Network visualization

##### 1. Executive summary

knitr::opts\_chunk$set(fig.width=14, fig.height=14)

# Load the required packages   
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(network)

## network: Classes for Relational Data  
## Version 1.16.0 created on 2019-11-30.  
## copyright (c) 2005, Carter T. Butts, University of California-Irvine  
## Mark S. Handcock, University of California -- Los Angeles  
## David R. Hunter, Penn State University  
## Martina Morris, University of Washington  
## Skye Bender-deMoll, University of Washington  
## For citation information, type citation("network").  
## Type help("network-package") to get started.

##   
## Attaching package: 'network'

## The following objects are masked from 'package:igraph':  
##   
## %c%, %s%, add.edges, add.vertices, delete.edges, delete.vertices,  
## get.edge.attribute, get.edges, get.vertex.attribute, is.bipartite,  
## is.directed, list.edge.attributes, list.vertex.attributes,  
## set.edge.attribute, set.vertex.attribute

library(sna)

## Loading required package: statnet.common

##   
## Attaching package: 'statnet.common'

## The following object is masked from 'package:base':  
##   
## order

## sna: Tools for Social Network Analysis  
## Version 2.5 created on 2019-12-09.  
## copyright (c) 2005, Carter T. Butts, University of California-Irvine  
## For citation information, type citation("sna").  
## Type help(package="sna") to get started.

##   
## Attaching package: 'sna'

## The following objects are masked from 'package:igraph':  
##   
## betweenness, bonpow, closeness, components, degree, dyad.census,  
## evcent, hierarchy, is.connected, neighborhood, triad.census

library(visNetwork)  
library(threejs)  
library(networkD3)  
library(ggraph)

## Loading required package: ggplot2

##### 2. Importation of the two csv files and the creation of a directed igraph network from these files.

# Load and inspect the datasets  
nodes=read.csv("Fb\_Nodes.csv",header = T)  
head(nodes)

## id group sex friend\_count  
## 1 SE F female 73  
## 2 PH G male 53  
## 3 PD G female 363  
## 4 LN G female 64  
## 5 MB F female 19  
## 6 AP C female 99

links=read.csv("FB\_Links.csv",header = T)  
head(links)

## from to  
## 1 LSF TSL  
## 2 GL TSL  
## 3 SLL TSL  
## 4 CSH TSL  
## 5 DSF TSL  
## 6 CS TSL

# create the igraph object  
net <- graph\_from\_data\_frame(d=links, vertices=nodes, directed=T)   
net

## IGRAPH 4e09799 DN-- 93 323 --   
## + attr: name (v/c), group (v/c), sex (v/c), friend\_count (v/n)  
## + edges from 4e09799 (vertex names):  
## [1] LSF->TSL GL ->TSL SLL->TSL CSH->TSL DSF->TSL CS ->TSL SE ->TSL ALL->LSF  
## [9] GL ->LSF DL ->LSF SLL->LSF ADL->LSF AL ->LSF CSH->LSF AAE->LSF DSF->LSF  
## [17] CS ->LSF SE ->LSF SL ->LSF BCL->LSF BL ->LSF MBM->JLT BB ->JLT AC ->JLT  
## [25] KM ->JLT BS ->JLT SS1->SS2 LBW->SS2 TR ->LRG JLG->LRG DSG->LRG TR ->DSG  
## [33] JLG->TR KM ->MD BS ->MD LL ->MD CWB->MD JK ->LBW JK ->DM RW ->DM   
## [41] NW ->DM TB1->DM JK ->TB1 RW ->TB1 NW ->TB1 SMH->TB1 ME ->TB1 TM ->TB1  
## [49] DSW->TB1 DK ->TB1 JK ->SMH NW ->SMH ME ->SMH TM ->SMH DSW->SMH RBW->SMH  
## [57] DK ->SMH TB2->SMH JK ->SS1 ASS->SS1 JK ->ASS RW ->JK ME ->JK TM ->JK   
## + ... omitted several edges

#### 3. Network attribute inspection as described in task 2

* Inspect the attributes of the network using the E and V function and use the as\_data\_frame function to describe the nodes and links. Plot the network using the default settings and describe the main problem with this plot in terms of its readability

# Use the E and V function to inspect the attributes of the network  
  
E(net) # The edges of the "net" object

## + 323/323 edges from 4e09799 (vertex names):  
## [1] LSF->TSL GL ->TSL SLL->TSL CSH->TSL DSF->TSL CS ->TSL SE ->TSL ALL->LSF  
## [9] GL ->LSF DL ->LSF SLL->LSF ADL->LSF AL ->LSF CSH->LSF AAE->LSF DSF->LSF  
## [17] CS ->LSF SE ->LSF SL ->LSF BCL->LSF BL ->LSF MBM->JLT BB ->JLT AC ->JLT  
## [25] KM ->JLT BS ->JLT SS1->SS2 LBW->SS2 TR ->LRG JLG->LRG DSG->LRG TR ->DSG  
## [33] JLG->TR KM ->MD BS ->MD LL ->MD CWB->MD JK ->LBW JK ->DM RW ->DM   
## [41] NW ->DM TB1->DM JK ->TB1 RW ->TB1 NW ->TB1 SMH->TB1 ME ->TB1 TM ->TB1  
## [49] DSW->TB1 DK ->TB1 JK ->SMH NW ->SMH ME ->SMH TM ->SMH DSW->SMH RBW->SMH  
## [57] DK ->SMH TB2->SMH JK ->SS1 ASS->SS1 JK ->ASS RW ->JK ME ->JK TM ->JK   
## [65] DK ->JK TB2->JK ECW->JK RW ->ECW NW ->RW ME ->RW TM ->RW TB2->RW   
## [73] AC ->MBM BS ->MBM KW ->MBM SE ->LZ ARL->LZ PW ->LZ DZB->LZ CW1->LZ   
## + ... omitted several edges

V(net) # The vertices of the "net" object

## + 93/93 vertices, named, from 4e09799:  
## [1] SE PH PD LN MB AP SM CB TB2 RBW NW AL DB CWB BS AC BW DZM KW   
## [20] VST TM CR AAE HRS LRS SL JJ TBC LE BCL SLL TNB MGE JLG DGT DK WT DSW  
## [39] ME MWB BD LL BB LSM KM SP LCP ADE KH ECS KT GB RBH JU JZ DSM BL   
## [58] CG CSH GGA ADL DL LBF GL CS DSF ALL CW1 DZB PW ARL LZ MBM RW ECW JK   
## [77] ASS SS1 SMH TB1 DM LBW MD TR DSG LRG SS2 JLT LSF TSL LBC LLB DW

V(net)$group # Vertex attribute "group"

## [1] "F" "G" "G" "G" "F" "C" "C" "W" "W" "W" "W" "F" "M" "M" "M" "M" "M" "M" "M"  
## [20] "W" "W" "F" "F" "F" "F" "F" "C" "C" "C" "F" "F" "G" "G" "G" "G" "W" "W" "W"  
## [39] "W" "M" "M" "M" "M" "M" "M" "S" "S" "S" "S" "S" "S" "H" "H" "H" "H" "M" "F"  
## [58] "F" "F" "F" "F" "F" "F" "F" "F" "F" "F" "B" "B" "B" "B" "B" "M" "W" "W" "W"  
## [77] "W" "W" "W" "W" "W" "W" "W" "G" "G" "G" "W" "M" "F" "F" "C" "C" "H"

# Find nodes and edges by attribute:  
# (that returns oblects of type vertex sequence/edge sequence)  
V(net)[group=="B"]

## + 5/93 vertices, named, from 4e09799:  
## [1] CW1 DZB PW ARL LZ

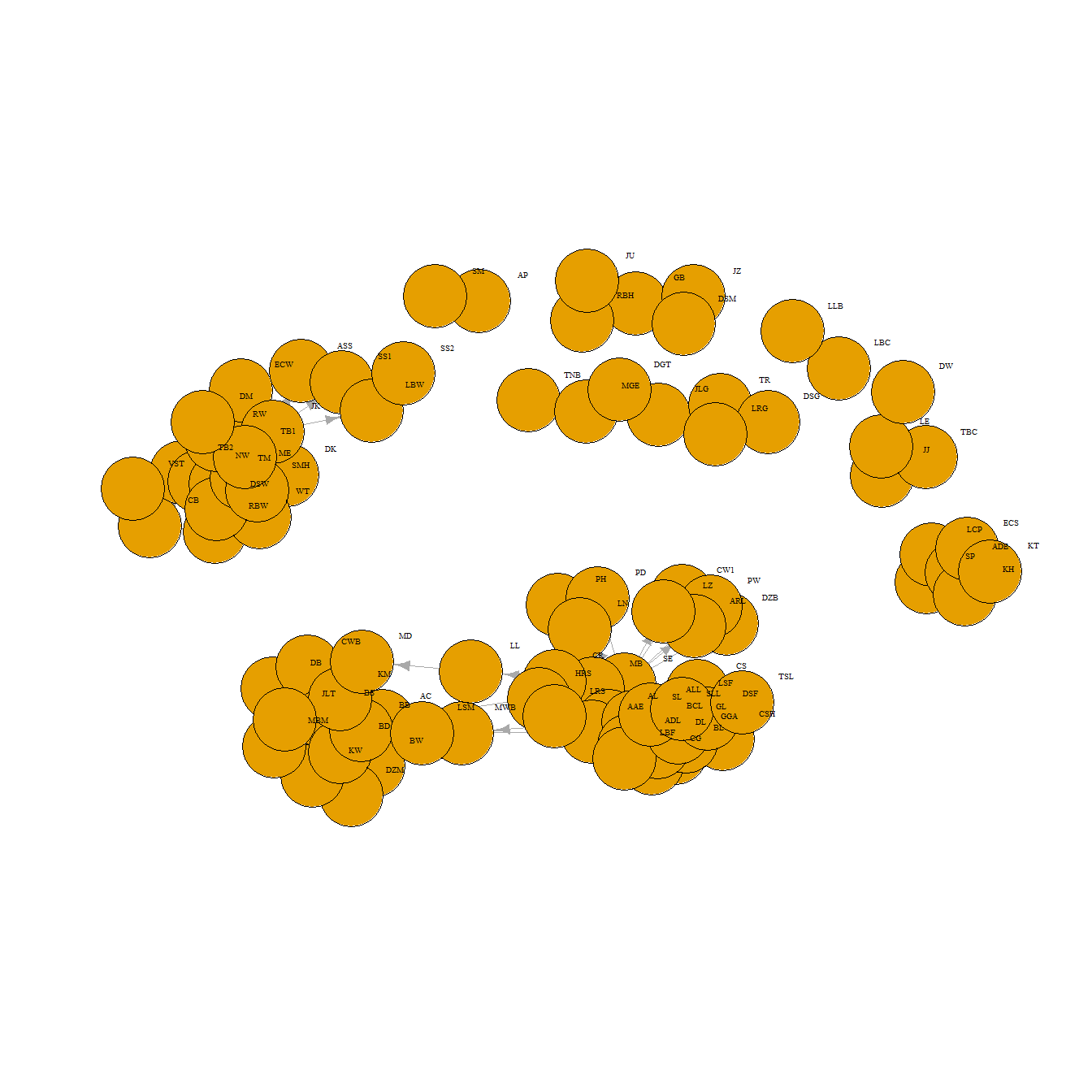
# use the as\_data\_frame function to describe the nodes and link  
df\_edge=as\_data\_frame(net, what="edges")  
head(df\_edge)

