

## Code for “Attract and Introduce” model and “Mirror Network” method

```
# R code for attract and introduce model
# using a mirror network method to measure
# the extent to which this model generates
# heritability of network characteristics
#
# NOTE: Many thanks to Referee 2 who wrote
# some of this code

rm(list=ls())

library(igraph) # Thanks to Gabor Csardi for the igraph package!

n<-750 # number of nodes
e<-3150 # number of edges
m<-e/n # average number of edges per node (degree)

alpha<-0.9
beta<-0.3

T<-10000 # number of simulations

twindeg1 <- rep(0, times=T)
twoutdeg1 <- rep(0, times=T)
twtr1 <- rep(0, times=T)
twbe1 <- rep(0, times=T)

twindeg2 <- rep(0, times=T)
twoutdeg2 <- rep(0, times=T)
twtr2 <- rep(0, times=T)
twbe2 <- rep(0, times=T)

for(t in 1:T) { # do simulation run

  if(t%%100==0) print(t)

  # randomly choose one individual in each network to be a twin
  twin1 <- sample(n,1)
  twin2 <- sample(n,1)

  # create first set of individual traits
  d1<-as.numeric(runif(n)<alpha)*runif(n) # distribution of attractiveness
  c11<-as.numeric(runif(n)<beta) # distribution of prob. of introducing friends

  # create original network
  g1<-graph.empty(n) # create empty graph of size n
  while(ecount(g1)<e) { # loop until enough edges are produced
    dyad<-sample(n,2) # choose a random pair
    if(runif(1)<d1[dyad[2]]) { # check if person 1 nominates person 2 as a friend
      if(runif(1)<c11[dyad[1]]) { # check if person 1 introduces person 2 to friends
        ne<-neighbors(g1,dyad[1]-1,mode="out") # get person 1's neighbors
        if(length(ne)>0) { # check if there is at least one existing neighbor

          for(i in 1:length(ne)) { # loop through neighbors
            if(runif(1)<d1[ne[i]+1]) g1<-add.edges(g1,c(dyad[2]-1,ne[i])) # introduce friends
            if(runif(1)<c11[dyad[2]]) g1<-add.edges(g1,c(ne[i],dyad[2]-1)) # and add edges if
            # they become friends
          }
        }
      }
    }
    g1<-add.edges(g1,dyad-1) # add random pair edge at end
  }
  g1<-simplify(g1) # remove duplicate edges and loops
}

# create second set of individual traits
d2<-as.numeric(runif(n)<alpha)*runif(n) # distribution of attractiveness
c12<-as.numeric(runif(n)<beta) # distribution of prob. of introducing friends

# copy genes
d2[twin2] <- d1[twin1]
c12[twin2] <- c11[twin1]

# create mirror network
g2<-graph.empty(n)
while(ecount(g2)<e) {
```

```

dyad<-sample(n,2)
if(runif(1)<d2[dyad[2]]) {
  if(runif(1)<c12[dyad[1]]) {
    ne<-neighbors(g2,dyad[1]-1,mode="out")
    if(length(ne)>0) {
      for(i in 1:length(ne)) {
        if(runif(1)<d2[ne[i]+1]) g2<-add.edges(g2,c(dyad[2]-1,ne[i]))
        if(runif(1)<d2[dyad[2]]) g2<-add.edges(g2,c(ne[i],dyad[2]-1))
      }
    }
  }
  g2<-add.edges(g2,dyad-1)
}
g2<-simplify(g2)

# generate/store node statistics for original network
twindeg1[t] <- degree(g1,v=twin1-1,mode="in")
twoutdeg1[t] <- degree(g1,v=twin1-1,mode="out")
twtr1[t] <- transitivity(g1,v=twin1-1,type="local")
twbe1[t] <- betweenness(g1,v=twin1-1,directed=F)

# generate/store node statistics for mirror network
twindeg2[t] <- degree(g2,v=twin2-1,mode="in")
twoutdeg2[t] <- degree(g2,v=twin2-1,mode="out")
twtr2[t] <- transitivity(g2,v=twin2-1,type="local")
twbe2[t] <- betweenness(g2,v=twin2-1,directed=F)
}

corindeg <- cor.test(twindeg1, twindeg2)
coroutdeg <- cor.test(twoutdeg1, twoutdeg2)
cortr <- cor.test(twtr1, twtr2, use="complete.obs")
corbe <- cor.test(twbe1, twbe2)

# Results obtained after 10000 simulations:
corindeg      # 0.46 (0.44,0.48)
coroutdeg     # 0.12 (0.10,0.14)
cortr         # 0.48 (0.46,0.50)
corbe        # 0.29 (0.27,0.31)

