Code to TAS

Dongmeng Liu

3/21/2017

Define the functions used to calculate the dissimilarity-based measures:

```
# Normalize the dissimilarity matrix if the entries are not between 0 and 1.
# Normalization is only necessary for the dissimilarity-based measure.
# dmat is the dissimilarity matrix.
# standardrange=TRUE means that the dissimilarity entries are between 0 and 1,
# and standardrange=FALSE otherwise.
# N is the number of individuals.
Dmat<-function(dmat, standardrange=TRUE, N){
  Dmat<-matrix(NA, nrow=N, ncol=N)
    end<-0
    for(i in 1:(N-1)){
        start<-end+1
        end<-start+N-i-1
        Dmat[i,(i+1):N]<-dmat[start:end]
        for(j in i:N) Dmat[j,i]<-Dmat[i,j]
    }
    diag(Dmat)<-0
    if(standardrange==FALSE) Dmat<-(Dmat-min(Dmat))/(max(Dmat)-min(Dmat))
    return(Dmat)
}

# Calculate the Euclidean distance matrix of the top coordinates obtained
# from Multiple Correspondence Analysis
# Normalize the distance matrix
# ind_coord are the coordinates obtained from Multiple Correspondence Analysis
# scale_range=FALSE means the matrix entries are between 0 and 1;
# Otherwise, scale_range=FALSE
Dmat_MCA<-function(ind_coord, scale_range=FALSE){
    N_ind<-nrow(ind_coord)
    Dmat_MCA<-matrix(0, ncol=N_ind, nrow=N_ind)
    for(i in 1:(nrow(Dmat_MCA)-1)){
        for(j in i:nrow(Dmat_MCA))
          Dmat_MCA[i,j]<-Dmat_MCA[j,i]<-mean((ind_coord[i,]-ind_coord[j,])^2)
    }
    if(scale_range==TRUE) Dmat_MCA<-
      (Dmat_MCA-min(Dmat_MCA))/(max(Dmat_MCA)-min(Dmat_MCA))
    return(Dmat_MCA)
}

# Calculate the distance between an individual and each cluster.
# Dmat is returned by the Dmat function.
# Z is a vector of cluster membership assignment given by the hard clustering method.
hmat<-function(Dmat, Z){
    C<-nlevels(as.factor(Z))
    hmat<-matrix(0, nrow=nrow(Dmat), ncol=C)
```

```
    for(j in 1:C){
        for(i in 1:nrow(Dmat))
          hmat[i,j]<-mean(Dmat[i,which(Z==j)[which(Z==j)!=i]])
        if(length(which(Z==j))==1) hmat[which(Z==j),j]<-0
    }
    return(hmat)
}

# Calculate the dissimilarity-based measure.
# hmat is returned by the hmat function.
# exp is the user-specified tuning parameter.
ClustMemb<-function(hmat, exp){
    ind_sim<-(1-hmat)^exp
    post_prob<-matrix(NA, nrow=nrow(hmat), ncol=ncol(hmat))
    for(i in 1:nrow(hmat)) post_prob[i,]<-ind_sim[i,]/sum(ind_sim[i,])
    post_prob<-data.frame(post_prob)
    colnames(post_prob)<-paste("cluster", 1:ncol(hmat), sep="")
    post_prob$zbest<-apply(post_prob, 1, which.max)
    return(round(post_prob,5))
}
```

Load the packages:

```
library(cluster) # to apply partitioning around medoids and FANNY algorithm
library(FactoMineR) # to do multiple correspondence analysis
library(ade4) # to calculate the simple matching distance
library(PReMiuM) # to run Bayesian profile regression
library(poLCA) # to run latent class analysis
library(mclust) # to run Gaussian model-based clustering
```

