```
#similar to Azarnoush2016 simulation with random effects

#Generating data and network
library(MASS)
library(matrixStats)
library(matrixcalc)
library(expm)
library(tictoc)
library(igraph)
library(psych)
library(lme4)
tic() # Start the clock!
rm(list=ls()) # clear workspace
cat("\014") # Clear console
V=50 # number of nodes
NS=10
b1=0.1 #covariate's coefficient
b2=-0.2
b3=0.3
sigma1<-0.5
sigma2<-1
t1<-matrix(rep(0, NS*(3+2*V)),NS,(3+2*V))
t4<-t1
for (i1 in 1:NS){
 teta<-matrix(rep(0, V*V*(5+2*V)),V*V,5+2*V)
 r1<-rnorm(V,0,sigma1) #random effect 1
 r2<-rnorm(V,0,sigma2) #random effect 2
 for (i in 1:V){
  for (j in 1:V){
   teta[(i-1)*V+j,1]<-i #node i
   teta[(i-1)*V+j,2]<-j  #node j
      }
 }
 teta[,4]<-runif(V*V,20,50) #first attribute
 teta[,5]<-rbinom(V*V, 1,.5) #second attribute
 A<-c(rep(0,V*V))
 for (i in 1:(V*V)){
  A[i]<-b1+b2*teta[i,4]+b3*teta[i,5]+r1[teta[i,1]]+r2[teta[i,2]]
  #with two random effects
  teta[i,3]<-exp(A[i])/(1+exp(A[i]))
  for (j in 1:(V-1)){
   if (teta[i,1]==j){
    teta[i,5+j]=1}  #ingoingness
   if (teta[i,2]==j){
```

```
      teta[i,5+V+j]=1} #outgoingness
   }
  }
  a<-seq(1,V*V,V+1)
  teta<-teta[-a,] #removing self loops
  A<-A[-a]
  E<-c(rep(0, nrow(teta))) #creating edges based on probability
  for (i in 1:nrow(teta)){
   E[i]<-rbinom(1,1,teta[i,3])} #Edges based on random effects"
   mydata<-as.data.frame(teta[,4:(5+2*V)])
   adj<-matrix(rep(0, V*V),V,V)
   mg<-cbind(teta[,1:2],E)
  for (i in 1:nrow(teta)){
   if (mg[i,3]==1){
     adj[mg[i,1],mg[i,2]]<-1
   }
  }
  g<-graph_from_adjacency_matrix(adj, weighted=NULL, mode="directed")
  plot(g,edge.arrow.size=0.25, vertex.size = 10, vertex.label.cex = 0.5)


  # parameter estimation usin GLM
  gl<-glm(E~teta[,4]+teta[,5],family=binomial())
  gl1<-glm(E~.,mydata,family=binomial())
  l1<-lm(A~.,mydata)
  for (i in 1:(2+2*V)){
   t1[i1,i]<-summary(gl1)$coefficients[i]
   t4[i1,i]<-summary(l1)$coefficients[i]}
}
   Est1<-colMeans(t1)
    Est4<-colMeans(t4)
lm(formula = log(teta[,3]/(1 - teta[,3])) ~ ., data = mydata)
lm(formula = log(teta[,3]/(1 - teta[,3])) ~ teta[,4]+teta[,5])


#Numerical Integral Approximation 1

library(cubature)       # load the package "cubature"
for (i in 1:40){
  pr<-function(x){b1+b2*teta[i,4]+b3*teta[i,5]+sigma1*x[1]+sigma2*x[2]}
  N<-function(x){(E[i]*exp(E[i]*pr(x))+E[i]*exp(E[i]*pr(x)+pr(x))-
exp(E[i]*pr(x)+pr(x)))/(1+exp(pr(x)))^2}
  L<-function(x){exp(E[i]*pr(x))/(1+exp(pr(x)))}
  f <- function(x) { 1/(2*pi) *N(x)* (exp((-x[1]^2 - x[2]^2) /2)) }
```

```
  g<-function(x) { 1/(2*pi) * L(x)*(exp((-x[1]^2 - x[2]^2) /2)) }# "x" is vector
  c1<-adaptIntegrate(f, lowerLimit = c(-Inf, -Inf), upperLimit = c(Inf, Inf))
  c2<-adaptIntegrate(g, lowerLimit = c(-Inf, -Inf), upperLimit = c(Inf, Inf))
  print(-as.numeric(c1[1])/as.numeric(c2[1]))
}
toc()

#Numerical Integral Approximation 2
tic()
pr<-teta[,3]
N<-c(rep(0, nrow(teta)))
L<-c(rep(0, nrow(teta)))
library(cubature)       # load the package "cubature"
for (i in 1:200){
  N[i]<-(E[i]*exp(E[i]*pr[i])+E[i]*exp(E[i]*pr[i]+pr[i])-exp(E[i]*pr[i]+pr[i]))/(1+exp(pr[i]))^2
  L[i]<-exp(E[i]*pr[i])/(1+exp(pr[i]))
  f <- function(x) { 1/(2*pi) *N[i]* (exp((-x[1]^2 - x[2]^2) /2)) }
  g<-function(x) { 1/(2*pi) * L[i]*(exp((-x[1]^2 - x[2]^2) /2)) }# "x" is vector
  c1<-adaptIntegrate(f, lowerLimit = c(-Inf, -Inf), upperLimit = c(Inf, Inf))
  c2<-adaptIntegrate(g, lowerLimit = c(-Inf, -Inf), upperLimit = c(Inf, Inf))
  print(-as.numeric(c1[1])/as.numeric(c2[1]))
}
toc()
```