubsection[, "FI"]=z[, "FI"]
z_mergingsubsection[, "NL"]=z[, "NL"]
z_mergingsubsection[, "SW"]=z[, "SW"]
z_mergingsubsection[, "PT"]=z[, "PT"]
z_mergingsubsection[, "IT"]=z[, "IT"]
z_mergingsubsection[, "EN"]=z[, "ENA"]+z[, "ENB"]
z_mergingsubsection[, "FR"]=z[, "FRA"]+z[, "FRB"]+z[, "FRC"]

z_to_compare=z_mergingsubsection

type="bernoulli"
verbosity=0
K_alg=20 # maximum number of groups to detect

# all estimations will be gathered in the following list
z_hatss=list()

# list of tested spectral methods, using my labelling convention
the_sp_methods_list=c("RegSP-lap","reg.sp_adjregpar0","reg.Ssp_adjregpar0")

l=length(the_sp_methods_list)+1 # total number of tested methods (spectral methods plus variational-EM

# computing the variational-expectation model
print("variational expectation-maximization")

## [1] "variational expectation-maximization"
mySBM2bis=BM_bernoulli("SBM_sym",adj=as.matrix(y_symmetric),explore_min=1,explore_max=K_alg,verbosity=v
mySBM2bis$plotting=""
mySBM2bis$estimate()

# optimal K according to the variational-expectation maximization model
K_hat=which.max(mySBM2bis$ICL)[1]
print(paste("number of communities:",K_hat))

## [1] "number of communities: 11"
print("-----")

## [1] "-----"
# z probability estimation according to the variational-EM model
z_hatprob2bis=mySBM2bis$memberships[[K_hat]]$Z

# z assignment estimation according to the variational-EM model
z_hat2bisVAREM=matrix(0,nrow=n,ncol=K_hat) # 2 estimations: the first by K_alg number of classes

```

```

for (i in 1:n) {index=which.max(z_hatprob2bis[i,])[1]; z_hat2bisVAREM[i,index]=1}

z_hatss[["VAR-EM"]]=z_hat2bisVAREM

# creating thetabss, ie. the table with the communities estimation, in terms of linguistic sections
thetabss=matrix(nrow=K_hat,ncol=dim(z_to_compare)[2])

# all such tables as thetabss will be computed for all estimator, and put in the list xtables_sections
tables_sections=list()

rownames(thetabss)=paste("Community",1:K_hat)
colnames(thetabss)=colnames(z_to_compare)
for (i in 1:K_hat) {
  thetabss[i,]= sapply(1:dim(z_to_compare)[2],function(k)sum((mattovec(z_hat2bisVAREM)==i)*mattovec(z_t
})

tables_sections[["VAR-EM"]]=thetabss

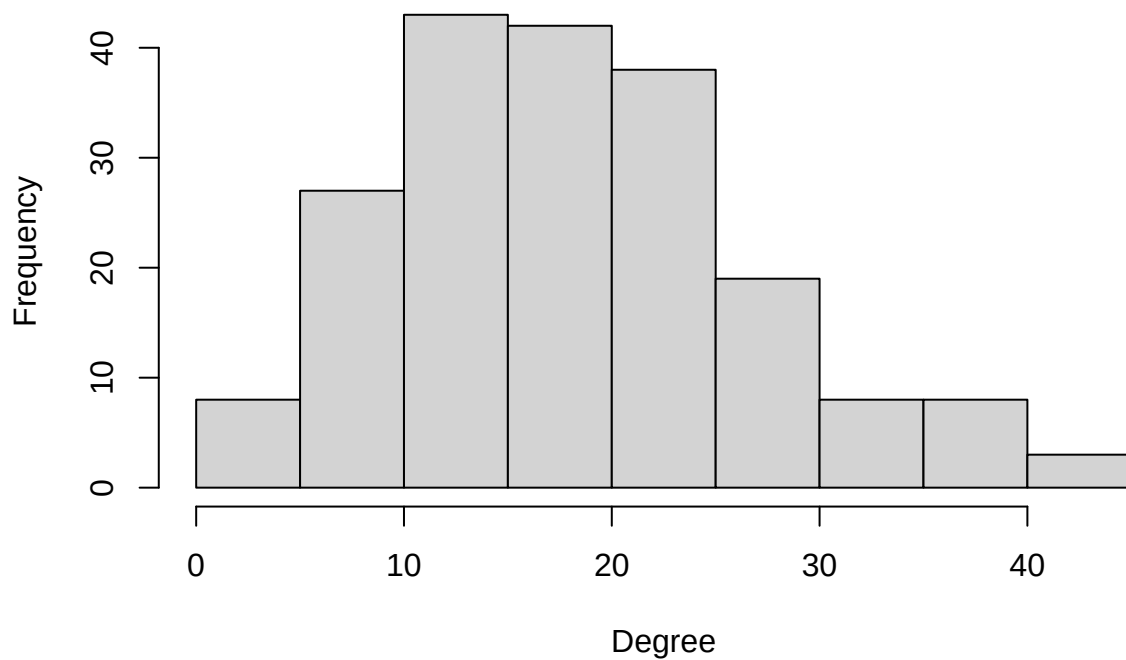
# more explicit names of the tested methods (those above, plus variation expectation-maximization)
goodnames=c("VAR-EM","Laplacian, reg.,"Adjacency","Adjacency, sph.")