Proposed Measures and FANNY Algorithm

Euclidean Distance Matrix

Simulate the data, and calculate the $P_{h1}^{(1)}$ and $P_{h1}^{(2)}$ as well as $M^{(1)}$ and $M^{(2)}$ for the Euclidean distance matrix ($l = 0.5$, $v = 5$). The posterior-probabilities and soft misclassification rate for FANNY with $r = 2$ is also obtained by running the following code.

```
set.seed(1214)
N_nonhyb<-40; N_hyb<-1 # 40 non-hybrid individuals and 1 hybrid individual
P_binary<-20 # 20 binary features
beta<-1.2; t<--3 # beta and t are the parameters of the simulation study setting
nite<-1000 # the number of iterations

# tuning parameter, l, for the silhouette-based measure
l_tuning<-0.5
# store the silhouette values (not the silhouette-based measures!)
Ph1_sil<-matrix(0, nrow=nite, ncol=2)
# store the silhouette-based measures of the hybrid individuals
Ph1_sil<-matrix(0,nrow=nrow(Ph1_sil),ncol=ncol(Ph1_sil))
# soft misclassification rate for the silhouette-based measure
M_sil<-0
```

```
# tuning parameter, v, for the dissimilarity-based measure
v_tuning<-5
# store the dissimilarity-based measures of the hybrid being assigned to cluster 1
Ph1_dissim<-NULL
# soft misclassification rate for the dissimilarity-based measure
M_dis<-0

# tuning parameter, r, for FANNY
r_tuning<-2
# store the FANNY posterior probabilities of hybrid individuals
Ph1_fanny<-NULL
# soft misclassification rate for FANNY
M_fanny<-0

for(count in 1:nite){
  # col 1-P_binary: binary features; col P_binary+1: outcome; col P_binary+2: true group
  mydat_disc<-matrix(0, nrow=(N_nonhyb+N_hyb), ncol=P_binary+2)

# Generate the outcome and true group of individuals in group 1
for(i in 1:(N_nonhyb/2)) {
    mydat_disc[i,P_binary+1]<-rbinom(1,1,0.7)
    mydat_disc[i,P_binary+2]<-1
}
# Generate the outcome and true group of individuals in group 2
for(i in (N_nonhyb/2+1):N_nonhyb) {
    mydat_disc[i,P_binary+1]<-rbinom(1,1,0.3)
    mydat_disc[i,P_binary+2]<-2
}
# Generate the outcome and true group of the hybrid individual (group 3)
mydat_disc[(N_nonhyb+1),P_binary+1]<-rbinom(1,1,0.5)
mydat_disc[(N_nonhyb+1),P_binary+2]<-3

# Generate the features in group 1: U=1
for(i in 1:(N_nonhyb/2)) {
    U<-1
    for(j in 1:P_binary) mydat_disc[i,j]<-rbinom(1,1,exp(t+beta*U)/(1+exp(t+beta*U)))
}

# Generate the features in group 2: U=4
for(i in (N_nonhyb/2+1):N_nonhyb) {
    U<-4
    for(j in 1:P_binary) mydat_disc[i,j]<-rbinom(1,1,exp(t+beta*U)/(1+exp(t+beta*U)))
}

# Generate the features of the hybrid individual:
for(j in 1:(P_binary/2)){
  mydat_disc[(N_nonhyb+1),j]<-rbinom(1,1,exp(t+beta*1)/(1+exp(t+beta*1)))
}
for(j in (P_binary/2+1):P_binary){
  mydat_disc[(N_nonhyb+1),j]<-rbinom(1,1,exp(t+beta*4)/(1+exp(t+beta*4)))
