The Reciprocal One-with-many Design for Indistinguishable Partners

Avraham N. (Avi) Kluger
School of Business Administration, The Hebrew University of Jerusalem
Mt. Scopus, Jerusalem 91905 ISRAEL
avik@savion.huji.ac.il
and
Wolfgang Viechtbauer
Department of Psychiatry and Neuropsychology, School for Mental Health and Neuroscience
Faculty of Health, Medicine, and Life Sciences,
Maastricht University, P.O. Box 616 (VIJV1), 6200 MD Maastricht, The Netherlands
wvb (at) wvbauer (dot) com

February 15, 2019

Abstract

This blog shows how to compute, in R, actor and partner variances, as well as generalized and dyadic reciprocities for *reciprocal one-with-many design for indistinguishable partners*. We show here how to code this model using MLM approach with the *nlme* package and using SEM approach using the *lavaan* package. The MLM approach is more verstaile as it does not assume a balanced design (each focal person must have the same number of partners), while the SEM approach may be easy for follow-up analysis, such as using predictors and outcomes for actor and partner effects.

Motivation

Kenny et al., (2006) demonstrate on pages 290-293 how to compute the social relations model (SRM) for the case of *reciprocal one-with-many design for indistinguishable partners*. This design is relevant, for example, when one collects data from therapists and their patients, where each therapist rate several patients and all patients rate their own therapist on the same variables (e.g., working alliance). To demonstrate the use of this model, Kenny et al. (2006) used data regarding attachment anxiety that a mother experienced with three family members, and the attachment anxiety experienced by each of these members with the mother, in 208 families. On page 292, the authors analyze this model with *MLwiN* and report that the actor variance is 0.207, the partner variance is 0.060, and the actor-partner correlation is .70. In addition, David Kenny shows on his web site how to analyze this model with SPSS. Specifically, this code

```
MIXED
  Outcome BY role WITH focalcode partcode
  /FIXED = focalcode partcode | NOINT
  /PRINT = SOLUTION TESTCOV
  /RANDOM focalcode partcode | SUBJECT(focalid) COVTYPE(UNR)
  /REPEATED = role | SUBJECT(focalid*dyadid) COVTYPE(UNR).
```

reproduces, **in bold**, the results from the book. Interestingly, this output also includes the correlation between the errors, which represent the dyadic correlation, .24, *in italics*.

Estimates of Covariance Parameters[a]

Parameter	_	Estimate	Std. Error	Wald Z	Sig.	95% CI Lower Bound	95% CI Upper Bound
Repeated Measures	Var(1)	.423155	.029341	14.422	.000	.369385	.484753
_	Var(2)	.549234	.038083	14.422	.000	.479444	.629184
_	Corr(2,1)	*.239029*	.046228	5.171	.000	.146585	.327334
focalcode + partcode [subject = focalid]	Var(1)	**.060898**	.027134	2.244	.025	.025430	.145838
_	Var(2)	**.208409**	.035715	5.835	.000	.148952	.291601
_	Corr(2,1)	**.698818**	.170996	4.087	.000	.206931	.908699

[a] Dependent Variable: outcome.

The goal of this post is to demonstrate how to reproduce the above results in R.

Acknowledgement

This post is the result of the first author asking in r-sig-mixed-models@r-project.org how to analyze this model with MLM. The second author showed first how to do so with the package *metafor* (not shown here), and eventually with *nlme*. The first author showed how to perform this analyses with *lavaan*.

The first author thanks David Kenny for clarifying the meaning of the error covariance (dyadic reciprocity); Limor Borut for figuring out the proper *lme4* code; James Uanhoro for pointing out the correct syntax for specifying correlations; Thierry Onkelinx for pointing out the uncertainty about the variance estimates; Ben Bolker for pointing out that setting residual to zero is difficult in *lme4*, and how to get results with with *nlme* by constraining residual to zero (which could be avoided, as done here, by specifying correlated residuals).

Read (in SPSS format) from Kenny's book site and replicate Table 9.1

```
# To read the SPSS data from Kenny's site:
if (!require('foreign')) install.packages('foreign'); library('foreign')
```

```
## Loading required package: foreign
```

```
Chapter10_df <- read.spss("http://davidakenny.net/kkc/c10/c10_recip.sav",
               to.data.frame = TRUE, use.value.labels = FALSE)

# Read copy of the data from GitHub
# Chapter10_df <- read.csv(paste0("https://raw.githubusercontent.com/",
#          "avi-kluger/RCompanion4DDABook/master/Chapter%2010/Chapter10_df.csv"))
```