yasm=as.matrix(y)

hist(colSums(y_simmetric),main="Degree distribution",xlab="Degree") # degree distribution

```

Degree distribution



```

# loop for each of the methods
for (themethodind in 1:length(the_sp_methods_list)) {

  themethod=the_sp_methods_list[themethodind]

  print(goodnames[themethodind+1])

  # compute all models
  if (themethod=="RegSP-lap") allmodels=lapply(1:K_alg,function(k) reg.SP(yasm,k,lap=TRUE))
  if (themethod=="reg.sp_adjregpar0") allmodels=lapply(1:K_alg,function(k) reg.SP(yasm,k,lap=FALSE,tau=0))

  # for the spherical method, the trivial one-class model must be manually defined
  if (themethod=="reg.Ssp_adjregpar0") {

    allmodels=lapply(1:K_alg,function(k){ if (k>1){

      return(reg.SSP(yasm,k,lap=FALSE,tau=0))} else {
        return(list(cluster=rep(1,n),loss=-Inf))
      })}

    # compute all ICL criteria (log likelihood -penalization) for all models
    alllikelihoods=sapply(1:K_alg,function(k) likelihood_without_penalization_estimator(yasm,vectomat(allmodels[[k]])))

    # select the K with maximum ICL
    thegoodk=which(alllikelihoods==max(alllikelihoods))[1]
    print(paste("number of communities:",thegoodk))
    print("-----")

    # select the optimal model (in terms of ICL) # note: the vectomat function is just a basic function
    z_hat2bis=vectomat(allmodels[[thegoodk]]$cluster,thegoodk)

    rownames(z_hat2bis)=rownames(z)
    rownames(z_hatprob2bis)=rownames(z)

    # just giving another name to the optimal K, to use the same conventions as in other scripts
    K_hat=thegoodk

    z_hatss[[themethod]]=z_hat2bis

    z_hat2bisv=mattovec(z_hat2bis)

    # just specifying the preferred order of the labels of the communities
    comm_order=1:K_hat

    # just using the preferred order of the labels of the communities
    z_hat2bisv_perm=mattovec(z_hat2bis[,comm_order])
  }
}

```

```

# preparing the table with the communities structure in term of linguistic sections
thetabss=matrix(nrow=K_hat,ncol=dim(z_to_compare)[2])
rownames(thetabss)=paste("Community",1:K_hat)
colnames(thetabss)=colnames(z_to_compare)

# for each estimated community, computing the number of students in each linguistic sections, and printing the table

for (i in 1:K_hat) {

  thetabss[i,]= sapply(1:dim(z_to_compare)[2],function(k)sum((z_hat2bisv==i)*mattovec(z_to_compare)==k))
  sapply(1:dim(z_to_compare)[2],function(k)sum((z_hat2bisv==i)*mattovec(z_to_compare)==k))

  atable=(table(indicenomi[indicenomi$Sigla %in% rownames(z_hat2bis)[z_hat2bisv_perm==i],"Sesso"),ind.
  atable2=cbind(atable,rowSums(atable))
  colnames(atable2)=c(colnames(atable),"Total")
  atable3=rbind(atable2,colSums(atable2))
  rownames(atable3)=c(rownames(atable2),"Total")

  if (printcompletecommunities==TRUE) print(indicenomi[indicenomi$Sigla %in% rownames(z_hat2bis)[z_hat2bisv_perm==i],"Sesso"),ind.
}

# the table with all communities

tables_sections[[themethod]]=thetabss

}

## [1] "Laplacian, reg."
## [1] "number of communities: 6"
## [1] "-----"
## [1] "Adjacency"
## [1] "number of communities: 8"
## [1] "-----"
## [1] "Adjacency, sph."
## [1] "number of communities: 7"
## [1] "-----"

## The loop is ended: we now print all the tables in xtables_sections

for (i in c(2,4,3,1)) print(tables_sections[[i]])

##
##      DE EN FI FR IT NL PT SW
## Community 1  0 34  2  1  0  0  0
## Community 2  0  0  0  0 18  0  0
## Community 3 19  0  0  0  0  0  0
## Community 4  0  0  0 46  0  1  0
## Community 5  0  0  7  2  0 19  0
## Community 6  0  0  0  0  0  0 28
##
##      DE EN FI FR IT NL PT SW
## Community 1  0  0  0  1  0 15  0
## Community 2  0  0  0  0 18  0  0
## Community 3  0  0  0 46  0  0  0

```