}

mydat_disc<-data.frame(mydat_disc)
```

```
covnames_mydat_disc<-paste("X", 1:P_binary, sep="_")
colnames(mydat_disc)<-c(covnames_mydat_disc, "outcome", "group")
for (i in 1:ncol(mydat_disc)) mydat_disc[,i]<-as.factor(mydat_disc[,i])

# Multiple correspondence analysis
mca_disc=MCA(mydat_disc[,-(P_binary+1)], quali.sup=(P_binary+1), graph=FALSE, ncp=2)
mydat_dissim_mca<-Dmat_MCA(mca_disc$ind$coord, scale_range=TRUE)
mydat_dissim_sil<-Dmat_MCA(mca_disc$ind$coord, scale_range=FALSE)

# Do Partitioning Around Medoids
mydat_mca_pam<-pam(mydat_dissim_mca, 2, diss=TRUE)
hmat_pam<-hmat(mydat_dissim_mca, mydat_mca_pam$clustering)

a<-sum(mydat_disc$group[which(mydat_mca_pam$clustering==1)]==1)
b<-sum(mydat_disc$group[which(mydat_mca_pam$clustering==2)]==1)
c<-sum(mydat_disc$group[which(mydat_mca_pam$clustering==1)]==2)
d<-sum(mydat_disc$group[which(mydat_mca_pam$clustering==2)]==2)
# H is defined as cluster 1 and H2 is defined as cluster 2, given the cluster membership assignment by
H<-which.max(c(a,b)); H2<-which.min(c(a,b))

# Obtain the dissimilarity-based measure of the hybrid individual to cluster 1
for(l in 1:20){
    M_dis<-M_dis+ClustMemb(hmat_pam, v_tuning)[l,H2]
}
for(l in 21:40){
    M_dis<-M_dis+ClustMemb(hmat_pam, v_tuning)[l,H]
}
Ph1_dissim[count]<-ClustMemb(hmat_pam, v_tuning)[N_nonhyb+N_hyb,H]

# Obtain the Silhouette-based measure of the hybrid individual
nonhyb_sil<-silhouette(mydat_mca_pam$clustering, mydat_dissim_sil)[1:N_nonhyb,3]
M_sil<-M_sil+
  sum((1-nonhyb_sil)^l_tuning/((nonhyb_sil+1)^l_tuning+(1-nonhyb_sil)^l_tuning))
Ph1_sil[count,1]<-
  silhouette(c(mydat_mca_pam$clustering[1:N_nonhyb],H), mydat_dissim_sil)[N_nonhyb+N_hyb,3]
Ph1_sil[count,2]<-
  silhouette(c(mydat_mca_pam$clustering[1:N_nonhyb],H2), mydat_dissim_sil)[N_nonhyb+N_hyb,3]
Ph1_sil[count,1]<-((Ph1_sil[count,1]+1)^l_tuning)/
  ((Ph1_sil[count,1]+1)^l_tuning+(Ph1_sil[count,2]+1)^l_tuning)
Ph1_sil[count,2]<-1-Ph1_sil[count,1]

# Define cluster 1 and cluster 2, given the clustering by FANNY
mydat_mca_fanny<-fanny(mydat_dissim_mca, 2, diss=TRUE, memb.exp=r_tuning)
a<-sum(mydat_disc$group[which(mydat_mca_fanny$clustering==1)]==1)
b<-sum(mydat_disc$group[which(mydat_mca_fanny$clustering==2)]==1)
c<-sum(mydat_disc$group[which(mydat_mca_fanny$clustering==1)]==2)
d<-sum(mydat_disc$group[which(mydat_mca_fanny$clustering==2)]==2)
H<-which.max(c(a,b)); H2<-which.min(c(a,b))
for(l in 1:20){
    M_fanny<-M_fanny+mydat_mca_fanny$membership[l,H2]
}
for(l in 21:40){
    M_fanny<-M_fanny+mydat_mca_fanny$membership[l,H]
```

```
}
# Obtain the FANNY posterior probabilities
Ph1_fanny[count]<-mydat_mca_fanny$membership[N_nonhyb+N_hyb,H]
}
```

