```
data<-read.csv(file="C:\\Users\\Charlie\\Desktop\\data.csv")

           miss<--999                    # Code was revised 7/24/2017

           serror=1

           nact=nrow(data)
         npar=ncol(data)
         ntot=0
           for (i in 1:nact) {
           for (j in 1:npar) {
           if (data[i,j]!=miss) ntot<-ntot+1 }}

           inde<-matrix(0,ntot/2,2)
           dv<-rep(0,ntot)
           k<-0
           l<-nact/2+1
           print (l)
         bata<-matrix(0,nact,npar)
         for (i in 1:nact) {
         for (j in 1:npar) {
         if (data[i,j]!=miss)
           {k<-k+1
            dv[k]<-data[i,j]
            bata[i,j]<-1
                             if (i<l) {inde[k,1]<-i   #For each data
point, first
                          inde[k,2]<-j} }             #and second
subscripts.
           } }
         ntot<-k/2
         nact<-nact/2
         ac<-matrix(0,ntot,nact)
           pa<-matrix(0,ntot,npar)
         for (k in 1:ntot) {
         ac[k,inde[k,1]]<-1      # indicator matrices for
           pa[k,inde[k,2]]<-1 }    # actor and partner

           apar<-colSums(ac)
           pact<-colSums(pa)


           mirr<-rep(0,nact)
           recip<-matrix(0,nact,npar)
         aa<-matrix(0,ntot,(6*ntot))  #matrix representation of SCP
         vv<-matrix(0,ntot,(6*ntot))  #var-cov matrix of observed scores
           uvv<-matrix(0,ntot,ntot)
           j0<-c(1:(6*ntot))
         j0[(2*ntot+1):(3*ntot)]<-c((3*ntot+1):(4*ntot))
         j0[(3*ntot+1):(4*ntot)]<-c((2*ntot+1):(3*ntot))



       for (i in 1:ntot) {
```

```
  aa[i,i]<-1
  vv[i,i]<-1
ii<-ntot+i
  aa[i,ii]<-1
  vv[i,ii]<-1
ii<-4*ntot+i
  aa[i,ii]<-1
  vv[i,ii]<-1 }


 for (i in 1:ntot) {
  for (j in (4*ntot+1):(5*ntot)) {
       aa[i,j]<-aa[i,j]-1/ntot } }

for (i in 1:ntot) {
aa[i,i]<-1-1/pact[inde[i,2]]
aa[i,(ntot+i)]<-1-1/apar[inde[i,1]]
for (j in 1:ntot)  {
  if ((inde[i,1]!=inde[j,1]) | (inde[i,2]!=inde[j,2])) {

if ((inde[i,1]==inde[j,2]) & (inde[i,2]==inde[j,1]))
                {recip[inde[i,1],inde[i,2]]<-1
                 aa[i,(5*ntot+i)]<-1
                 aa[i,(5*ntot+j)]<--1
                 vv[i,(2*ntot+j)]<-1
               vv[i,(3*ntot+j)]<-1
                 vv[i,(5*ntot+j)]<-1}

jj<-j+ntot
if (inde[i,1]==inde[j,1]) aa[i,jj]<--1/apar[inde[i,1]]
if (inde[i,1]==inde[j,1]) vv[i,j]<-1
if (inde[i,2]==inde[j,2]) vv[i,jj]<-1
if (inde[i,2]==inde[j,2]) aa[i,j]<--1/pact[inde[i,2]]
if (inde[i,1]==inde[j,2]) vv[i,(2*ntot+j)]<-1
if (inde[i,2]==inde[j,1]) vv[i,(3*ntot+j)]<-1
                           } } }

mirr<-rowSums(recip)
  for (i in 1:ntot) {
  for (j in 1:ntot) {
if ((recip[inde[i,1],inde[i,2]]!=0) &
(recip[inde[j,1],inde[j,2]]!=0)) {
  if ((inde[i,1]==inde[j,2]) & (inde[i,2]==inde[j,1]))
          {    aa[i,(3*ntot+j)]<-(mirr[inde[i,1]]-
1)/mirr[inde[i,1]]
             aa[i,(2*ntot+j)]<-(mirr[inde[j,1]]-1)/mirr[inde[j,1]]
}
if ((inde[i,1]!=inde[j,2]) & (inde[i,2]==inde[j,1]))
aa[i,(2*ntot+j)]<--1/mirr[inde[j,1]]
if ((inde[i,1]==inde[j,2]) & (inde[i,2]!=inde[j,1]))
aa[i,(3*ntot+j)]<--1/mirr[inde[i,1]]
           } } }
        uvv<-vv[,(2*ntot+1):(3*ntot)]+vv[,(3*ntot+1):(4*ntot)]
  ss<-matrix(0,6,3)      # Sums of Squares and Cross-Products
```

```