```

## Community 4 0 0 6 1 0 3 0 19
## Community 5 0 0 0 0 0 0 28 0
## Community 6 19 0 0 0 0 2 0 0
## Community 7 0 34 3 1 0 0 0 0
##
##      DE EN FI FR IT NL PT SW
## Community 1 1 1 3 0 0 18 0 0
## Community 2 0 0 0 0 18 0 0 0
## Community 3 18 0 0 0 0 0 0 0
## Community 4 0 0 1 1 0 1 0 19
## Community 5 0 1 3 47 0 0 1 0
## Community 6 0 32 2 1 0 0 0 0
## Community 7 0 0 0 0 0 0 27 0
## Community 8 0 0 0 0 0 1 0 0
##
##      DE EN FI FR IT NL PT SW
## Community 1 0 0 0 7 0 0 0 2
## Community 2 0 25 1 1 0 0 0 0
## Community 3 0 0 0 0 0 0 28 0
## Community 4 0 0 0 0 18 0 0 0
## Community 5 0 1 4 0 0 12 0 1
## Community 6 4 0 1 0 0 5 0 0
## Community 7 14 0 0 0 0 0 0 0
## Community 8 0 0 0 33 0 0 0 0
## Community 9 0 2 0 8 0 0 0 0
## Community 10 1 6 0 0 0 1 0 3
## Community 11 0 0 3 0 0 2 0 13

# computing the NMI, Rand Index, Adjusted Rand Index and Jaccard index between estimations
my_criteria=c("Normalized Mutual Information","Rand Index","Adjusted Rand Index", "Jaccard Index")
my_criteria_tab=array(NA,dim=c(length(my_criteria),1,1))

dimnames(my_criteria_tab)=list(my_criteria,c("VAR-EM",the_sp_methods_list),c("VAR-EM",the_sp_methods_list))

for (criterium in my_criteria) {
  for (i in 1:1){
    for (j in 1:1){

      if (criterium=="Normalized Mutual Information") my_criteria_tab[criterium,i,j]=my_standMI(z_hatss[i],mat)
      if (criterium=="Rand Index") my_criteria_tab[criterium,i,j]=mclustcomp(mattovec(z_hatss[[i]]),mat)
      if (criterium=="Adjusted Rand Index") my_criteria_tab[criterium,i,j]=mclustcomp(mattovec(z_hatss[[i]]),mat)
      if (criterium=="Jaccard Index") my_criteria_tab[criterium,i,j]=mclustcomp(mattovec(z_hatss[[i]]),mat)

    }
  }
}

dimnames(my_criteria_tab)=list(my_criteria,goodnames,goodnames)

# computing the agreement between all estimations, only for spectral methods estimations. Indeed for the
# a dozen lines of script are needed because the agreement is computed manually and not by the mclustcomp

my_criteria_tab_plus_agreement=array(NA,dim=c(length(my_criteria)+1,1,1))
dimnames(my_criteria_tab_plus_agreement)=list(c(my_criteria,"Agreement"),goodnames,goodnames)

my_criteria_tab_plus_agreement[1:length(my_criteria),,]=my_criteria_tab

```

```

my_criteria_tab_plus_agreement["Agreement",,]=matrix(NA,1,1)

for (i in 2:(1)){
  for (j in 2:(1)){if (i<j) { # (in order to avoid computing twice the agreement, the condition i<j mus

    z1=z_hatss[[i]];
    z2=z_hatss[[j]];

    if (dim(z1)[2]<dim(z2)[2]){
      z1_expanded=cbind(z1,matrix(0,n,-dim(z1)[2]+dim(z2)[2]))
      z2_expanded=z2}

    if (dim(z1)[2]==dim(z2)[2]){

      z1_expanded=z1
      z2_expanded=z2}

    if (dim(z1)[2]>dim(z2)[2]){

      z1_expanded=z1
      z2_expanded=cbind(z2,matrix(0,n,dim(z1)[2]-dim(z2)[2]))}

    my_criteria_tab_plus_agreement["Agreement",i,j]=agreement_anyK(z1_expanded,z2_expanded)$agreement
  }
}}

# computing the agreement between all estimations for i>j, using previous calculations
for (i in 2:(1)){
  for (j in 2:(1)){

    if(j==i) my_criteria_tab_plus_agreement["Agreement",i,j]=1
    if (j<i) my_criteria_tab_plus_agreement["Agreement",i,j]=my_criteria_tab_plus_agreement["Agreement"