The mean and sd of $P_{h1}^{(1)}$, $P_{h1}^{(2)}$ and FANNY posterior probabilities are shown as below

```
print(paste0('the silhouette-based measure: mean=', round(mean(Ph1_sil[,1]), digits=2),
             ' sd=', round(sd(Ph1_sil[,1]), digits=2)))
```

```
## [1] "the silhouette-based measure: mean=0.5 sd=0.11"
```

```
print(paste0('the dissimilarity-based measure: mean=', round(mean(Ph1_dissim), digits=2),
             ' sd=', round(sd(Ph1_dissim), digits=2)))
```

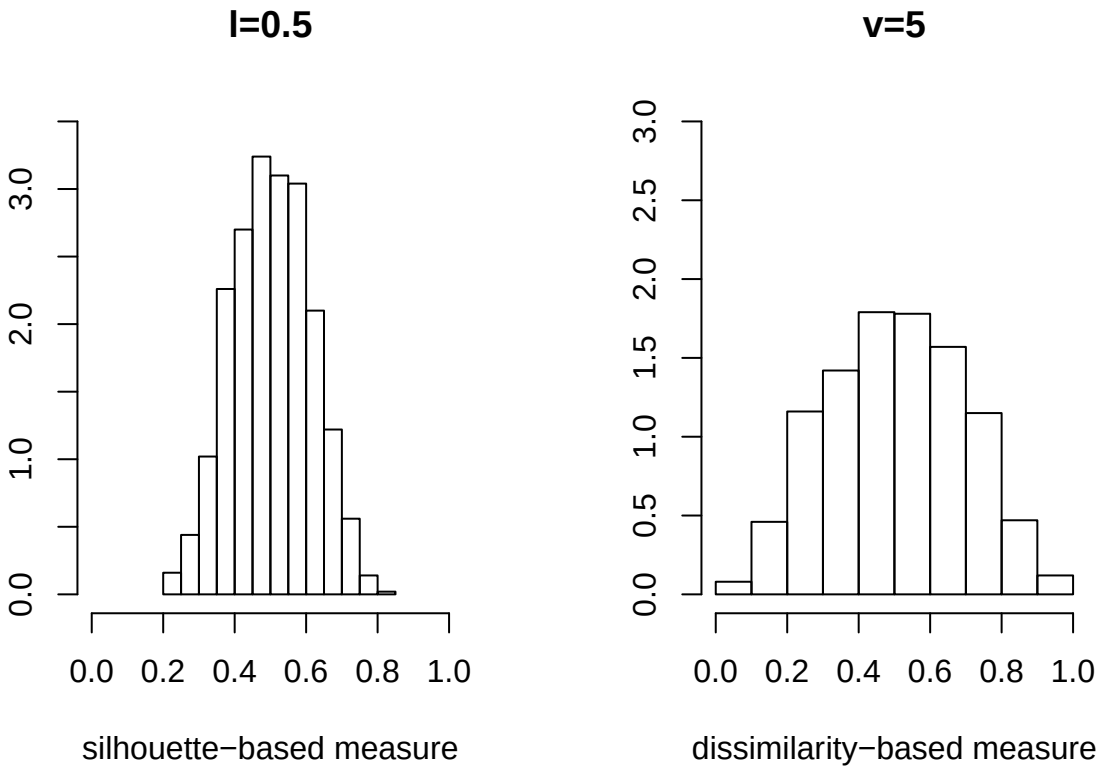
```
## [1] "the dissimilarity-based measure: mean=0.5 sd=0.19"
```

```
print(paste0('the FANNY posterior probabilities: mean=', round(mean(Ph1_fanny), digits=2),
             ' sd=', round(sd(Ph1_fanny), digits=2)))
```

```
## [1] "the FANNY posterior probabilities: mean=0.5 sd=0.16"
```

The empirical distribution of $P_{h1}^{(1)}$ and $P_{h1}^{(2)}$ when $l = 0.5$ and $v = 5$ can be obtained as below:

```
par(mfrow=c(1,2))
hist(Ph1_sil[,1], freq=FALSE, xlim=c(0,1), ylim=c(0,3.5),
     xlab="silhouette-based measure", ylab="", main="l=0.5")
hist(Ph1_dissim, freq=FALSE, xlim=c(0,1), ylim=c(0,3),
     xlab="dissimilarity-based measure", ylab="", main="v=5")
```



$M^{(1)}$, $M^{(2)}$ and the soft misclassification rates for FANNY are calculated as below:

```
# M^{(1)}, the soft misclassification rate of the silhouette-based measure
M_sil/(nite*N_nonhyb)
```

```
## [1] 0.1695618
```

```
# M^{(2)}, the soft misclassification rate of the dissimilarity-based measure
M_dis/(nite*N_nonhyb)
```

```
## [1] 0.02463846
```

```
# the soft misclassification rate of FANNY
M_fanny/(nite*N_nonhyb)
```

```
## [1] 0.03875335
```