          c<-matrix(0,6,6)       # Searle, Casella, & McCullough, eq 18,
p. 173
         w<-matrix(0,6,276)     # Var-Covar matrix for SS and SCP
         uw<-matrix(0,6,126)
         zz<-matrix(0,ntot,ntot)
           uu<-matrix(0,ntot,ntot)


       for (i in 1:6) {
           dumi<-((i-1)*ntot+1):(i*ntot)
             if ((i<3) | (i>4))
               {  for (j in 1:6)  {
                dumj<-((j-1)*ntot+1):(j*ntot)
                  c[i,j]<-sum(diag(aa[,dumi]%*%vv[,dumj])) }
                       }
        if ((i==3) | (i==4))
                    {zz<-aa[,dumi]
                      for (j in 1:6) {
                     dumj<-((j-1)*ntot+1):(j*ntot)
                   uu<-vv[,dumj]
                     c[i,j]<-.5*(sum(diag(zz%*%t(uu))) +
sum(diag((t(zz)%*%uu))) ) }  }  }

         cin<-solve(c)
              for (i in 1:6)
            { dumi<-((i-1)*ntot+1):(i*ntot)
         ss[i,1]<-t(dv[1:ntot])%*%aa[,dumi]%*%dv[1:ntot]
         ss[i,2]<-
t(dv[(ntot+1):(2*ntot)])%*%aa[,dumi]%*%dv[(ntot+1):(2*ntot)]
           ss[i,3]<-t(dv[1:ntot])%*%aa[,dumi]%*%dv[(ntot+1):(2*ntot)] }
         est<-matrix(0,6,3)
         est<-cin%*%ss          # Here are the parameter estimates

         if (serror==1)   {

           cwu=matrix(0,6,6)
           j=1
           foo=c(0,6,12,18,24,30)
         for (i in foo) {
                   cwu[1,j]=i
                   cwu[j,1]=i
                   j=j+1 }
              j=2
           foo=c(36,42,48,54,60)
           for (i in foo) {
         cwu[2,j]=i
         cwu[j,2]=i
         j=j+1          }
         j=3
           foo=c(66,72,78,84)
         for (i in foo) {
         cwu[3,j]=i
         cwu[j,3]=i
         j=j+1 }
```

```
          j=4
            foo=c(90,96,102)
          for (i in foo) {
          cwu[4,j]=i
          cwu[j,4]=i
          j=j+1 }

            cwu[5,5]=108
          cwu[6,5]=114            # an accounting variable,
            cwu[5,6]=114             # cwu will point us to the
            cwu[6,6]=120             # right parms, SS, and SCP



                                     # Here comes the time-consuming
computation
                                     #    So here I start playing with the
code
for (i1 in 1:6)
 { dumi1=(ntot*(i1-1)+1):(i1*ntot)
     w1=0
for (j1 in 1:6)  {
          startw=w1
   dumj1=(ntot*(j1-1)+1):(j1*ntot)
   garmat1<- aa[,dumi1] %*% vv[,j0[dumj1]]
   garmat2<- aa[,j0[dumi1]] %*% vv[,dumj1]
   garmat3<- aa[,dumi1] %*%  vv[,dumj1]
for (i2 in 1:6)                {
   dumi2=(ntot*(i2-1)+1):(i2*ntot)
              w1<-startw+i2-6
   barmat1<- garmat1 %*% aa[,dumi2]
   barmat2<- garmat2 %*% aa[,j0[dumi2]]
   barmat3<- garmat3 %*% aa[,dumi2]
            for (j2 in j1:6) {
                 w1=w1+6
            dumj2=(ntot*(j2-1)+1):(j2*ntot)
            cons=.5
            if (j2!=j1) cons=1
          w[i1,w1]=w[i1,w1]+cons*sum(diag(barmat1 %*% vv[,j0[dumj2]]))
            w[i1,w1]=w[i1,w1]+cons*sum(diag(barmat2 %*% vv[,dumj2]))
                             }


                  for (j2 in 1:6)  {          # setting up var-cov
                jj2=j2                      # matrix for SS & SCP
                if ((j2==3) | (j2==4)) jj2=7-j2
                dumj2=(ntot*(jj2-1)+1):(jj2*ntot)
                w2=cwu[j1,j2]+i2