Very Important Note. The original data coded with 0 the focal person. Therefore, the first random variable above is partner variance. Reversing the codes below make the results more intuitive. We thank David Kenny for clarifying this issue.

```
Chapter10_df$focalcode <- 1- Chapter10_df$focalcode
Chapter10_df$partcode  <- 1- Chapter10_df$partcode
head(Chapter10_df, 20)
```

```
##    focalid role dyadid  outcome focalcode partcode obs
## 1        3    1     31 1.000000         1        0 311
## 2        3    2     31 1.000000         0        1 312
## 3        3    1     32 1.500000         1        0 321
## 4        3    2     32 1.000000         0        1 322
## 5        3    1     33 1.500000         1        0 331
## 6        3    2     33 3.833333         0        1 332
## 7        5    1     51 2.166667         1        0 511
## 8        5    2     51 2.333333         0        1 512
## 9        5    1     52 2.000000         1        0 521
## 10       5    2     52 3.500000         0        1 522
## 11       5    1     53 2.166667         1        0 531
## 12       5    2     53 2.000000         0        1 532
## 13       6    1     61 1.166667         1        0 611
## 14       6    2     61 1.000000         0        1 612
## 15       6    1     62 1.333333         1        0 621
## 16       6    2     62 1.166667         0        1 622
## 17       6    1     63 1.000000         1        0 631
## 18       6    2     63 3.666667         0        1 632
## 19       8    1     81 3.333333         1        0 811
## 20       8    2     81 1.166667         0        1 812
```

nlme solution

```
if (!require("nlme")) install.packages("nlme"); suppressMessages(library(nlme))
```

```
## Loading required package: nlme
```

```
mlm <- lme(outcome ~  0 + focalcode + partcode,
           random = ~ 0 + focalcode + partcode | focalid,
           correlation = corSymm(form = ~ 1 | focalid / dyadid),
           weights = varIdent(form = ~ 1 | role),
           data = Chapter10_df)
summary(mlm)
```

```
## Linear mixed-effects model fit by REML
##  Data: Chapter10_df
##        AIC      BIC    logLik
##   2842.207 2883.228 -1413.103
##
## Random effects:
##  Formula: ~0 + focalcode + partcode | focalid
##  Structure: General positive-definite, Log-Cholesky parametrization
##           StdDev    Corr
## focalcode 0.4565201 foclcd
## partcode  0.2467782 0.699
## Residual  0.6505052
##
## Correlation Structure: General
##  Formula: ~1 | focalid/dyadid
##  Parameter estimate(s):
##  Correlation:
##   1
## 2 0.239
```

```
## Variance function:
##  Structure: Different standard deviations per stratum
##  Formula: ~1 | role
##  Parameter estimates:
##        1        2
## 1.000000 1.139271
## Fixed effects: outcome ~ 0 + focalcode + partcode
##              Value  Std.Error   DF  t-value p-value
## focalcode 1.807695 0.04098915 1039 44.10180       0
## partcode  1.698269 0.03424858 1039 49.58655       0
##  Correlation:
##          foclcd
## partcode 0.401
##
## Standardized Within-Group Residuals:
##        Min         Q1        Med         Q3        Max
## -2.1516411 -0.6652762 -0.2491716  0.4600343  4.1404815
##
## Number of Observations: 1248
## Number of Groups: 208
```

```
cis <- intervals(mlm)
cis$fixed
```

```
##              lower     est.    upper
## focalcode 1.727264 1.807695 1.888126
## partcode  1.631065 1.698269 1.765473
## attr(,"label")
## [1] "Fixed effects:"
```

```
### focalid Var(1) and Var(2) and Corr(2,1) and CI
### That is, actor effect, partner effect, and generalized reciprocity
VarCorr(mlm)
```

```
## focalid = pdLogChol(0 + focalcode + partcode)
##           Variance   StdDev    Corr
## focalcode 0.20841061 0.4565201 foclcd
## partcode  0.06089948 0.2467782 0.699
## Residual  0.42315700 0.6505052
```

```
# CI for actor and partner variance
(cis$reStruct$focalid[1:2, ])^2
```

```
##                     lower       est.     upper
## sd(focalcode) 0.14891633 0.20841061 0.2916737
## sd(partcode)  0.02536454 0.06089948 0.1462178
```

```
# CI for generalized reciprocity
cis$reStruct$focalid[3, ]
```

```
##                            lower      est.     upper
## cor(focalcode,partcode) 0.201806 0.6988213 0.9096292
```

```
### dyadid Var(1) and Var(2)
sigma(mlm)^2
```

```
## [1] 0.423157
```

```
coef(mlm$modelStruct$varStruct, unconstrained=FALSE)^2 * sigma(mlm)^2
```

```
##         2
## 0.5492319
```

```
### dyadid Corr(2,1) and CI
### That is, dyadic reciprocity and its CI
coef(mlm$modelStruct$corStruct, unconstrained=FALSE)
```

```
## [1] 0.2390297
```

```
cis$corStruct
```

```
##              lower      est.     upper
## cor(1,2) 0.1463309 0.2390297 0.3275672
## attr(,"label")
## [1] "Correlation structure:"
```

```
# Organize the relevant results into a table
SRMTable <- as.data.frame(matrix(NA, 6, 4))
colnames(SRMTable) <- c("Parameter", "Estimate", "CI95.LL", "CI95.UL")
SRMTable$Parameter <- c("Actor variance", "Partner variance",
                        "Generalized Reciprocity",
                        "Focal person dyadic variance + error",
                        "Partner dyadic variance + error"," Dyadic Reciprocity")

SRMTable[1:2, 2:4] <- cis$reStruct$focalid[1:2,  c("est.", "lower", "upper")]^2
SRMTable[3, 2:4]   <- cis$reStruct$focalid[3   ,  c("est.", "lower", "upper")]
SRMTable[4, 2:4]   <- as.data.frame(cis$sigma^2)[c("est.", "lower", "upper"), ]
SRMTable[5, 2:4]   <- as.data.frame(cis$varStruct^2)[,
                              c("est.", "lower", "upper")]*SRMTable[4, 2]
SRMTable[6, 2:4]   <- cis$corStruct[1, c("est.", "lower", "upper")]

library(knitr)
SRMTable[, 2:4] <- round(SRMTable[, 2:4], 3)
kable(SRMTable, caption =
"SRM estimates for the reciprocal one-with-many design for indistinguishable partners")
```