Simple Matching Distance Matrix

Likewise, $P_{h1}^{(1)}$, $P_{h1}^{(2)}$, $M^{(1)}$, $M^{(2)}$ as well as the posterior-probabilities and soft misclassification rate of FANNY for the simple matching matrix (SMD) can be obtained as below:

```
set.seed(1214)
N_nonhyb<-40; N_hyb<-1; P_binary<-20
beta<-1.2
t<--3
nite<-1000

l_tuning<-3.0 # the tuning parameter, l, for the silhouette-based measure
Ph1_sil<-matrix(0, nrow=nite, ncol=2)
M_sil<-0

v_tuning<-2.3 # the tuning parameter, v, for the dissimilarity-based measure
M_dis<-0
Ph1_dissim<-NULL

r_tuning<-2 # tuning parameter, r, for FANNY
Ph1_fanny<-NULL
M_fanny<-0

# Start generating the data set, the same as before:
for(count in 1:nite){
mydat_disc<-matrix(0, nrow=(N_nonhyb+N_hyb), ncol=P_binary+2)

for(i in 1:(N_nonhyb/2)) {
    mydat_disc[i,P_binary+1]<-rbinom(1,1,0.7)
    mydat_disc[i,P_binary+2]<-1
}
for(i in (N_nonhyb/2+1):N_nonhyb) {
    mydat_disc[i,P_binary+1]<-rbinom(1,1,0.3)
    mydat_disc[i,P_binary+2]<-2
}
mydat_disc[(N_nonhyb+1),P_binary+1]<-rbinom(1,1,0.5)
mydat_disc[(N_nonhyb+1),P_binary+2]<-3

for(i in 1:(N_nonhyb/2)) {
    U<-1
```

```
    for(j in 1:P_binary) mydat_disc[i,j]<-rbinom(1,1,exp(t+beta*U)/(1+exp(t+beta*U)))
}

for(i in (N_nonhyb/2+1):N_nonhyb) {
    U<-4
    for(j in 1:P_binary) mydat_disc[i,j]<-rbinom(1,1,exp(t+beta*U)/(1+exp(t+beta*U)))
}

for(j in 1:(P_binary/2)){
  mydat_disc[(N_nonhyb+1),j]<-rbinom(1,1,exp(t+beta*1)/(1+exp(t+beta*1)))
}
for(j in (P_binary/2+1):P_binary){
  mydat_disc[(N_nonhyb+1),j]<-rbinom(1,1,exp(t+beta*4)/(1+exp(t+beta*4)))
}

mydat_disc<-data.frame(mydat_disc)
covnames_mydat_disc<-paste("X", 1:P_binary, sep="_")
colnames(mydat_disc)<-c(covnames_mydat_disc, "outcome", "group")
for (i in 1:(ncol(mydat_disc)-1)) mydat_disc[,i]<-as.numeric(mydat_disc[,i])

# Obtain the simple matching distance (SMD) matrix
d.mydat<-dist.binary(mydat_disc[,1:P_binary], method=2, diag=TRUE)
mydat_smd<-Dmat(d.mydat, standardrange=TRUE, N_nonhyb+N_hyb)

# Do Partitioning Around Medoids
mydat_smd_pam<-pam(mydat_smd, 2, diss=TRUE)
hmat_pam<-hmat(mydat_smd, mydat_smd_pam$clustering)

a<-sum(mydat_disc$group[which(mydat_smd_pam$clustering[1:N_nonhyb]==1)]==1)
b<-sum(mydat_disc$group[which(mydat_smd_pam$clustering[1:N_nonhyb]==2)]==1)
c<-sum(mydat_disc$group[which(mydat_smd_pam$clustering[1:N_nonhyb]==1)]==2)
d<-sum(mydat_disc$group[which(mydat_smd_pam$clustering[1:N_nonhyb]==2)]==2)
H<-which.max(c(a,b)); H2<-which.min(c(a,b))

# Dissimilarity-based measure
for(l in 1:20){
    M_dis<-M_dis+ClustMemb(hmat_pam, v_tuning)[l,H2]
}
for(l in 21:40){
    M_dis<-M_dis+ClustMemb(hmat_pam, v_tuning)[l,H]
}
Ph1_dissim[count]<-ClustMemb(hmat_pam, v_tuning)[N_nonhyb+N_hyb,H]

# Silhouette-based measure
nonhyb_sil<-silhouette(mydat_smd_pam$clustering, mydat_smd)[1:N_nonhyb,3]
M_sil<-M_sil+
  sum((1-nonhyb_sil)^l_tuning/((nonhyb_sil+1)^l_tuning+(1-nonhyb_sil)^l_tuning))
Ph1_sil[count,1]<-
  silhouette(c(mydat_smd_pam$clustering[1:N_nonhyb],H), mydat_smd)[N_nonhyb+N_hyb,3]
Ph1_sil[count,2]<-
  silhouette(c(mydat_smd_pam$clustering[1:N_nonhyb],H2), mydat_smd)[N_nonhyb+N_hyb,3]
Ph1_sil[count,1]<-((Ph1_sil[count,1]+1)^l_tuning)/
  ((Ph1_sil[count,1]+1)^l_tuning+(Ph1_sil[count,2]+1)^l_tuning)
```

```
Ph1_sil[count,2]<-1-Ph1_sil[count,1]

# Obtain the FANNY posterior probabilities
mydat_smd_fanny<-fanny(mydat_smd, 2, diss=TRUE, memb.exp=r_tuning)
a<-sum(mydat_disc$group[which(mydat_smd_fanny$clustering==1)]==1)
b<-sum(mydat_disc$group[which(mydat_smd_fanny$clustering==2)]==1)
c<-sum(mydat_disc$group[which(mydat_smd_fanny$clustering==1)]==2)
d<-sum(mydat_disc$group[which(mydat_smd_fanny$clustering==2)]==2)
# Define cluster 1 and cluster 2, given the clustering by FANNY
H<-which.max(c(a,b)); H2<-which.min(c(a,b))
for(l in 1:20){
    M_fanny<-M_fanny+mydat_smd_fanny$membership[l,H2]
}
for(l in 21:40){
    M_fanny<-M_fanny+mydat_smd_fanny$membership[l,H]
}
Ph1_fanny[count]<-mydat_smd_fanny$membership[N_nonhyb+N_hyb,H]
}
```