## from to  
## 1 LSF TSL  
## 2 GL TSL  
## 3 SLL TSL  
## 4 CSH TSL  
## 5 DSF TSL  
## 6 CS TSL

df\_vertices=as\_data\_frame(net, what="vertices")  
head(df\_vertices)

## name group sex friend\_count  
## SE SE F female 73  
## PH PH G male 53  
## PD PD G female 363  
## LN LN G female 64  
## MB MB F female 19  
## AP AP C female 99

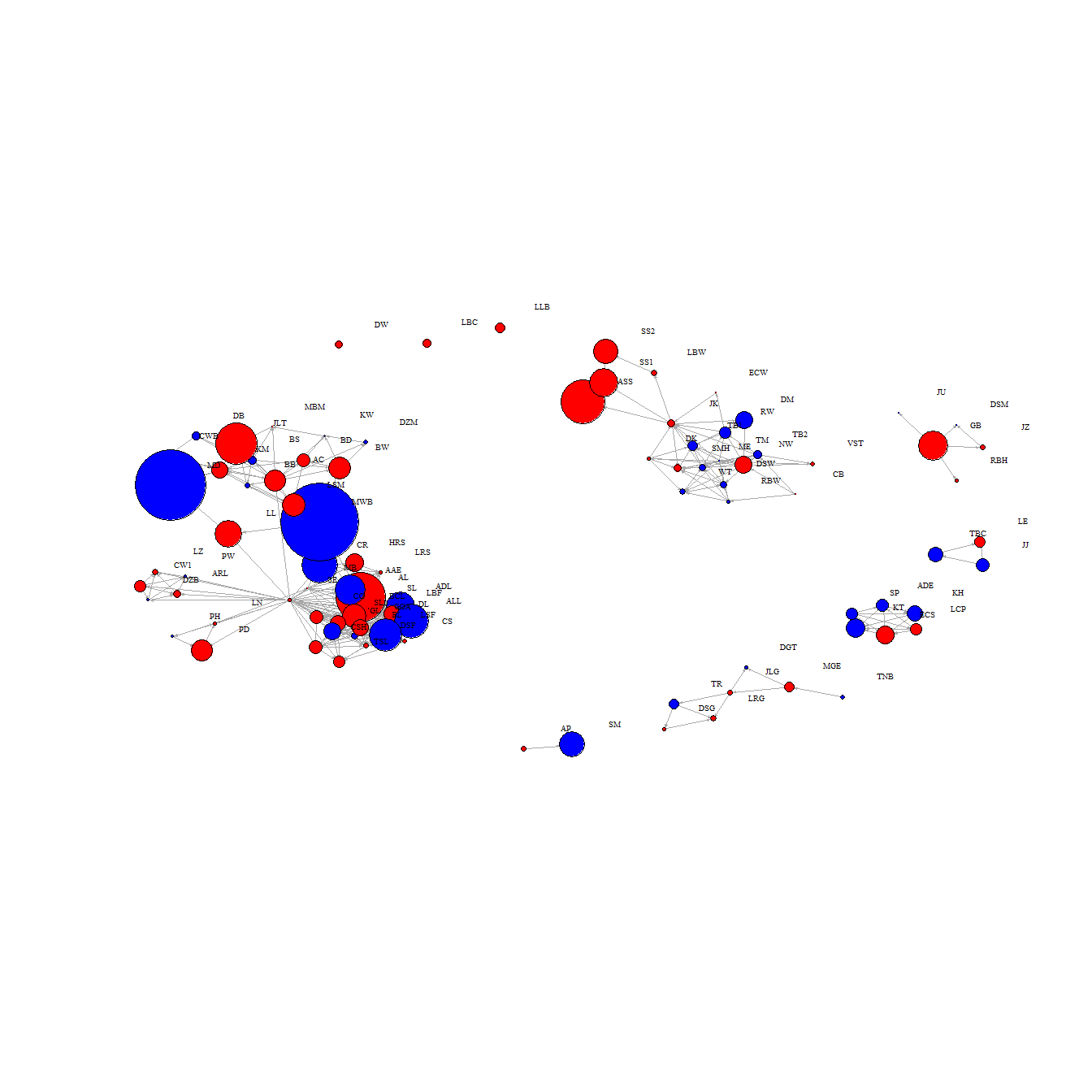
# Plot the network   
plot(net,vertex.label.cex=0.6,vertex.label.dist=2, vertex.label.color="black",asp=0.6)



##### 4. Re-plotting of the network with adjusted attribute settings as described in task 3.

* Plot the network again using the following attributes settings; the colour of the nodes should be blue for male Facebook users and red for female (hint: use the ifelse function - see p23 of the R Studio Student Guide on Moodle), the node size should be set to value of the friend\_count attribute divided by 70 and the arrow.size attribute should be reset to an appropriate value that improves the clarity of the plot

# Generate colors based on media type:  
V(net)$color <- ifelse(V(net)$sex=="male","blue","red")  
  
# We use the friend count to set node size  
V(net)$size <- V(net)$friend\_count/70  
  
#change arrow size   
E(net)$arrow.size <- 0.2  
  
plot(net,layout=layout\_with\_fr,vertex.label.cex=0.6,vertex.label.dist=2, vertex.label.color="black",asp=0.5)  
legend("topright", legend = paste( c('male','female')),  
 pch = 15, inset=c(0.8,-0.3),  
 col = c("blue","red"), cex=0.5)



#### 5. Plotting a series of networks where each plot contains only Facebook users that belong within a particular group, as described in task 4.

* You should plot a total of 8 networks for this task - one network plot for each group. You should adjust the node size to a more appropriate value by dividing the friend\_count value with a smaller number than one used in task 3. Each plot should be displayed using the layout style of layout\_with\_fr.

# Group "B"  
  
library(dplyr)

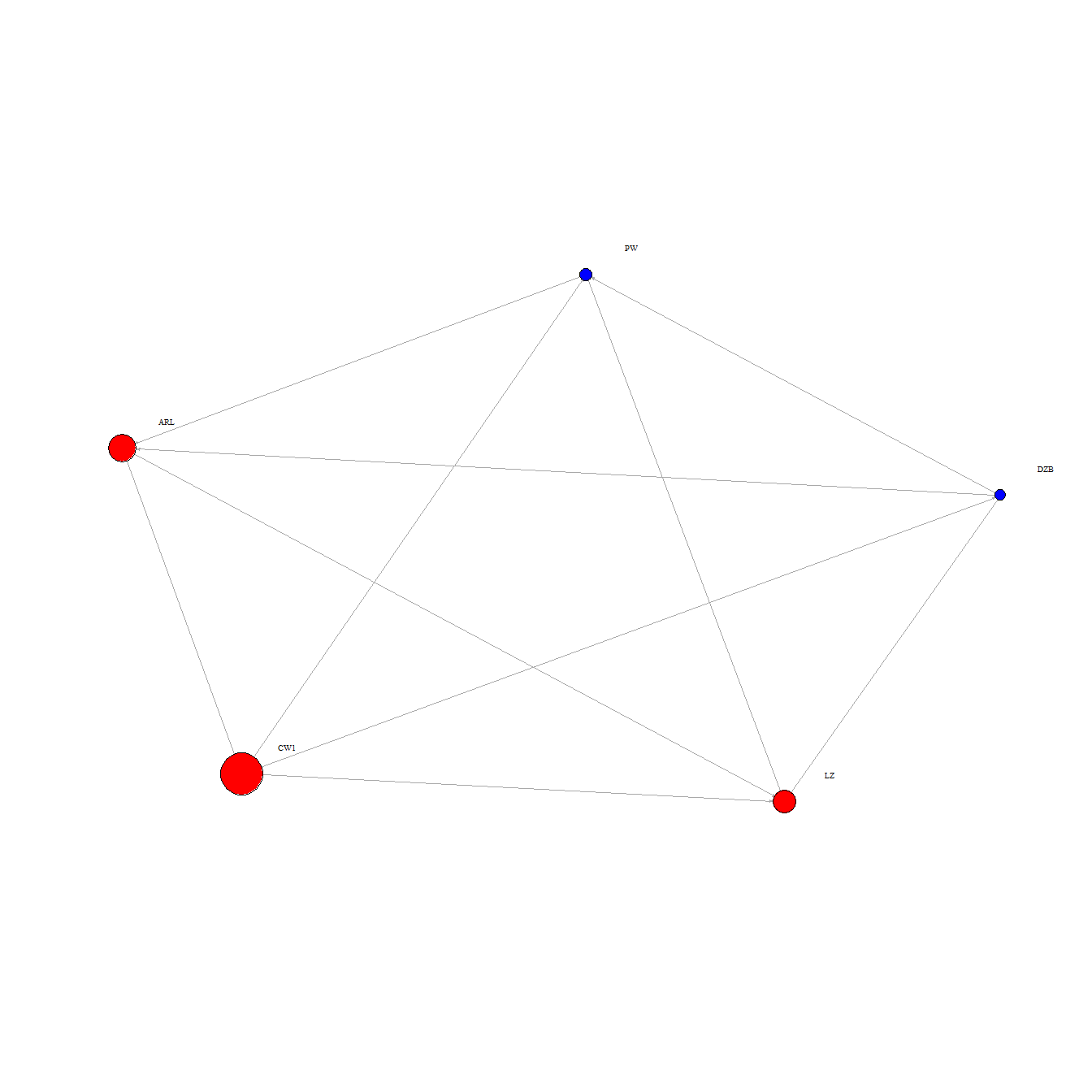
##   
## Attaching package: 'dplyr'