```

## Code for “Fitness” model and “Mirror Network” method

```

# R code for fitness model
# using a mirror network method to measure
# the extent to which this model generates
# heritability of network characteristics
#
# NOTE: Many thanks to Referee 2 who wrote
# some of this code

rm(list=ls())

library(igraph) # Thanks to Gabor Csardi for the igraph package!

n<-750 # number of nodes
e<-3150 # number of edges
m<-e/n # average number of edges per node (degree)

T<-10000 # number of simulations

twindeg1 <- rep(0, times=T)
twoutdeg1 <- rep(0, times=T)
twtr1 <- rep(0, times=T)
twbe1 <- rep(0, times=T)

twindeg2 <- rep(0, times=T)
twoutdeg2 <- rep(0, times=T)
twtr2 <- rep(0, times=T)
twbe2 <- rep(0, times=T)

for(t in 1:T) { # do simulation run

  if(t%%100==0) print(t)

  # randomly choose one individual to be a twin
  twin1 <- sample(n,1)
  twin2 <- sample(n,1)

  # create first set of individual traits
  d1<-runif(n) # distribution of fitness

  # create original network
  g1<-graph.empty(n)
  g1<-add.edges(g1,c(0,1,1,0)) # initialize network
  for(i in 2:(n-1)) {
    p<-degree(g1,v=0:(i-1))*d1[1:i]
    p<-m*p/sum(p)
    ins<-which(runif(i)<p)
    if(length(ins)>0) g1<-add.edges(g1,c(rbind(i,ins-1)))
    g1<-simplify(g1)
  }

  # create second set of individual traits
  d2<-runif(n) # distribution of fitness

  # copy genes
  d2[twin2] <- d1[twin1]

  # create mirror network
  g2<-graph.empty(n)
  g2<-add.edges(g2,c(0,1,1,0))
  for(i in 2:(n-1)) {
    p<-degree(g2,v=0:(i-1))*d2[1:i]
    p<-m*p/sum(p)
    ins<-which(runif(i)<p)
    if(length(ins)>0) g2<-add.edges(g2,c(rbind(i,ins-1)))
    g2<-simplify(g2)
  }

  # generate/store node statistics for original network
  twindeg1[t] <- degree(g1,v=twin1-1,mode="in")
  twoutdeg1[t] <- degree(g1,v=twin1-1,mode="out")
  twtr1[t] <- transitivity(g1,v=twin1-1,type="local")
  twbe1[t] <- betweenness(g1,v=twin1-1,directed=F)

  # generate/store node statistics for mirror network

```

```
twindeg2[t] <- degree(g2,v=twin2-1,mode="in")
twoutdeg2[t] <- degree(g2,v=twin2-1,mode="out")
twtr2[t] <- transitivity(g2,v=twin2-1,type="local")
twbe2[t] <- betweenness(g2,v=twin2-1,directed=F)
}

corindeg <- cor.test(twindeg1, twindeg2)
coroutdeg <- cor.test(twoutdeg1, twoutdeg2)
cortr <- cor.test(twtr1, twtr2, use="complete.obs")
corbe <- cor.test(twbe1, twbe2)

# Results obtained after 10000 simulations:
corindeg      # 0.05 (-0.03,0.01)
coroutdeg     # 0.00 (-0.02,0.02)
cortr         # 0.06 ( 0.04,0.08)
corbe        # 0.02 ( 0.00,0.04)
```

## Code for “Social Space” model and “Mirror Network” method

```

# R code for social space model
# using a mirror network method to measure
# the extent to which this model generates
# heritability of network characteristics
#
# NOTE: Many thanks to Referee 2 who wrote
# some of this code

rm(list=ls())

library(igraph) # Thanks to Gabor Csardi for the igraph package!

n<-750 # number of nodes
e<-3150 # number of edges
m<-e/n # average number of edges per node (degree)

T<-10000 # number of simulations

alpha<-1.45
beta<-0.00115

twindeg1 <- rep(0, times=T)
twoutdeg1 <- rep(0, times=T)
twtr1 <- rep(0, times=T)
twbe1 <- rep(0, times=T)

twindeg2 <- rep(0, times=T)
twoutdeg2 <- rep(0, times=T)
twtr2 <- rep(0, times=T)
twbe2 <- rep(0, times=T)

for(t in 1:T) { # do simulation run

  if(t%%100==0) print(t)

  # randomly choose one individual in each network to be a twin
  twin1 <- sample(n,1)
  twin2 <- sample(n,1)

  # create first set of individual traits
  d1<-runif(n) # distribution of distances

  # create original network
  dhh1<-abs(outer(d1,d1,"-"))
  rhh1<-1/(1+(dhh1/beta)^alpha)
  A1<-runif(n^2)<rhh1
  diag(A1)<-F
  g1<-simplify(graph.adjacency(A1,mode="upper"))

  # create second set of individual traits
  d2<-runif(n) # distribution of distances

  # copy genes
  d2[twin2] <- d1[twin1]

  # create mirror network
  dhh2<-abs(outer(d2,d2,"-"))
  rhh2<-1/(1+(dhh2/beta)^alpha)
  A2<-runif(n^2)<rhh2
  diag(A2)<-F
  g2<-simplify(graph.adjacency(A2,mode="upper"))

  # generate/store node statistics for original network
  twindeg1[t] <- degree(g1,v=twin1-1,mode="in")
  twoutdeg1[t] <- degree(g1,v=twin1-1,mode="out")
  twtr1[t] <- transitivity(g1,v=twin1-1,type="local")
  twbe1[t] <- betweenness(g1,v=twin1-1,directed=F)

  # generate/store node statistics for mirror network
  twindeg2[t] <- degree(g2,v=twin2-1,mode="in")
  twoutdeg2[t] <- degree(g2,v=twin2-1,mode="out")
  twtr2[t] <- transitivity(g2,v=twin2-1,type="local")
  twbe2[t] <- betweenness(g2,v=twin2-1,directed=F)

}