PReMiuM Co-occurrence Matrix

For the {PReMiuM} co-occurrence dissimilarity matrix:

```
set.seed(1214)
N_nonhyb<-40; N_hyb<-1; P_binary<-20
beta<-1.2
t<--3
nite<-1000

l_tuning<-0.6 # the tuning parameter, l, for the silhouette-based measure
Ph1_sil<-matrix(0, nrow=nite, ncol=2)
nonhyb_sil<-rep(0,N_nonhyb+N_hyb)
M_sil<-0

v_tuning<-0.13 # the tuning parameter, v, for the dissimilarity-based measure
M_dis<-0
Ph1_dissim<-NULL

r_tuning<-2 # tuning parameter, r, for FANNY
Ph1_fanny<-NULL
M_fanny<-0

for(count in 1:nite){
mydat_disc<-matrix(0, nrow=(N_nonhyb+N_hyb), ncol=P_binary+2)
for(i in 1:(N_nonhyb/2)) {
    mydat_disc[i,P_binary+1]<-rbinom(1,1,0.7)
    mydat_disc[i,P_binary+2]<-1
}
for(i in (N_nonhyb/2+1):N_nonhyb) {
    mydat_disc[i,P_binary+1]<-rbinom(1,1,0.3)
    mydat_disc[i,P_binary+2]<-2
}
mydat_disc[(N_nonhyb+1),P_binary+1]<-rbinom(1,1,0.5)
mydat_disc[(N_nonhyb+1),P_binary+2]<-3
```

```
# Group 1: U=1
for(i in 1:(N_nonhyb/2)) {
    U<-1
    for(j in 1:P_binary) mydat_disc[i,j]<-rbinom(1,1,exp(t+beta*U)/(1+exp(t+beta*U)))
}
# Group 2: U=4
for(i in (N_nonhyb/2+1):N_nonhyb) {
    U<-4
    for(j in 1:P_binary) mydat_disc[i,j]<-rbinom(1,1,exp(t+beta*U)/(1+exp(t+beta*U)))
}
# hybrid individuals:
for(j in 1:(P_binary/2)){
  mydat_disc[(N_nonhyb+1),j]<-rbinom(1,1,exp(t+beta*1)/(1+exp(t+beta*1)))
}
for(j in (P_binary/2+1):P_binary){
  mydat_disc[(N_nonhyb+1),j]<-rbinom(1,1,exp(t+beta*4)/(1+exp(t+beta*4)))
}
mydat_disc<-data.frame(mydat_disc)
covnames_mydat_disc<-paste("X", 1:P_binary, sep="_")
colnames(mydat_disc)<-c(covnames_mydat_disc, "outcome", "group")

# Do Bayesian profile regression:
runInfoObj<-profRegr(yModel="Bernoulli", xModel="Discrete", nSweeps=10000, nBurn=300,
                     data=mydat_disc[1:(N_nonhyb+N_hyb),1:(P_binary+1)],
                     output="PReMiuM", covNames=colnames(mydat_disc)[1:P_binary],
                     nClusInit=20, run=TRUE, seed=3459, excludeY=FALSE)
# Calculate the co-occurrence dissimilarity matrix
dissimObj<-calcDissimilarityMatrix(runInfoObj)
# Standardize the co-occurrence dissimilarity matrix
mydat_premium_dissim<-
  Dmat(dmat=dissimObj$disSimMat, standardrange=TRUE, N=(N_nonhyb+N_hyb))