## The following objects are masked from 'package:igraph':  
##   
## as\_data\_frame, groups, union

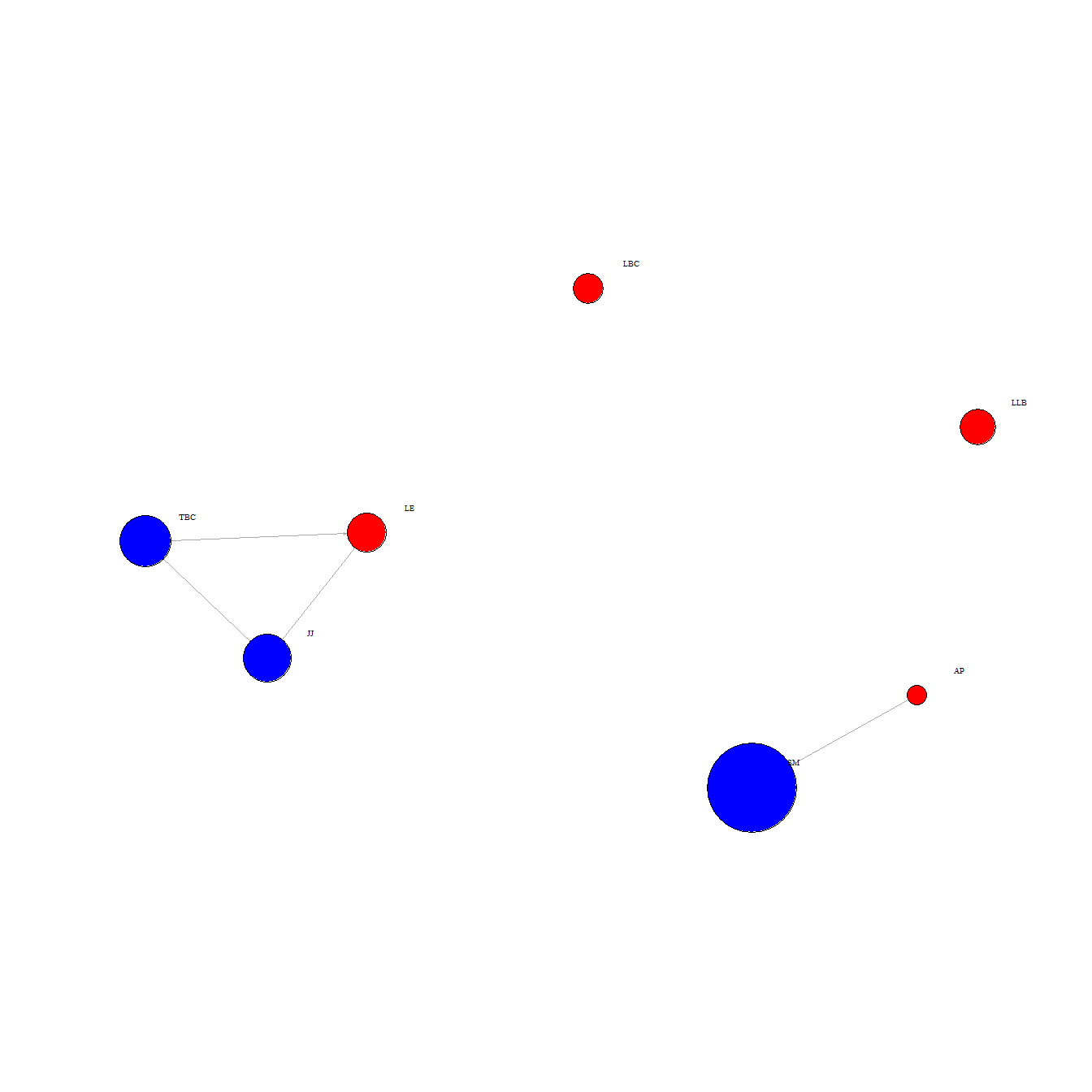
## The following objects are masked from 'package:stats':  
##   
## filter, lag

## The following objects are masked from 'package:base':  
##   
## intersect, setdiff, setequal, union

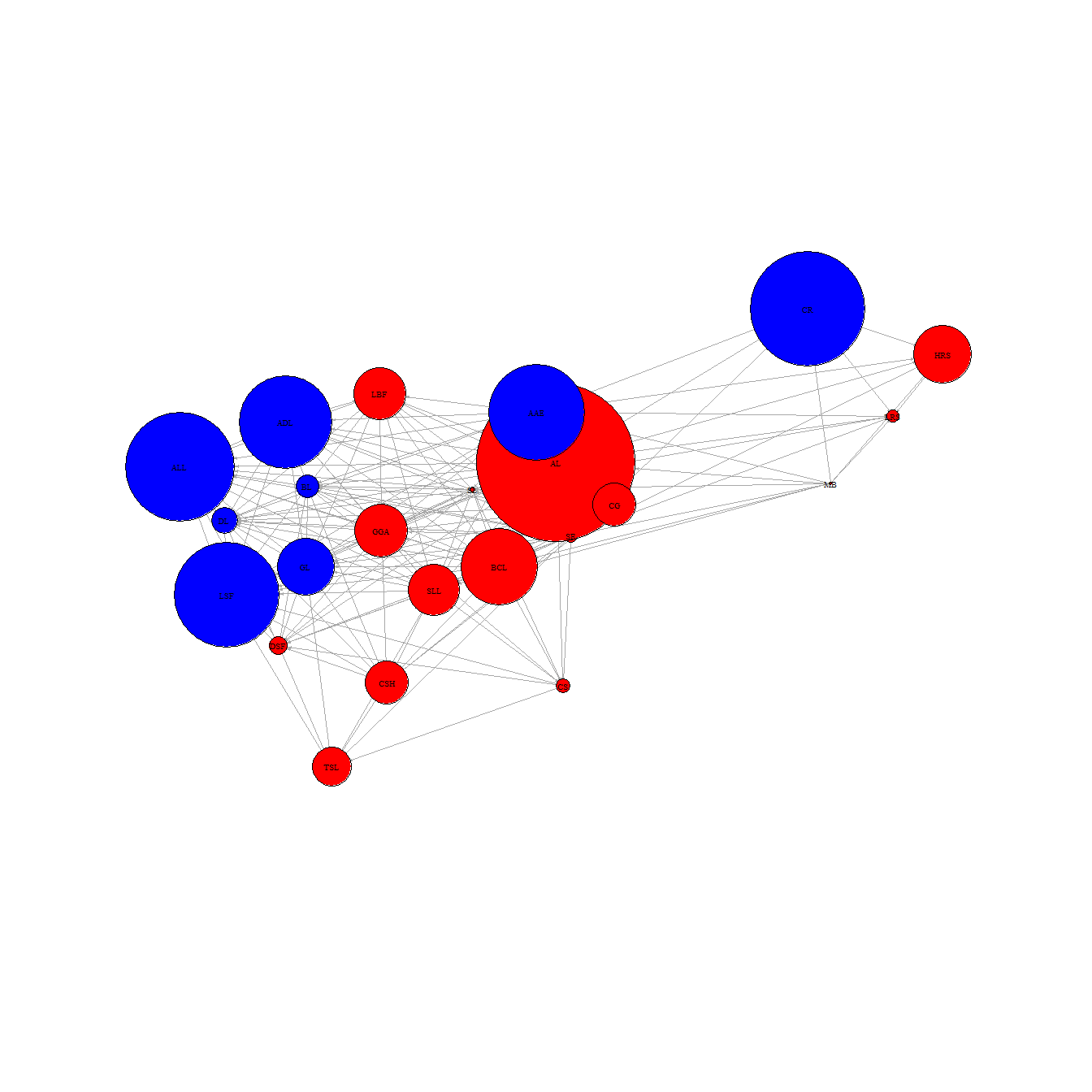
# Filter the nodes and links from group B  
node\_B =nodes%>%   
 group\_by(group) %>%   
 filter("B" %in% group)  
  
names\_B=as.character(node\_B$id)  
links\_B=links[(links$to %in% c(names\_B) & links$from %in% c(names\_B)), ]  
  
net3 <- graph\_from\_data\_frame(d=links\_B, vertices=node\_B, directed=T)   
  
# Male to have color blue and female to have color red  
V(net3)$color <- ifelse(V(net3)$sex=="male","blue","red")  
# We use the friend count to set node size  
V(net3)$size <- V(net3)$friend\_count/20  
  
#change arrow size  
E(net3)$arrow.size <- 0.2  
  
# We set the network layout:  
graph\_attr(net3, "layout") <- layout\_with\_fr  
# Plot the network   
plot(net3,layout=layout\_with\_fr,vertex.label.cex=0.6,vertex.label.dist=2, vertex.label.color="black",asp=0.6)  
legend("topright", legend = paste(c('male','female')),title = "SEX",  
 pch = 15, inset=c(0.8,-0.3),  
 col = c("blue","red"), cex=0.7)



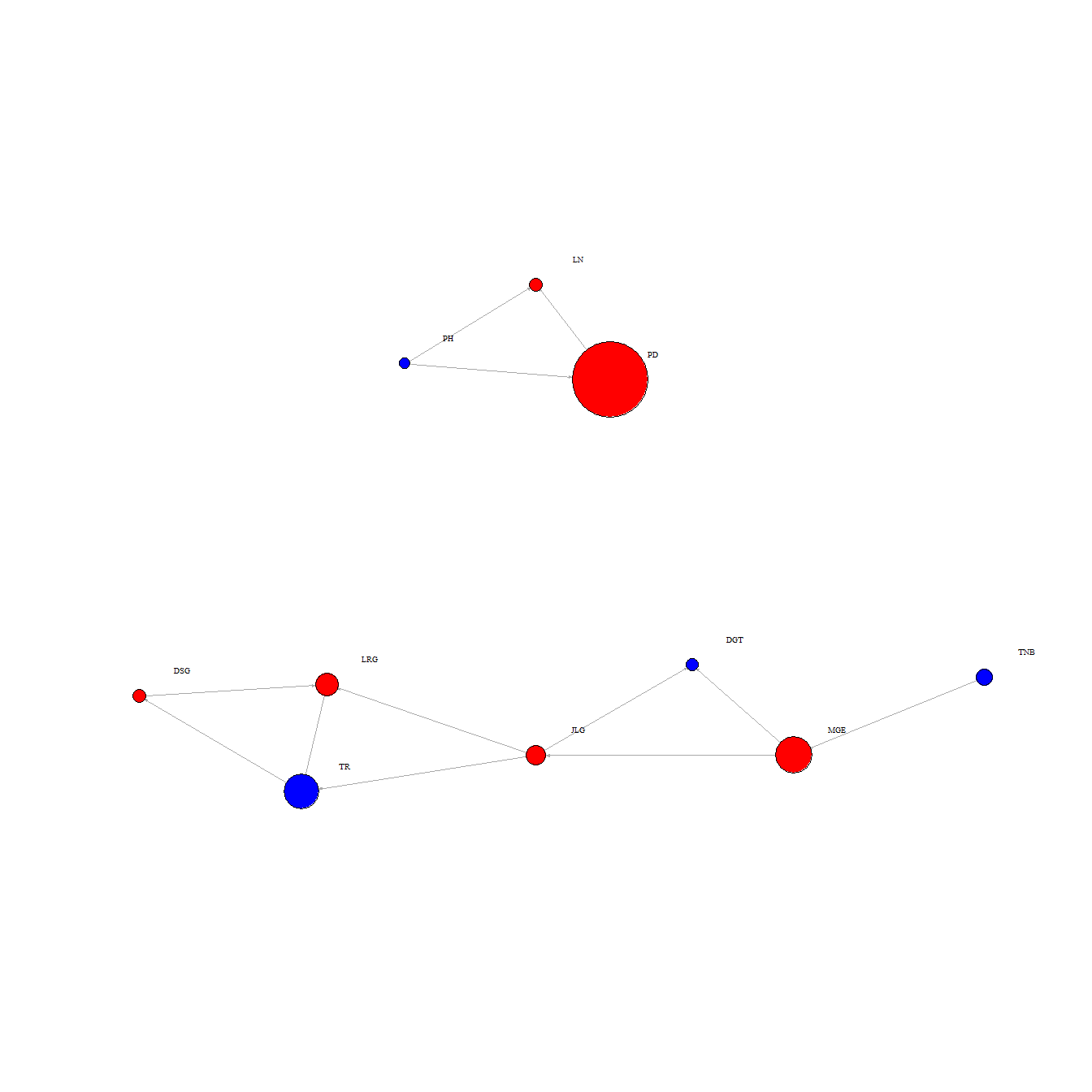
#Group "C"  
library(dplyr)  
  
