R Code for
Triad-based Comparison and Signatures of Directed Networks

Xiaochuan Xu and Gesine Reinert

September 17, 2018

```
#Generate ER random graphs
#n=500,p=0.1;n=500,p=0.2;n=1000,p=0.05
g=erdos.renyi.game(n,p, directed = TRUE)
g=get.edgelist(g)
write.table(g,paste0("er_",n,"_",p))

#Generate BA random graphs
#n=500,e=1;n=500,e=2;n=1000,e=2
g=barabasi.game(n,m=e, directed = TRUE)
g=get.edgelist(g)
write.table(g,paste0("ba_",n,"_",e))

#Generate Geometric random graphs
#With the same number of edges and nodes as ER networks
#n=500,p=0.1;n=500,p=0.2;n=1000,p=0.05
r=sqrt((n-1)/n*p/pi)
grg.game(n,r,torus = TRUE)->g
g=as.directed(g)
g=get.edgelist(g)
write.table(g,paste0("geo_",n,"_",r^2))


#trd gives the Triad Degree distribution.
#Input: Directed graph g. Output: The Triad matrix M.
#Each row represents a node and
#each column represents an orbit.
#The (i,j)th entry is the number of orbit j that
#node i is involved in.
trd<-function(g){
  A=get.adjacency(g)
```

```
N=vcount(g)
M=matrix(0,N,30)
for (i in 1:N){
  a1=which(A[i,]>0)
  a2=which(A[,i]>0)
  a3=intersect(a1,a2)
  a1=setdiff(a1,a3)
  a2=setdiff(a2,a3)
  n1=length(a1)
  n2=length(a2)
  n3=length(a3)
  if (n1>0){
    for (j in 1:n1){
      b1=which(A[a1[j],]>0)
      b2=which(A[,a1[j]]>0)
      b1=setdiff(b1,c(i,a1,a2,a3))
      b2=setdiff(b2,c(i,a1,a2,a3))
      b3=intersect(b1,b2)
      b1=setdiff(b1,b3)
      b2=setdiff(b2,b3)
      for (k in b1){M[i,14]=M[i,14]+1}
      for (k in b2){M[i,2]=M[i,2]+1}
      for (k in b3){M[i,17]=M[i,17]+1}
    }
    if (n1>1){
      for (j in 1:(n1-1)){
        for (k in (j+1):n1){
          if ((A[a1[j],a1[k]]+A[a1[k],a1[j]])==0){M[i,3]=M[i,3]+1}
          if ((A[a1[j],a1[k]]+A[a1[k],a1[j]])==1){M[i,6]=M[i,6]+1}
          if ((A[a1[j],a1[k]]+A[a1[k],a1[j]])==2){M[i,10]=M[i,10]+1}
        }
      }
    }
    if (n2>0){
      for (j in a1){
        for (k in a2){
          if ((A[j,k]+A[k,j])==0){M[i,13]=M[i,13]+1}
          if (A[j,k]==0&A[k,j]==1){M[i,7]=M[i,7]+1}
          if (A[j,k]==1&A[k,j]==0){M[i,16]=M[i,16]+1}
          if ((A[j,k]+A[k,j])==2){M[i,23]=M[i,23]+1}
        }
      }
    }

  }
  if (n2>0){
    for (j in a2){
      b1=which(A[j,]>0)
      b2=which(A[,j]>0)
      b1=setdiff(b1,c(i,a1,a2,a3))
      b2=setdiff(b2,c(i,a1,a2,a3))
      b3=intersect(b1,b2)
      b1=setdiff(b1,b3)
      b2=setdiff(b2,b3)
      for (k in b1){M[i,4]=M[i,4]+1}
      for (k in b2){M[i,15]=M[i,15]+1}
      for (k in b3){M[i,20]=M[i,20]+1}
    }
    if (n2>1){
      for (j in 1:(n2-1)){
        for (k in (j+1):n2){
          if ((A[a2[j],a2[k]]+A[a2[k],a2[j]])==0){M[i,1]=M[i,1]+1}
          if ((A[a2[j],a2[k]]+A[a2[k],a2[j]])==1){M[i,5]=M[i,5]+1}
          if ((A[a2[j],a2[k]]+A[a2[k],a2[j]])==2){M[i,8]=M[i,8]+1}
        }
      }
    }
    if (n3>0){
      for (j in a2){
        for (k in a3){
          if ((A[j,k]+A[k,j])==0){M[i,18]=M[i,18]+1}