# Do Partitioning Around Medoids
mydat_premium_pam<-pam(mydat_premium_dissim, 2, diss=TRUE)
hmat_pam<-hmat(mydat_premium_dissim, mydat_premium_pam$clustering)

a<-sum(mydat_disc$group[which(mydat_premium_pam$clustering[1:N_nonhyb]==1)]==1)
b<-sum(mydat_disc$group[which(mydat_premium_pam$clustering[1:N_nonhyb]==2)]==1)
c<-sum(mydat_disc$group[which(mydat_premium_pam$clustering[1:N_nonhyb]==1)]==2)
d<-sum(mydat_disc$group[which(mydat_premium_pam$clustering[1:N_nonhyb]==2)]==2)
H<-which.max(c(a,b)); H2<-which.min(c(a,b))

# Dissimilarity-based measure
for(l in 1:20){
    M_dis<-M_dis+ClustMemb(hmat_pam, v_tuning)[l,H2]
}
for(l in 21:40){
    M_dis<-M_dis+ClustMemb(hmat_pam, v_tuning)[l,H]
    }
Ph1_dissim[count]<-ClustMemb(hmat_pam, v_tuning)[N_nonhyb+N_hyb,H]

# Silhouette-based measure
nonhyb_sil<-silhouette(mydat_premium_pam$clustering, mydat_premium_dissim)[1:N_nonhyb,3]
```

```
M_sil<-M_sil+
  sum((1-nonhyb_sil)^l_tuning/((nonhyb_sil+1)^l_tuning+(1-nonhyb_sil)^l_tuning))
Ph1_sil[count,1]<-silhouette(c(mydat_premium_pam$clustering[1:N_nonhyb],H),
                             mydat_premium_dissim)[N_nonhyb+N_hyb,3]
Ph1_sil[count,2]<-silhouette(c(mydat_premium_pam$clustering[1:N_nonhyb],H2),
                             mydat_premium_dissim)[N_nonhyb+N_hyb,3]
Ph1_sil[count,1]<-((Ph1_sil[count,1]+1)^l_tuning)/
  ((Ph1_sil[count,1]+1)^l_tuning+(Ph1_sil[count,2]+1)^l_tuning)
Ph1_sil[count,2]<-1-Ph1_sil[count,1]

# Obtain the FANNY posterior probabilities
mydat_premium_fanny<-fanny(mydat_premium_dissim, 2, diss=TRUE, memb.exp=r_tuning)
a<-sum(mydat_disc$group[which(mydat_premium_fanny$clustering==1)]==1)
b<-sum(mydat_disc$group[which(mydat_premium_fanny$clustering==2)]==1)
c<-sum(mydat_disc$group[which(mydat_premium_fanny$clustering==1)]==2)
d<-sum(mydat_disc$group[which(mydat_premium_fanny$clustering==2)]==2)
# Define cluster 1 and cluster 2, given the clustering by FANNY
H<-which.max(c(a,b)); H2<-which.min(c(a,b))
for(l in 1:20){
    M_fanny<-M_fanny+mydat_premium_fanny$membership[l,H2]
}
for(l in 21:40){
    M_fanny<-M_fanny+mydat_premium_fanny$membership[l,H]
    }
Ph1_fanny[count]<-mydat_premium_fanny$membership[N_nonhyb+N_hyb,H]
}
```