# Filter the nodes and links from group C  
node\_C =nodes%>%   
 group\_by(group) %>%   
 filter("C" %in% group)  
  
names\_C<-as.character(node\_C$id)  
links\_C<-links[(links$to %in% c(names\_C) & links$from %in% c(names\_C)), ]  
net4 <- graph\_from\_data\_frame(d=links\_C, vertices=node\_C, directed=T)   
  
# Male to have color blue and female to have color red  
V(net4)$color <- ifelse(V(net4)$sex=="male","blue","red")  
# We use the friend count to set node size  
V(net4)$size <- V(net4)$friend\_count/20  
  
#change arrow size   
E(net4)$arrow.size <- 0.2  
  
# We set the network layout:  
graph\_attr(net4, "layout") <- layout\_with\_fr  
# Plot the network   
plot(net4,layout=layout\_with\_fr,vertex.label.cex=0.6,vertex.label.dist=2, vertex.label.color="black",asp=0.6)  
legend("topright", legend = paste(c('MALE','FEMALE')),title = "SEX",  
 pch = 15, inset=c(0.8,-0.2),  
 col = c("blue","red"), cex=0.5)



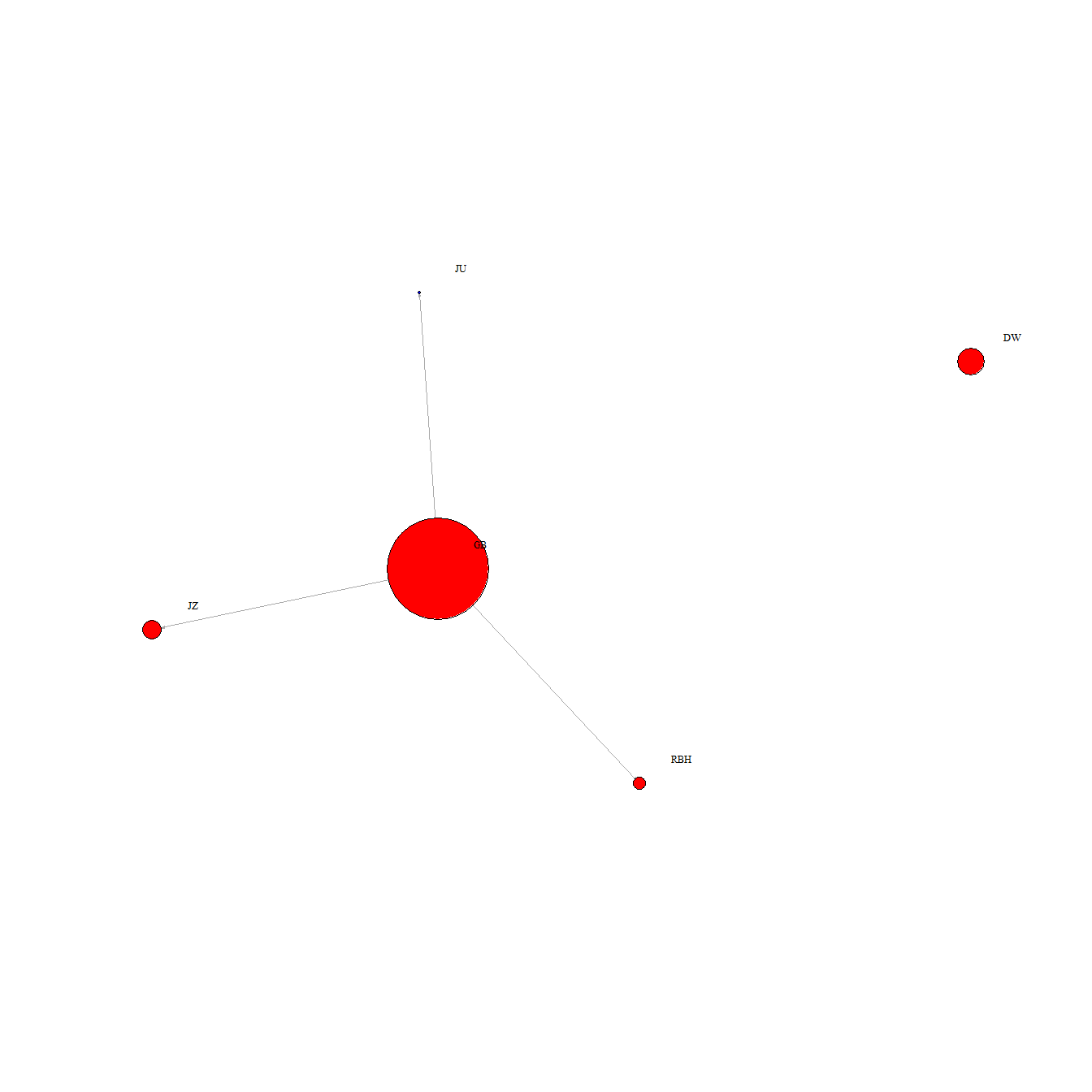
# Group "F"  
library(dplyr)  
  
# Filter the nodes and links from group F  
node\_F =nodes%>%   
 group\_by(group) %>%   
 filter("F" %in% group)  
  
names\_F=as.character(node\_F$id)  
  
links\_F=links[(links$to %in% c(names\_F)  
 & links$from %in% c(names\_F)), ]  
  
net5 <- graph\_from\_data\_frame(d=links\_F, vertices=node\_F, directed=T)   
  
# Male to have color blue and female to have color red  
V(net5)$color <- ifelse(V(net5)$sex=="male","blue","red")  
# We use the friend count to set node size  
V(net5)$size <- V(net5)$friend\_count/20  
  
#change arrow size   
E(net5)$arrow.size <- 0.2  
  
# We set the network layout:  
graph\_attr(net5, "layout") <- layout\_with\_fr  
  
# Plot the network   
# Plot the network   
plot(net5,vertex.label.cex=0.6,vertex.label.color="black",asp=0.6)  
legend("topright", legend = paste(c('MALE','FEMALE')),title = "SEX",  
 pch = 15, inset=c(0.8,-0.2),  
 col = c("blue","red"), cex=0.6)



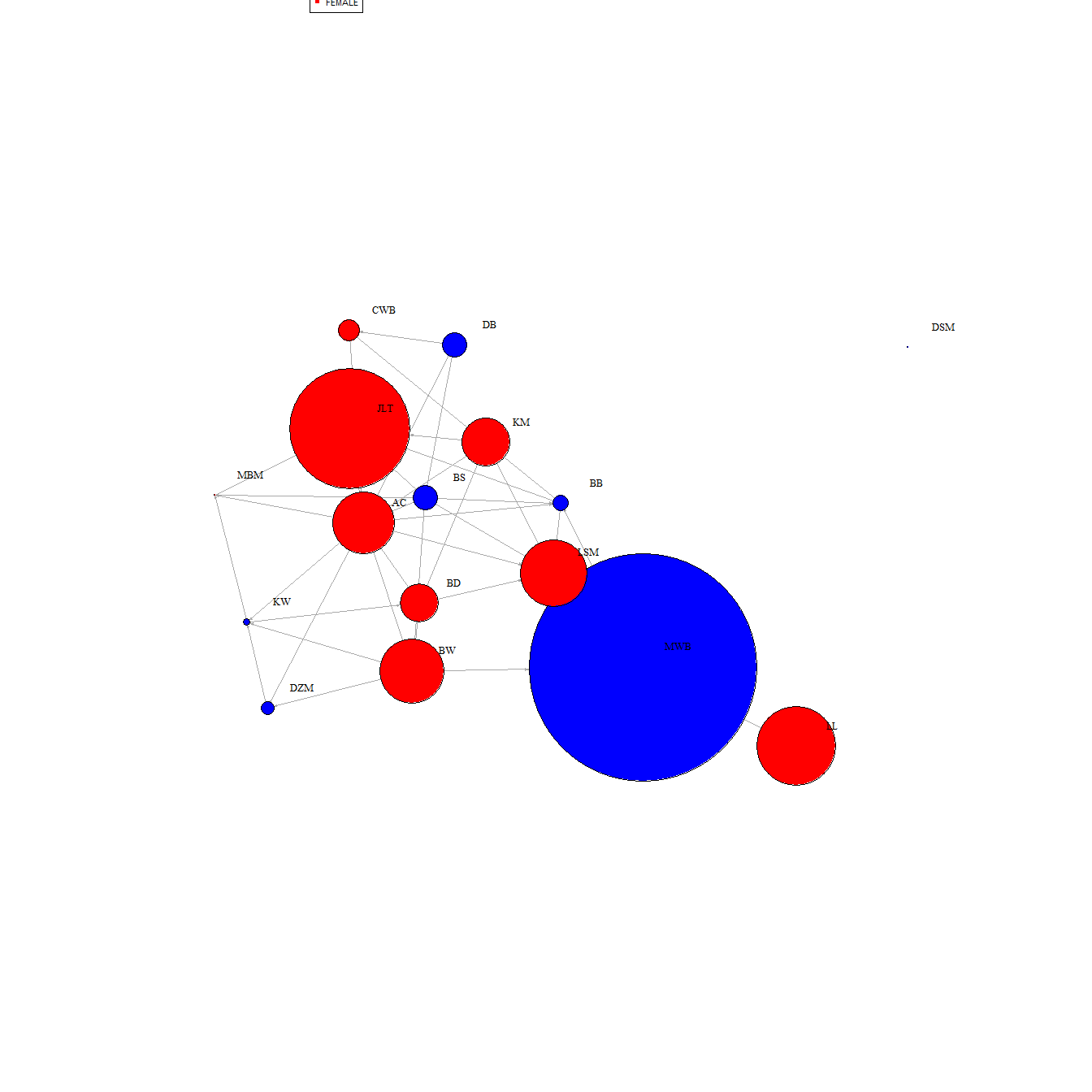
# Group "G"  
library(dplyr)  
  