          if (A[j,k]==0&A[k,j]==1){M[i,25]=M[i,25]+1}
          if (A[j,k]==1&A[k,j]==0){M[i,11]=M[i,11]+1}
          if ((A[j,k]+A[k,j])==2){M[i,29]=M[i,29]+1}
        }
      }
    }
  }
  if(n3>0){
    for (j in a3){
      b1=which(A[j,]>0)
      b2=which(A[,j]>0)
      b1=setdiff(b1,c(i,a1,a2,a3))
      b2=setdiff(b2,c(i,a1,a2,a3))
      b3=intersect(b1,b2)
      b1=setdiff(b1,b3)
      b2=setdiff(b2,b3)
      for (k in b1){M[i,21]=M[i,21]+1}
      for (k in b2){M[i,19]=M[i,19]+1}
      for (k in b3){M[i,26]=M[i,26]+1}
    }
    if (n3>1){
          for (j in 1:(n3-1)){
            for (k in (j+1):n3){
              if ((A[a3[j],a3[k]]+A[a3[k],a3[j]])==0){M[i,27]=M[i,27]+1}
              if ((A[a3[j],a3[k]]+A[a3[k],a3[j]])==1){M[i,30]=M[i,30]+1}
              if ((A[a3[j],a3[k]]+A[a3[k],a3[j]])==2){M[i,12]=M[i,12]+1}
            }
          }
        }
        if (n1>0){
          for (j in a1){
            for (k in a3){
              if ((A[j,k]+A[k,j])==0){M[i,22]=M[i,22]+1}
              if (A[j,k]==0&A[k,j]==1){M[i,9]=M[i,9]+1}
              if (A[j,k]==1&A[k,j]==0){M[i,24]=M[i,24]+1}
              if ((A[j,k]+A[k,j])==2){M[i,28]=M[i,28]+1}
            }
          }
        }
      }
    }
  return(M)
}

#NNscore gives the Nearest Neighbour score.
#Input: Dissimilarity matrix d. Output: NNscore
NNscore<-function(d){
  n=ncol(d)
  diag(d)=max(d)
  t=0
  for (j in 1:n){
    i=which.min(d[,j])
    if (abs(i-j)==1){
      t=t+1
    }
  }
  return(t/n)
}

#N2Nscore gives the Nearest Neighbour score.
#Input: Dissimilarity matrix d. Output: N2Nscore
N2Nscore<-function(d){
  n=ncol(d)
  m=max(d)
  diag(d)=m
  t=0
  for (j in c(1,n)){
    i=which.min(d[,j])
    if (abs(i-j)==1){
      t=t+1
    }
  }
  for (j in 2:n-1){
    i1=which.min(d[,j])
    d[i1,j]=m
    i2=which.min(d[,j])
    if ((abs(i1-j)==1)&(abs(i2-j)==1)){
      t=t+1
    }
  }
  return(t/n)
}

#NN gives the nearest neighbour.
#Input: Dissimilarity matrix d. Output: Nearest neighbour.
NN<-function(d){
  n=ncol(d)
  diag(d)=max(d)
  t=c()
  for (j in 1:n){
    i=which.min(d[,j])
    t=c(t,i)
  }
  return(t)
}

#N2N gives the nearest two neighbours.
#Input: Dissimilarity matrix d. Output: Nearest two neighbours.
N2N<-function(d){
  n=ncol(d)
  m=max(d)
  diag(d)=m
  t=c()
  for (j in 1:n){
    i1=which.min(d[,j])
    d[i1,j]=m
    i2=which.min(d[,j])
    t=rbind(t,c(i1,i2))
  }
  return(t)
}

#Find the signature triads
```

```
ty1=matrix(0,n,n)
for (i in 1:(n-1)){
  for (j in (i+1):n){
    ty1[i,j]=which.max(abs(e[i,]-e[j,]))+3
    ty1[i,j]=ty1[i,j]*sign(e[i,ty1[i,j]-3]-e[j,ty1[i,j]-3])
  }
}
s=matrix(0,31,33)
for (i in 1:(n-1)){
  for (j in (i+1):n){
    a=ty1[i,]
    b=ty1[j,]
    a=sum(which(a==ty1[i,j]))
    b=sum(which(b==-ty1[i,j]))
    if (a>=b){s[i,ty1[i,j]+17]=s[i,ty1[i,j]+17]+1}
    if (a<=b){s[j,-ty1[i,j]+17]=s[j,ty1[i,j]+17]+1}
  }
}
l=matrix(0,31,16)
for (j in 1:16){
  l[,j]=s[,j+17]-s[,17-j]
}
```