Model-Based Clustering Methods

For comparison, the posterior probabilities for the Latent Class Analysis and Gaussian Model-based Clustering can be obtained as below:

```
set.seed(1214)
N_nonhyb<-40; N_hyb<-1; P_binary<-20
beta<-1.2
t<--3
nite<-1000

# store the posterior prob of LCA
Ph1_lca<-NULL
# the soft misclassification rate for LCA
M_lca<-0

# store the posterior prob of Gaussian model-based clustering
Ph1_Gaussian<-NULL
# the soft misclassification rate for Gaussian model-based clustering
M_Gaussian<-0

for(count in 1:nite){
mydat_disc<-matrix(0, nrow=(N_nonhyb+N_hyb), ncol=P_binary+2)
for(i in 1:(N_nonhyb/2)) {
    mydat_disc[i,P_binary+1]<-rbinom(1,1,0.7)
    mydat_disc[i,P_binary+2]<-1
```

```
}
for(i in (N_nonhyb/2+1):N_nonhyb) {
    mydat_disc[i,P_binary+1]<-rbinom(1,1,0.3)
    mydat_disc[i,P_binary+2]<-2
}
mydat_disc[(N_nonhyb+1),P_binary+1]<-rbinom(1,1,0.5)
mydat_disc[(N_nonhyb+1),P_binary+2]<-3
for(i in 1:(N_nonhyb/2)) {
    U<-1
    for(j in 1:P_binary) mydat_disc[i,j]<-rbinom(1,1,exp(t+beta*U)/(1+exp(t+beta*U)))
}
for(i in (N_nonhyb/2+1):N_nonhyb) {
    U<-4
    for(j in 1:P_binary) mydat_disc[i,j]<-rbinom(1,1,exp(t+beta*U)/(1+exp(t+beta*U)))
}
for(j in 1:(P_binary/2)){
  mydat_disc[(N_nonhyb+1),j]<-rbinom(1,1,exp(t+beta*1)/(1+exp(t+beta*1)))
}
for(j in (P_binary/2+1):P_binary){
  mydat_disc[(N_nonhyb+1),j]<-rbinom(1,1,exp(t+beta*4)/(1+exp(t+beta*4)))
}
mydat_disc<-data.frame(mydat_disc)
covnames_mydat_disc<-paste("X", 1:P_binary, sep="_")
colnames(mydat_disc)<-c(covnames_mydat_disc, "outcome", "group")
for (i in 1:(ncol(mydat_disc)-1)) mydat_disc[,i]<-as.factor(mydat_disc[,i])

# Method: LCA
mydat_lca<-poLCA(formula=
  cbind(X_1,X_2,X_3,X_4,X_5,X_6,X_7,X_8,X_9,X_10,X_11,X_12,X_13,X_14,X_15,
  X_16,X_17,X_18,X_19,X_20) ~ 1, data=mydat_disc, nclass=2, na.rm=TRUE, nrep=30)
# Do Latent Class Analysis
lca_posterior<-data.frame(mydat_lca$posterior, mydat_lca$predclass)

a<-sum(mydat_disc$group[which(mydat_lca$predclass[1:N_nonhyb]==1)]==1)
b<-sum(mydat_disc$group[which(mydat_lca$predclass[1:N_nonhyb]==2)]==1)
c<-sum(mydat_disc$group[which(mydat_lca$predclass[1:N_nonhyb]==1)]==2)
d<-sum(mydat_disc$group[which(mydat_lca$predclass[1:N_nonhyb]==2)]==2)
# Define cluster 1 and cluster 2 given the cluster membership assignment from LCA
H<-which.max(c(a,b)); H2<-which.min(c(a,b))

for(l in 1:20){
    M_lca<-M_lca+lca_posterior[l,H2]
}
for(l in 21:40){
    M_lca<-M_lca+lca_posterior[l,H]
    }
# Obtain the LCA posterior probabilities
Ph1_lca[count]<-lca_posterior[N_nonhyb+N_hyb,H]

# Gaussian model-based clustering on the top 2 coordinates obtained from MCA
# Obtain the top 2 coordinates from MCA
mca_disc=MCA(mydat_disc[,-(P_binary+1)], quali.sup=(P_binary+1), graph=FALSE, ncp=2)
mca_pc<-mca_disc$ind$coord[,1:2]
```

```
# Do Gaussian mixture model
mydat_mca_Gaussian<-Mclust(mca_pc, G=2, modelNames="EEI")

a<-sum(mydat_disc$group[which(mydat_mca_Gaussian$classification[1:N_nonhyb]==1)]==1)
b<-sum(mydat_disc$group[which(mydat_mca_Gaussian$classification[1:N_nonhyb]==2)]==1)
c<-sum(mydat_disc$group[which(mydat_mca_Gaussian$classification[1:N_nonhyb]==1)]==2)
d<-sum(mydat_disc$group[which(mydat_mca_Gaussian$classification[1:N_nonhyb]==2)]==2)
H<-which.max(c(a,b)); H2<-which.min(c(a,b))

for(l in 1:20){
    M_Gaussian<-M_Gaussian+mydat_mca_Gaussian$z[l,H2]
}
for(l in 21:40){
    M_Gaussian<-M_Gaussian+mydat_mca_Gaussian$z[l,H]
    }
Ph1_Gaussian[count]<-mydat_mca_Gaussian$z[N_nonhyb+N_hyb,H]
}
```