# Filter the nodes and links from group G  
node\_G = nodes%>%   
 group\_by(group) %>%   
 filter("G" %in% group)  
  
names\_G = as.character(node\_G$id)  
  
links\_G=links[(links$to %in% c(names\_G) & links$from %in% c(names\_G)), ]  
  
net6 <- graph\_from\_data\_frame(d=links\_G, vertices=node\_G, directed=T)   
  
  
# Male to have color blue and female to have color red  
V(net6)$color <- ifelse(V(net6)$sex=="male","blue","red")  
# We use the friend count to set node size  
V(net6)$size <- V(net6)$friend\_count/20  
  
#change arrow size  
E(net6)$arrow.size <- 0.2  
  
# We set the network layout:  
graph\_attr(net6, "layout") <- layout\_with\_fr  
  
# Plot the network   
plot(net6,vertex.label.cex=0.6,vertex.label.dist=2, vertex.label.color="black",asp=0.6)  
legend("topright", legend = paste(c('MALE','FEMALE')),title = "SEX",  
 pch = 15, inset=c(0.8,-0.2),  
 col = c("blue","red"), cex=0.7)



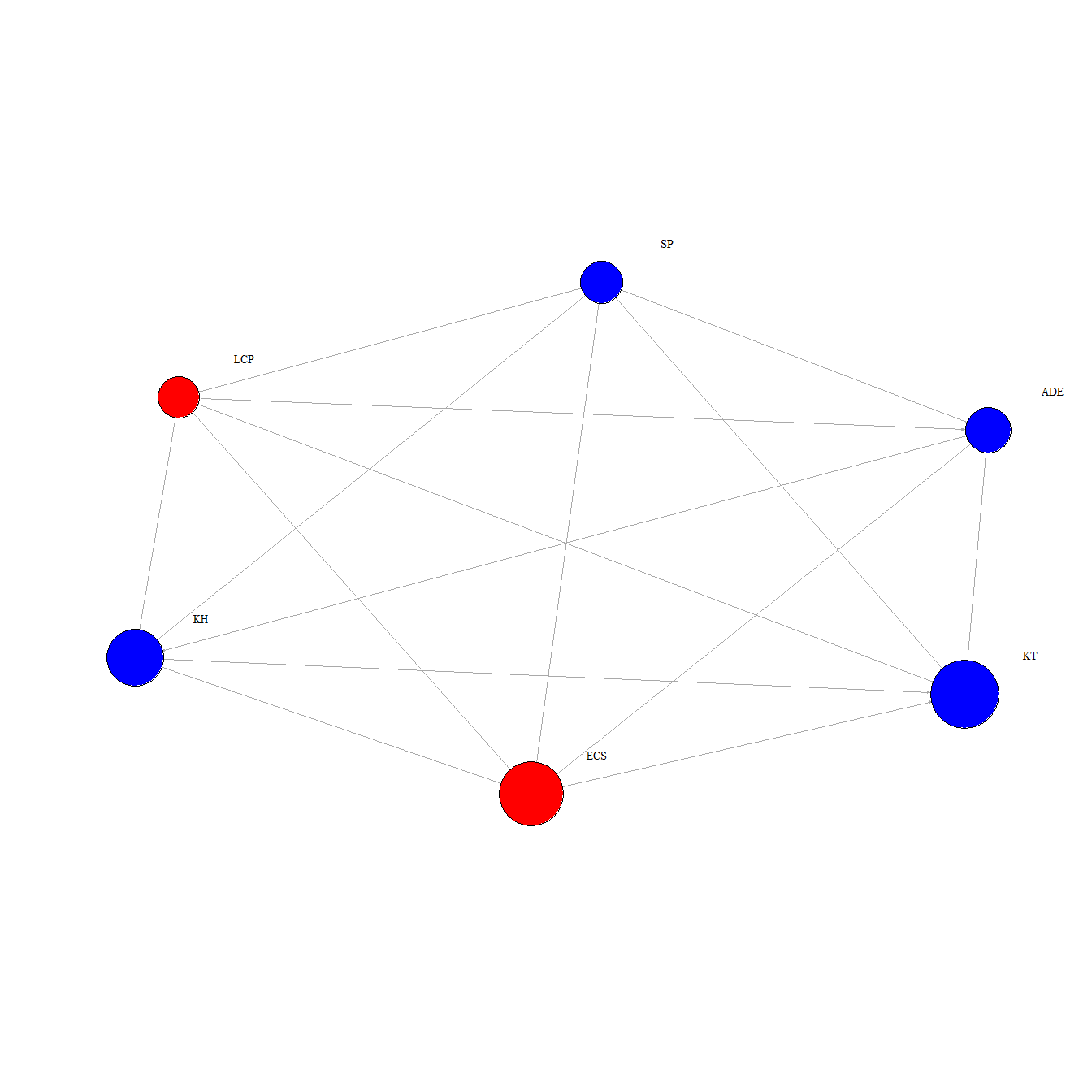
# Group "H"  
library(dplyr)  
  
# Filter the nodes and links from group H  
node\_H = nodes%>%   
 group\_by(group) %>%   
 filter("H" %in% group)  
  
names\_H = as.character(node\_H$id)  
  
links\_H=links[(links$to %in% c(names\_H) & links$from %in% c(names\_H)), ]  
  
net7 <- graph\_from\_data\_frame(d=links\_H, vertices=node\_H, directed=T)   
  
  
# Male to have color blue and female to have color red  
V(net7)$color <- ifelse(V(net7)$sex=="male","blue","red")  
# We use the friend count to set node size  
V(net7)$size <- V(net7)$friend\_count/20  
  
#change arrow size  
E(net7)$arrow.size <- 0.2  
  
# We set the network layout:  
graph\_attr(net7, "layout") <- layout\_with\_fr  
  
  
# Plot the network   
plot(net7,vertex.label.cex=0.8,vertex.label.dist=2, vertex.label.color="black",asp=0.6)  
legend("topright", legend = paste(c('MALE','FEMALE')),title = "SEX",  
 pch = 15, inset=c(0.7,-0.2),  
 col = c("blue","red"), cex=0.7)



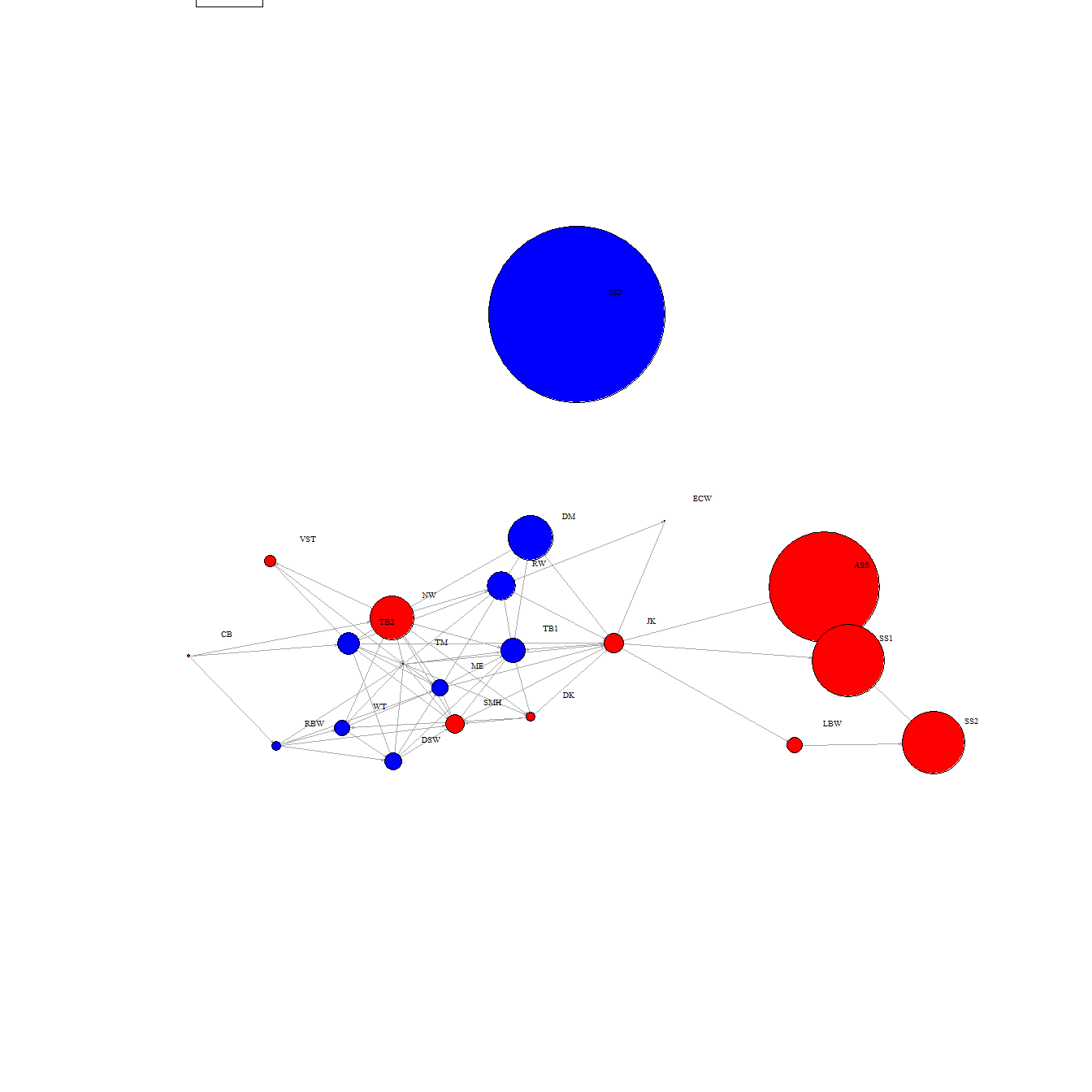
# Group "M"  
library(dplyr)  
  
# Filter the nodes and links from group M  
node\_M = nodes%>%   
 group\_by(group) %>%   
 filter("M" %in% group)  
  
names\_M = as.character(node\_M$id)  
  
links\_M=links[(links$to %in% c(names\_M) & links$from %in% c(names\_M)), ]  
  
net8 <- graph\_from\_data\_frame(d=links\_M, vertices=node\_M, directed=T)   
  
# Male to have color blue and female to have color red  
V(net8)$color <- ifelse(V(net8)$sex=="male","blue","red")  
# We use the friend count to set node size  
V(net8)$size <- V(net8)$friend\_count/20  
  