                  uw[i1,w2]=uw[i1,w2]+2*sum(diag(barmat3 %*% vv[,dumj2]))
}
                                      }   # end i2 loop
}                                         # end j1 loop
   for (i2 in 1:6) {
      dumi2=(ntot*(i2-1)+1):(i2*ntot)
```

```
   w1=i2+120

        for (j1 in 1:5)    {
          jj1=j1
          if (j1>3) jj1=j1+1
            uu=vv[,(ntot*(jj1-1)+1):(jj1*ntot)]
          if (j1==3) uu=uvv
        for (j2 in 1:5)  {
          w1=w1+6
          jj2=j2
          if (jj2>3) jj2=j2+1
          zz=vv[,(ntot*(jj2-1)+1):(jj2*ntot)]
          if (j2==3) zz=uvv

w[i1,w1]=w[i1,w1]+sum(diag(aa[,j0[dumi1]]%*%uu%*%aa[,dumi2]%*%zz))
                        }  }
   }  }                                  # End i2 and i1 loops and end
time-consuming computation

        b=matrix(0,46,46)
        var=matrix(0,6,6)

        for (j1 in 1:46) {
        dum=((j1-1)*6+1):(j1*6)
        var=cin%*%w[,dum]%*%t(cin)    # Searle, Casella, & McCullough
        r1=0                          # Equation 21, page 176
               for (i1 in 1:6)  {
               for (i2 in i1:6) {
                 r1=r1+1
               b[r1,j1]=var[i1,i2] } }

        if (j1<22) {
        r2=21
        var=cin%*%uw[,dum]%*%t(cin)
            for (i1 in 1:5)     {
            ii1=i1
          if (i1>3) ii1=i1+1
            for (i2 in 1:5)  {
          ii2=i2
          if (i2>3) ii2=i2+1      # Accommodating the
            r2=r2+1                 #  asymmetrical SCPs
          bare=var[ii1,ii2]
          if (i1==3) bare=(var[3,ii2]+var[4,ii2])/2
          if (i2==3) bare=(var[ii1,3]+var[ii1,4])/2
          if ((i1==3) & (i2==3))
bare=(var[3,3]+var[3,4]+var[4,3]+var[4,4])/4
            b[r2,j1]=bare     }   }  # b gives exact vars and covs
                                  #   among parameter estimates
                  }  }

          bi=diag(46)
          bi=b+bi
            bi=solve(bi)               #  the biggest inversion
```

```
           bb<-matrix(0,6,46)
           dn<-c(1,7,12,16,19,21)
           for (i in 1:6) {
           bb[i,]<-b[dn[i],] }

           bb= bb %*% bi

                                                  # Having set everything up,
                                                  #  we now begin using the
data

           oot<-matrix(0,6,3)
           kk=3
           estp<-rep(0,46)
                                                                  #Computing
products of estimates
                                                                  # via the R
operator for outer product:   %o%
          xvec<-c(est[1,1],est[2,1],est[3,1],est[5,1],est[6,1])
          yvec<-c(est[1,2],est[2,2],est[3,2],est[5,2],est[6,2])

          for (kk in 1:3) {
          xyvec<-
c(est[1,kk],est[2,kk],est[3,kk],est[4,kk],est[5,kk],est[6,kk])
          matr<-xyvec %o% xyvec
          estp[1:21]<-
c(matr[,1],matr[2:6,2],matr[3:6,3],matr[4:6,4],matr[5:6,5],matr[6,6])
          matrx<- xvec %o% xvec
          if (kk==2) matrx<- yvec %o% yvec
          if (kk==3) matrx<- yvec %o% xvec
          estp[22:46]<-c(matrx)

           oot[,kk] = bb %*% estp   }}      # oot gives estimated
variances and covs
         oot = oot**.5    # converting var to se

         print (est)
         if (serror==1) print (oot)
```