# printing the tables of the different estimations similarity, in term of each criterion

for (criterium in c("Normalized Mutual Information")) {

  if (criterium!="Agreement") {print(my_criteria_tab_plus_agreement[criterium,,])

} else {print(my_criteria_tab_plus_agreement[criterium,2:1,2:1])

}

}

```

```

##          VAR-EM Laplacian, reg. Adjacency Adjacency, sph.
## VAR-EM          1.0000000      0.7585038 0.7546998      0.7456295
## Laplacian, reg. 0.7585038      1.0000000 0.8509015      0.9201845

```

```

## Adjacency      0.7546998      0.8509015 1.0000000      0.8469664
## Adjacency, sph. 0.7456295      0.9201845 0.8469664      1.0000000

criterianames=c("NMI","Rand Index","Adjusted Rand Index","Jaccard Index","Agreement")
best_z_hat_criterias=array(dim=c(length(goodnames),1+1),dimnames=list(goodnames,criterianames))

# computing all the estimations quality in terms of the usual criteria (the "true" communities being th

for (best_z_hat_index in 1:l){
  best_z_hat=z_hatss[[best_z_hat_index]]

  best_zhat_criteria=rep(NA,length(my_criteria))
  names(best_zhat_criteria)=my_criteria
  for (criterium in my_criteria) {
    for (i in 1:l){
      for (j in 1:l){
        # computing the NMI, rand index, adj rand index and jaccard index

        if (criterium=="Normalized Mutual Information") best_zhat_criteria[criterium]=my_standMI(best_z_hat,
        if (criterium=="Rand Index")best_zhat_criteria[criterium]=mclustcomp(mattovec(best_z_hat),mattovec(z_hat))
        if (criterium=="Adjusted Rand Index")best_zhat_criteria[criterium]=mclustcomp(mattovec(best_z_hat),mattovec(z_hat))
        if (criterium=="Jaccard Index") best_zhat_criteria[criterium]=mclustcomp(mattovec(best_z_hat),mattovec(z_hat))

      }}}

  # computing the agreement, using the standard function, only for models with less than 10 communities
  the_agreement=NA
  if (max(dim(best_z_hat)[2],dim(z_mergingsubsection)[2])<10 ){
    if (dim(best_z_hat)[2]<dim(z_mergingsubsection)[2]){

      best_z_hat_expanded=cbind(best_z_hat,matrix(0,n,-dim(best_z_hat)[2]+dim(z_mergingsubsection)[2]))
      z_mergingsubsection_expanded=z_mergingsubsection}

    if (dim(best_z_hat)[2]==dim(z_mergingsubsection)[2]){

      best_z_hat_expanded=best_z_hat
      z_mergingsubsection_expanded=z_mergingsubsection}

    if (dim(best_z_hat)[2]>dim(z_mergingsubsection)[2]){

      best_z_hat_expanded=best_z_hat
      z_mergingsubsection_expanded=cbind(z_mergingsubsection,matrix(0,n,dim(best_z_hat)[2]-dim(z_mergingsubsection)[2]))

    the_agreement=agreement_anyK(best_z_hat_expanded,z_mergingsubsection_expanded)

    Agreement=the_agreement$agreement} else {Agreement=NA}

  best_z_hat_criterias[goodnames[best_z_hat_index],]=cbind(t(best_zhat_criteria),Agreement)

```

```

}

if (dim(z_to_compare)[2]==8){

  ## the following 20 lines allow computing the agreement between the var-EM algorithm estimate and the
  ## indeed, by explicitly looking at the estimated communities, the correct label-permutation is the f

  perm=rep(NA,11)

  perm[9]=1
  perm[2]=2
  perm[7]=3
  perm[5]=4
  perm[6]=5
  perm[1]=7
  perm[4]=8
  perm[8]=11
  perm[3]=6
  perm[10]=9
  perm[11]=10

  z_hat=z_hatss[[1]]
  z_hatpermuted=z_hat[,perm]

  best_z_hat_criterias["VAR-EM","Agreement"]=sum(sapply(1:8,function(k) sum( (z_hatpermuted[,k]*z_mergin
}

#print(xtable(best_z_hat_criterias[,c(1,2,5)],caption=paste("Normalized Mututal Information, Rand Index

print(best_z_hat_criterias[,c(1,2,5)])

##              NMI Rand Index Agreement
## VAR-EM      0.7543263  0.9243328 0.7346939
## Laplacian, reg. 0.8714628  0.9455782 0.8367347
## Adjacency     0.8564672  0.9598116 0.9132653
## Adjacency, sph. 0.8836294  0.9660387 0.9132653

number_of_comm=as.matrix(sapply(1:4, function(i) dim(z_hatss[[i]])[2]))
rownames(number_of_comm)=goodnames
colnames(number_of_comm)="Number of communities"

```