#change arrow size  
E(net8)$arrow.size <- 0.2  
  
# We set the network layout:  
graph\_attr(net8, "layout") <- layout\_with\_fr  
  
# Plot the network   
  
plot(net8,vertex.label.cex=0.8,vertex.label.dist=2, vertex.label.color="black",asp=0.6)  
legend("topright", legend = paste(c('MALE','FEMALE')),title = "SEX",  
 pch = 15, inset=c(0.7,-0.1),  
 col = c("blue","red"), cex=0.7)



# Group "S"  
library(dplyr)  
  
# Filter the nodes and links from group S  
node\_S = nodes%>%   
 group\_by(group) %>%   
 filter("S" %in% group)  
  
names\_S = as.character(node\_S$id)  
  
links\_S=links[(links$to %in% c(names\_S) & links$from %in% c(names\_S)), ]  
  
net9 <- graph\_from\_data\_frame(d=links\_S, vertices=node\_S, directed=T)   
  
# Male to have color blue and female to have color red  
V(net9)$color <- ifelse(V(net9)$sex=="male","blue","red")  
# We use the friend count to set node size  
V(net9)$size <- V(net9)$friend\_count/20  
  
#change arrow size   
E(net9)$arrow.size <- 0.2  
  
# We set the network layout:  
graph\_attr(net9, "layout") <- layout\_with\_fr  
# Plot the network   
plot(net9,vertex.label.cex=0.9,vertex.label.dist=3, vertex.label.color="black",asp=0.6)  
legend("topright", legend = paste( c('MALE','FEMALE')),  
 pch = 15, inset=c(0.7,-0.2),  
 col = c("blue","red"), cex=0.8)



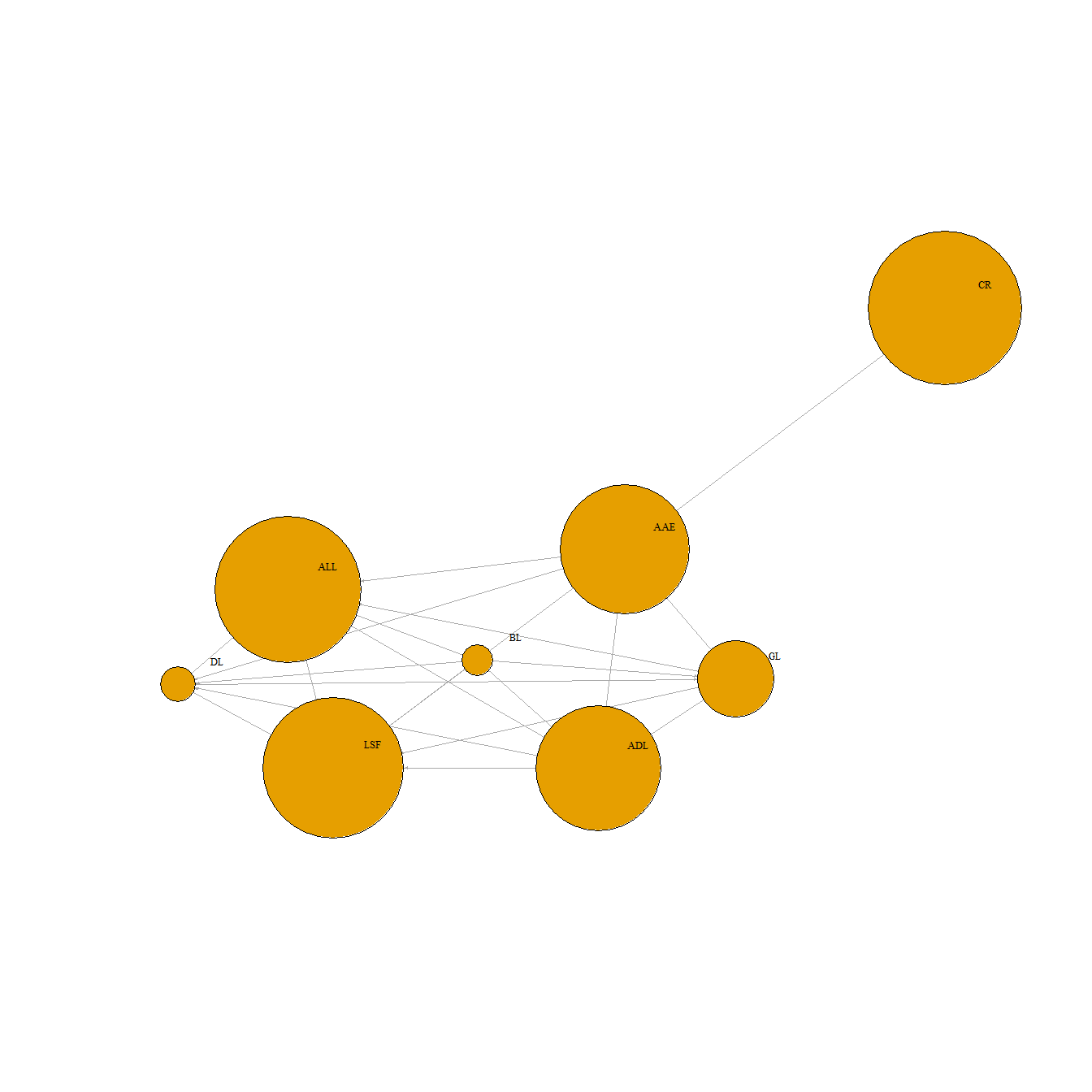
# Group "W"  
library(dplyr)  
  
# Filter the nodes and links from group S  
node\_W = nodes%>%   
 group\_by(group) %>%   
 filter("W" %in% group)  
  
names\_W = as.character(node\_W$id)  
  
links\_W=links[(links$to %in% c(names\_W) & links$from %in% c(names\_W)), ]  
  
net10 <- graph\_from\_data\_frame(d=links\_W, vertices=node\_W, directed=T)   
  
  
# Male to have color blue and female to have color red  
V(net10)$color <- ifelse(V(net10)$sex=="male","blue","red")  
# We use the friend count to set node size  
  
V(net10)$size <- V(net10)$friend\_count/25  
  
#change arrow size   
E(net10)$arrow.size <- 0.2  
  
# We set the network layout:  
graph\_attr(net10, "layout") <- layout\_with\_fr  
# Plot the network   
  
plot(net10,vertex.label.cex=0.6,vertex.label.dist=2, vertex.label.color="black",asp=0.6)  
legend("topright", legend = paste(c('MALE','FEMALE')),  
 pch = 15, inset=c(0.8,-0.1),  
 col = c("blue","red"), cex=0.8)



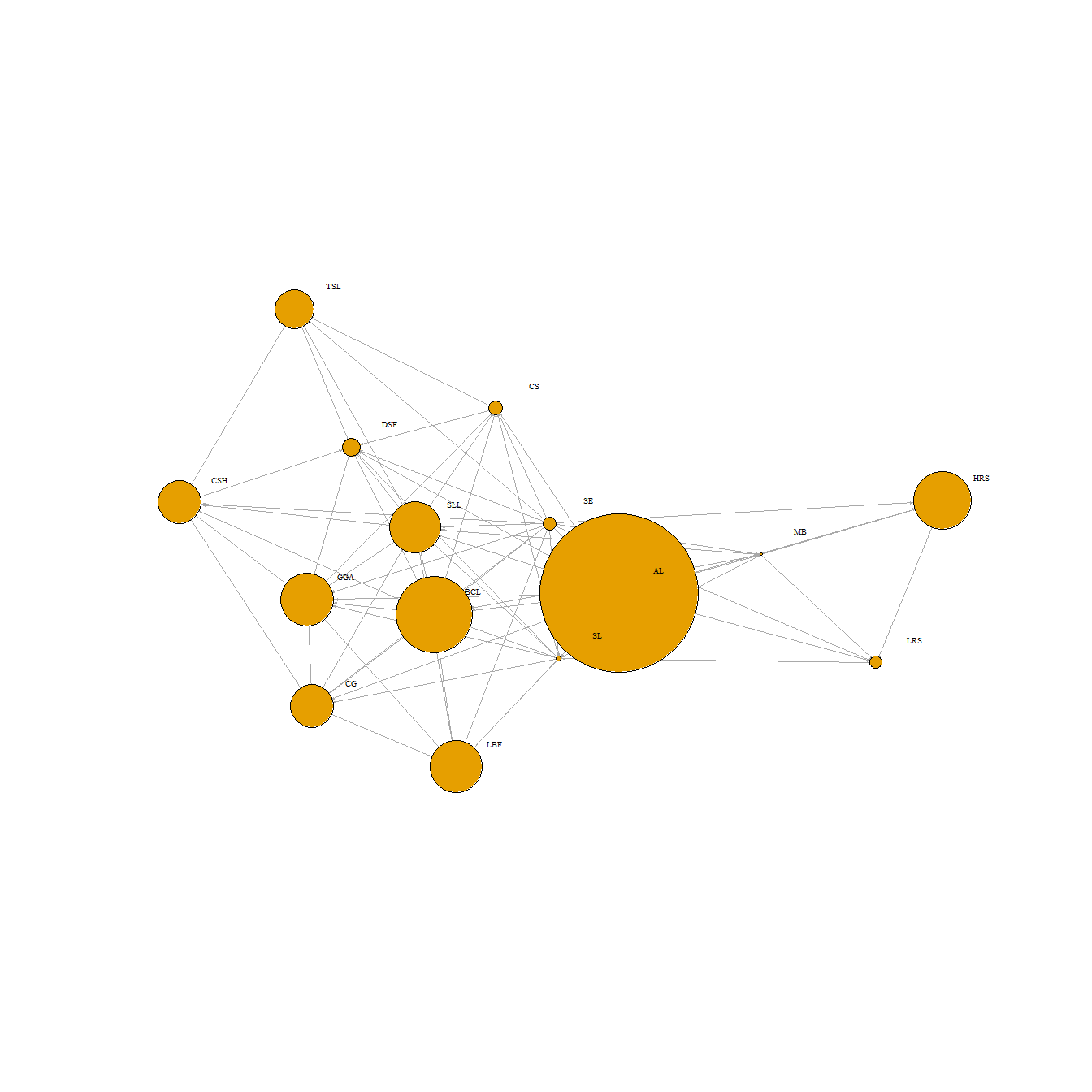
##### 6. Re-plotting the network containing group F Facebook users by gender, as described in task 5.

* Re-plot the network containing group F Facebook users so that only males within that group are displayed. Perform the same task again, but this time only plot the females in this group. Both plots should be displayed using the layout style of layout\_with\_fr.