```

```
corindeg <- cor.test(twindeg1, twindeg2)
coroutdeg <- cor.test(twoutdeg1, twoutdeg2)
cortr <- cor.test(twtr1, twtr2, use="complete.obs")
corbe <- cor.test(twbe1, twbe2)

# Results obtained after 10000 simulations:
corindeg      # 0.01 (-0.01,0.03)
coroutdeg     # 0.01 (-0.01,0.03)
cortr         # 0.00 (-0.02,0.02)
corbe        # 0.00 (-0.02,0.02)
```

## Code for Exponential Random Graph Model (“ERGM”) and “Mirror Network” method

```

# R code for ERGM
# using a mirror network method to measure
# the extent to which this model generates
# heritability of network characteristics
#
# NOTE: Many thanks to Referee 2 who wrote
# some of this code

rm(list=ls())

library(ergm)
library(network)

n<-750 # number of nodes
e<-3150 # number of edges
m<-e/n # average number of edges per node (degree)

T<-10000 # number of simulations

twindeg1 <- rep(0, times=T)
twoutdeg1 <- rep(0, times=T)
twtr1 <- rep(0, times=T)
twbel <- rep(0, times=T)

twindeg2 <- rep(0, times=T)
twoutdeg2 <- rep(0, times=T)
twtr2 <- rep(0, times=T)
twbe2 <- rep(0, times=T)

for(t in 1:T) { # do simulation run

  if(t%%100==0) print(t)

  # randomly choose one individual in each network to be a twin
  twin1 <- sample(n,1)
  twin2 <- sample(n,1)

  # create first set of individual traits
  d1<-runif(n) # distribution of attractiveness

  # create original network
  g1.use <- network(as.matrix(cbind(sample(n,e,replace=T),sample(n,e,replace=T))))
  g1.use %v% "attract" <- d1 # assign intrinsic characteristic to each node
  g1 <- simulate(~ nodeicov("attract")+ # in degree proportional to intrinsic characteristics
    triadcensus(c(8,11:15)), # count number of transitive triplets
    theta0=c(2.5,rep(2.7,6)), # coefficients 2.5 (for in-degree) and 2.7 (for triplets)
    constraints = ~ edges, # constrain the network to yield e edges
    basis=g1.use, # use basis network to get total number of edges and nodes
    burnin=100000) # let Monte Carlo Chain run a long time before sampling a network

  # create second set of individual traits
  d2<-runif(n) # distribution of attractiveness

  # copy genes to one individual in new network
  d2[twin2] <- d1[twin1]

  # create mirror network
  g2.use <- network(as.matrix(cbind(sample(n,e,replace=T),sample(n,e,replace=T))))
  g2.use %v% "attract" <- d2 # assign intrinsic characteristic to each node
  g2 <- simulate(~ nodeicov("attract")+ # in degree proportional to intrinsic characteristics
    triadcensus(c(8,11:15)), # count number of transitive triplets
    theta0=c(2.5,rep(2.7,6)), # coefficients 2.5 (for in-degree) and 2.7 (for triplets)
    constraints = ~ edges, # constrain the network to yield e edges
    basis=g2.use, # use basis network to get total number of edges and nodes
    burnin=100000) # let Monte Carlo Chain run a long time before sampling a network
}

```

```

# convert networks to igraph graph objects
m1 = g1[,]
m2 = g2[,]

require(igraph)

g1i <- graph.adjacency(m1)
g2i <- graph.adjacency(m2)

# generate/store node statistics for original network
twindeg1[t] <- degree(g1i,v=twin1-1,mode="in")
twoutdeg1[t] <- degree(g1i,v=twin1-1,mode="out")
twtr1[t] <- transitivity(g1i,v=twin1-1,type="local")
twbe1[t] <- betweenness(g1i,v=twin1-1,directed=F)

# generate/store node statistics for mirror network
twindeg2[t] <- degree(g2i,v=twin2-1,mode="in")
twoutdeg2[t] <- degree(g2i,v=twin2-1,mode="out")
twtr2[t] <- transitivity(g2i,v=twin2-1,type="local")
twbe2[t] <- betweenness(g2i,v=twin2-1,directed=F)

detach(package:igraph)
}

corindeg <- cor.test(twindeg1, twindeg2)
coroutdeg <- cor.test(twoutdeg1, twoutdeg2)
cortr <- cor.test(twtr1, twtr2, use="complete.obs")
corbe <- cor.test(twbe1, twbe2)

# Results obtained after 10000 simulations:
corindeg # 0.50 (0.48,0.52)
coroutdeg # 0.12 (0.10,0.14)
cortr # 0.02 (0.00,0.04)
corbe # 0.34 (0.32,0.36)

```