# Filter the nodes and links from sex "male"  
node\_F\_male =node\_F%>%   
 group\_by(sex) %>%   
 filter("male" %in% sex)  
  
names\_F\_male=as.character(node\_F\_male$id)  
  
links\_F\_male=links[(links$to %in% c(names\_F\_male)  
 & links$from %in% c(names\_F\_male)), ]  
  
net.male <- graph\_from\_data\_frame(d=links\_F\_male, vertices=node\_F\_male, directed=T)   
  
  
# We use the friend count to set node size  
V(net.male)$size <- V(net.male)$friend\_count/15  
  
#change arrow size   
E(net.male)$arrow.size <- 0.2  
  
# We set the network layout:  
graph\_attr(net.male, "layout") <- layout\_with\_fr  
  
# Plot the network   
plot(net.male,vertex.label.cex=0.8,vertex.label.dist=2,  
 vertex.label.color="black",asp=0.6)



# Filter the nodes and links from sex "female"  
node\_F\_female =node\_F%>%   
 group\_by(sex) %>%   
 filter("female" %in% sex)  
  
names\_F\_female=as.character(node\_F\_female$id)  
  
links\_F\_female=links[(links$to %in% c(names\_F\_female)  
 & links$from %in% c(names\_F\_female)), ]  
  
net.female <- graph\_from\_data\_frame(d=links\_F\_female, vertices=node\_F\_female, directed=T)  
  
# We use the friend count to set node size  
V(net.female)$size <- V(net.female)$friend\_count/20  
  
#change arrow size   
E(net.female)$arrow.size <- 0.2  
  
# We set the network layout:  
graph\_attr(net.female, "layout") <- layout\_with\_fr  
# Plot the network   
plot(net.female,vertex.label.cex=0.6,vertex.label.dist=2, vertex.label.color="black",asp=0.6)



##### 7. Detection of communities in each plot created in task 5, as described in task 6.

* Using the cluster\_optimal function, detect the communities in each plot you created in task 5. Both plots should be displayed using the layout style of layout\_with\_fr.

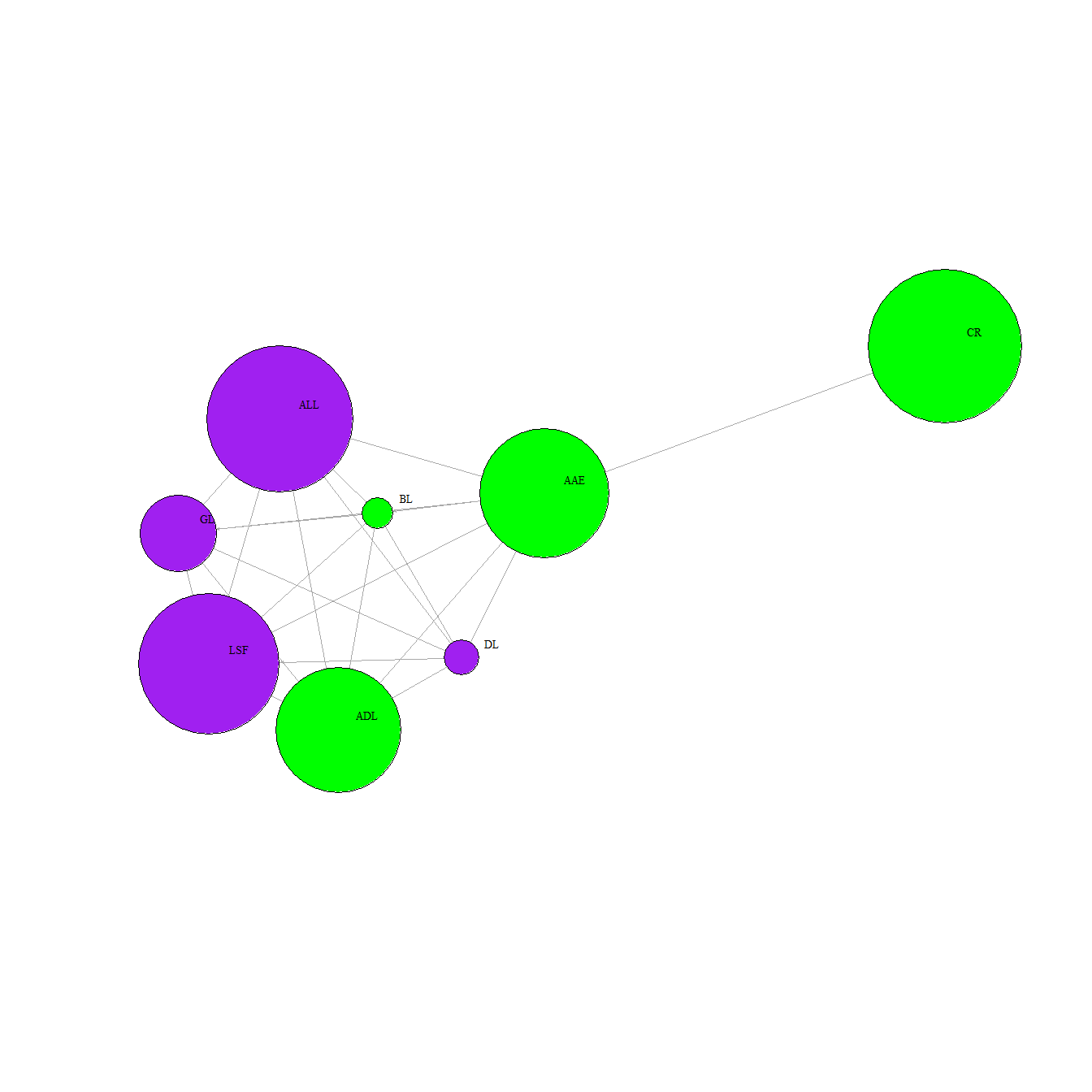
library(igraph)  
# For Male plot  
oc <- cluster\_optimal(net.male)  
oc

## IGRAPH clustering optimal, groups: 2, mod: 0.099  
## + groups:  
## $`1`  
## [1] "CR" "AAE" "BL" "ADL"  
##   
## $`2`  
## [1] "DL" "GL" "ALL" "LSF"  
##

oc$membership #the clusters that will be attached to the dataframe Node\_F\_male

## [1] 1 1 1 1 2 2 2 2

node\_F\_male$membership = oc$membership  
  
net.male.new <- graph\_from\_data\_frame(d=links\_F\_male, vertices=node\_F\_male, directed=T)   
  
  
  
# We use the friend count to set node size  
V(net.male.new)$size <- V(net.male.new)$friend\_count/15  
V(net.male.new)$color <- ifelse(V(net.male.new)$membership==1,"green","purple")  
  
  
#change arrow size   
E(net.male.new)$arrow.size <- 0.2  
  
# We set the network layout:  
graph\_attr(net.male.new, "layout") <- layout\_with\_fr  
  
# Plot the network   
  
plot(net.male.new,vertex.label.cex=0.9,vertex.label.dist=1.5,  
 vertex.label.color="black",asp=0.5)  
legend("topright", legend = paste('community', c('1','2')),  
 pch = 15, inset=c(0.7,-0.2),  
 col = c("green","purple"), cex=0.8)



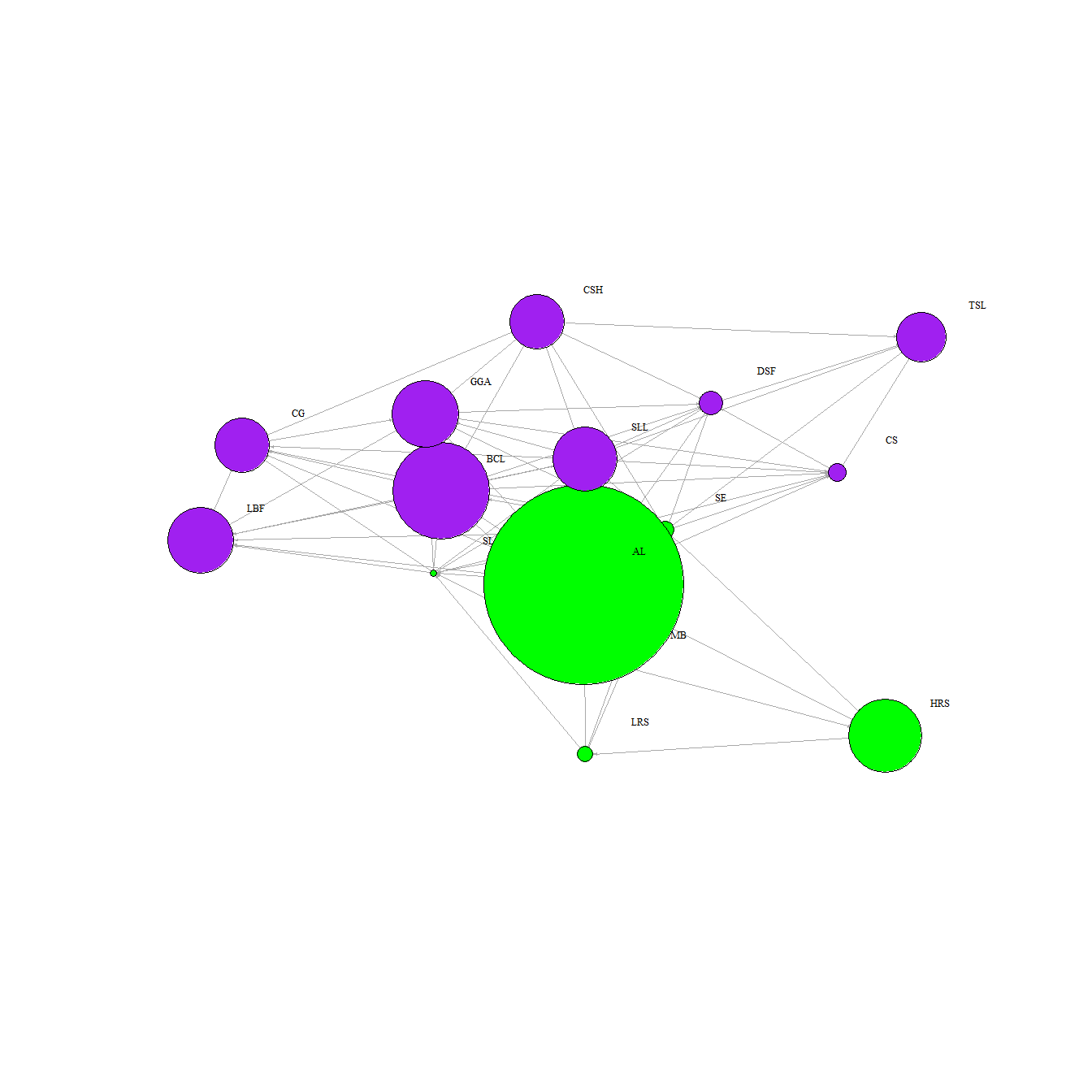
# Female plot  
oc2 <- cluster\_optimal(net.female)  
oc2

## IGRAPH clustering optimal, groups: 2, mod: 0.17  
## + groups:  
## $`1`  
## [1] "SE" "MB" "AL" "HRS" "LRS" "SL"   
##   
## $`2`  
## [1] "BCL" "SLL" "CG" "CSH" "GGA" "LBF" "CS" "DSF" "TSL"  
##

oc2$membership #the clusters that will be attached to the dataframe Node\_F\_male

## [1] 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2

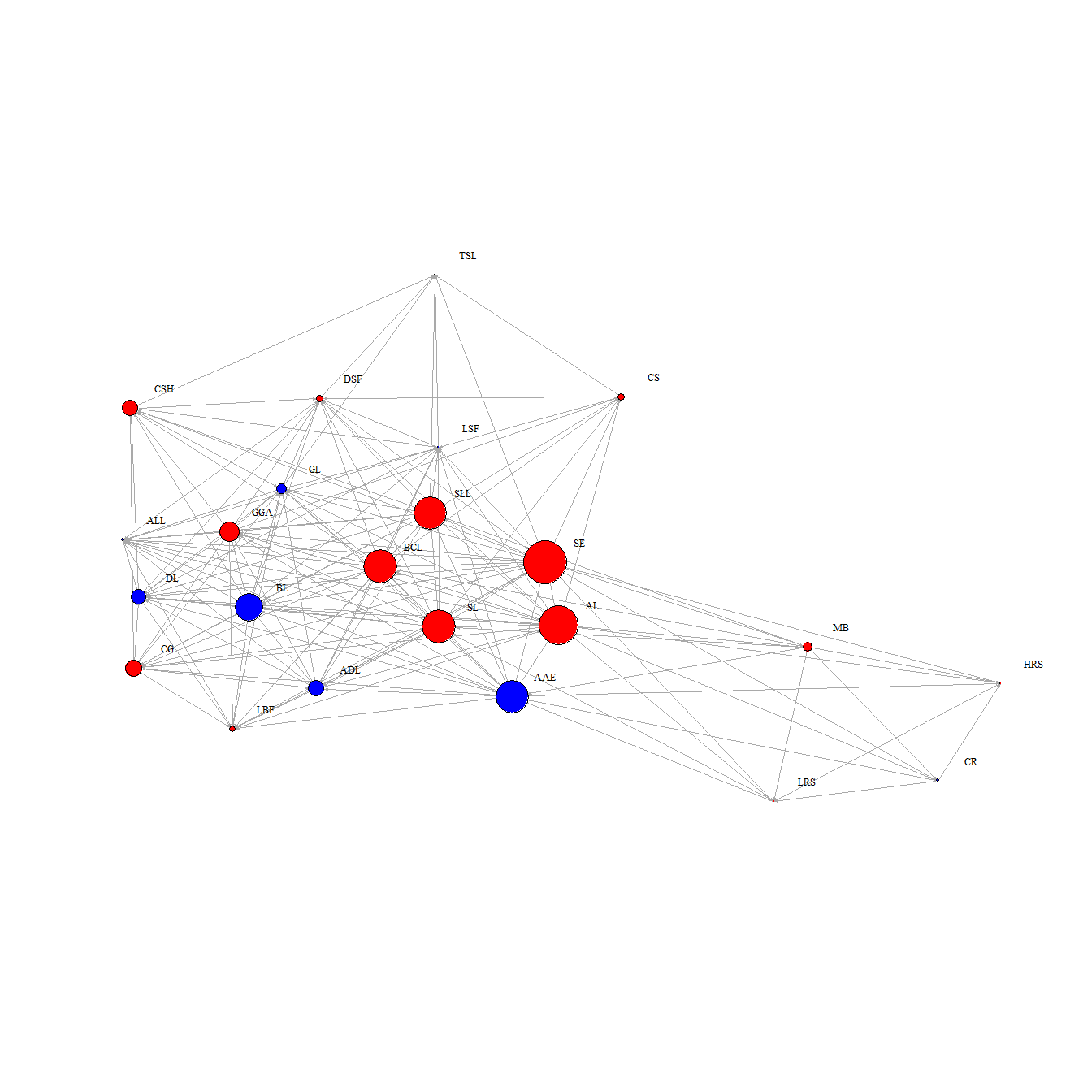
node\_F\_female$membership = oc2$membership  
  
net.female.new <- graph\_from\_data\_frame(d=links\_F\_female, vertices=node\_F\_female, directed=T)   
  
  
# We use the friend count to set node size  
V(net.female.new)$size <- V(net.female.new)$friend\_count/15  
V(net.female.new)$color <- ifelse(V(net.female.new)$membership==1,"green","purple")  
  
  
#change arrow size   
E(net.female.new)$arrow.size <- 0.2  
  
# We set the network layout:  
graph\_attr(net.female.new, "layout") <- layout\_with\_fr  
  
# Plot the network   
  
plot(net.female.new,vertex.label.cex=0.8,vertex.label.dist=3,   
 vertex.label.color="black",asp=0.6)  
legend("topright", legend = paste('community', c('1','2')),  
 pch = 15, inset=c(0.8,-0.2),  
 col = c("green","purple"), cex=0.7)



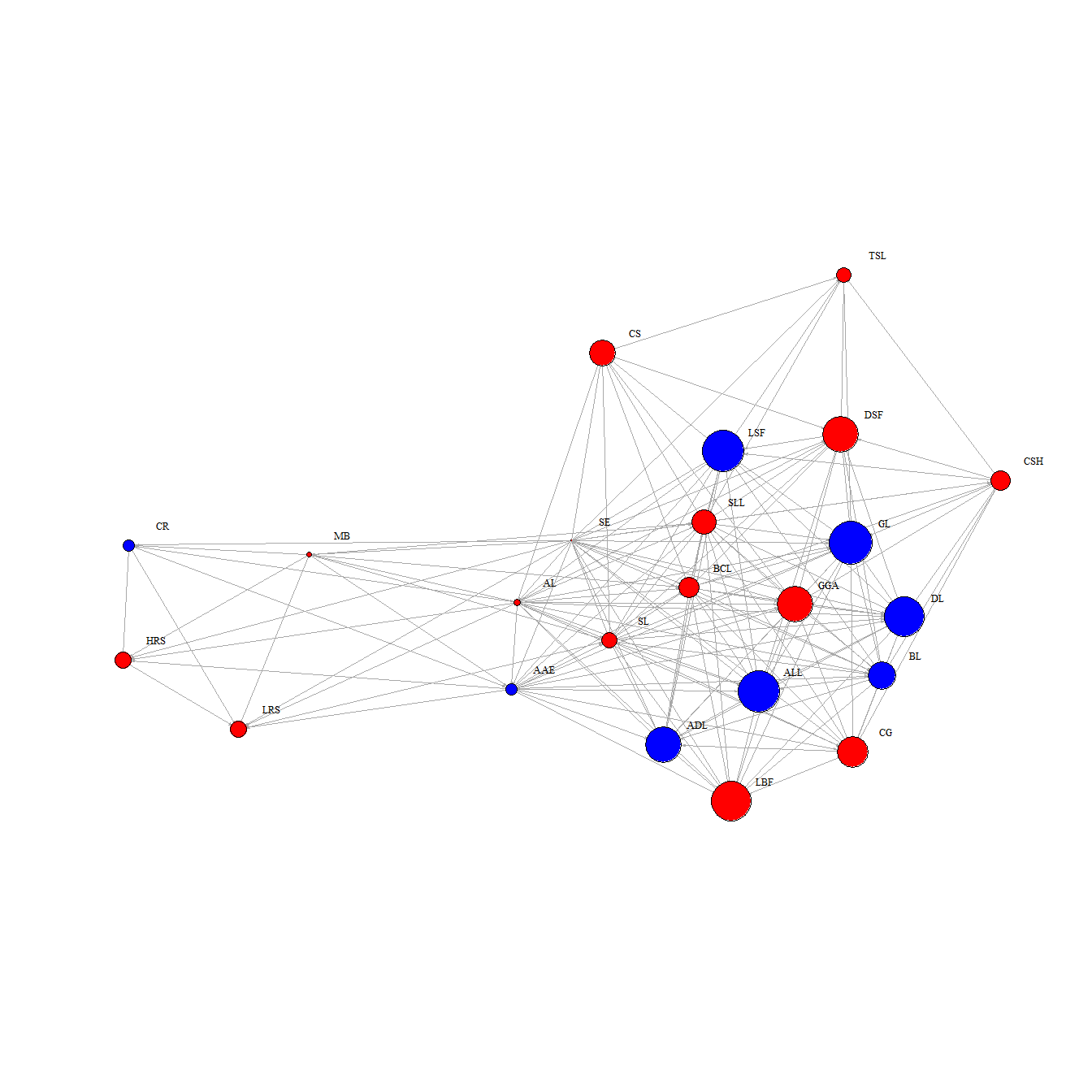
##### 8. Simplifying the network and calculating hub and authority scores and displaying these details in two networks plots, as described in task 7

* Simplify the original network containing all Facebook users so that only those nodes with a degree of greater than 10 are plotted. In this simplified network use the hub\_score and authority\_score functions to calculate these scores. Create two network plots – one showing the hubs and another showing the authorities. The node size in each of these plots should be 10 times the value of the hub and authority scores. Your plots should be appropriately labelled and displayed using the layout style of layout\_with\_fr. By visually inspecting the network displaying hubs, identify the 5 most important female Facebook users.

# The hub score and plots  
hub=hub\_score(net5)  
  
graph\_attr(net5, "layout") <- layout\_with\_fr #set the layout  
  
V(net5)$size<-hub$vector\*10 #adjust the node size to be same as the hub score  
#change arrow size   
E(net5)$arrow.size <- 0.2  
  
# plot based on hubs scores  
  
plot(net5,layout=layout\_with\_fr,vertex.label.cex=0.8,  
 vertex.label.dist=1.5,vertex.label.color="black",asp=0.6)  
legend("topright", legend = paste(c('MALE','FEMALE')),  
 pch = 15, inset=c(0.7,-0.2),  
 col = c("blue","red"), cex=0.7)



# The Authorities score and plot  
authority=authority\_score(net5)  
  
graph\_attr(net5, "layout") <- layout\_with\_fr #set the layout  
  
V(net5)$size<-authority$vector\*10 #adjust the node size to be same as the authority score  
  
#change arrow size   
E(net5)$arrow.size <- 0.2  
  
# The plot based on authorities  
  
plot(net5,layout=layout\_with\_fr,vertex.label.cex=0.8,  
 vertex.label.dist=1.5,vertex.label.color="black",asp=0.6)  
legend("topright", legend = paste( c('MALE','FEMALE')),  
 pch = 15, inset=c(0.8,-0.2),  
 col = c("blue","red"), cex=0.7)



##### 9. Calculating measures of centrality and density, as described in task 8.

* Calculate the betweeness (ignoring loops) of each node and edge density value for the network. Identify the node with the highest betweeness value in this network.

deg <- centralization.degree(net5)$res  
deg

## [1] 22 9 20 6 18 6 7 18 18 19 16 13 11 16 14 16 13 17 9 14 14 15 7