Table 2: SRM estimates for the reciprocal one-with-many design for indistinguishable partners

Parameter	Estimate	CI95.LL	CI95.UL
Actor variance	0.208	0.149	0.292
Partner variance	0.061	0.025	0.146
Generalized Reciprocity	0.699	0.202	0.910
Focal person dyadic variance + error	0.423	0.369	0.485
Partner dyadic variance + error	0.549	0.455	0.662
Dyadic Reciprocity	0.239	0.146	0.328

As can be seen in the results above, the results for the parameters from *nlme* are identical to the SPSS results. In the Table below, we show the difference between the SPSS and the *nlme* results. The estimates of of most *CI* are practically the same. In the correlations, the largest difference amounting to half of a correlation point (a negligible difference). For the error variance of the partners, the CI from *nlme* are wider than in SPSS. It is possible to obtain the same CIs produced by SPSS in *R* by resorting to the *metafor* package. However, this solution takes few hours to converge, and unless the *CI* for the *partner dyadic variance + error* are needed, this is not recommended. The reason that the CI for the *partner dyadic variance + error* computed by *nlme* differ from SPSS, and the way to compute CIs with *metafor* are explained in the Appendix.

```
SPSSTable <- SRMTable
SPSSTable [1, 2:4] <- c(.208409, .148952, .291601)
SPSSTable [2, 2:4] <- c(.060898, .025430, .145838)
SPSSTable [3, 2:4] <- c(.698818, .206931, .908699)
SPSSTable [4, 2:4] <- c(.423155, .369385, .484753)
SPSSTable [5, 2:4] <- c(.549234, .479444, .629184)
SPSSTable [6, 2:4] <- c(.239029, .146585, .327334)


SPSSTable[2:4] <- round((SPSSTable[, 2:4]) - SRMTable[, 2:4], 3)
kable(SPSSTable, caption = "Subtraction of MLM results from SPSS results")
```

Table 3: Subtraction of MLM results from SPSS results

Parameter	Estimate	CI95.LL	CI95.UL
Actor variance	0	0.000	0.000
Partner variance	0	0.000	0.000
Generalized Reciprocity	0	0.005	-0.001
Focal person dyadic variance + error	0	0.000	0.000
Partner dyadic variance + error	0	0.024	-0.033
Dyadic Reciprocity	0	0.001	-0.001

SEM solution with *lavaan*

This solution is based on *fSRM* package. *fSRM* is designed for family social relations model for round robin with roles. We used their *lavaan* code to model one-with-many design and applied constraints on the roles to be equal to force the model to be indistinguishable. This is a viable solution if the data are balanced (every focal person has the same number of partners). Note that this analysis requires to input data in a wide format. The code below reads the data already set in a wide format. The first output demonstrates a model for distinguishable partners (cf. p. 293); the second output replicates the MLM results above (note that due the constraints, the dyadic covariance and the variances are printed three times with the same value).

```
# Read (in SPSS format) from Kenny's book site and replicate Table 9.1
library('foreign')

# Read data from David Kenny's site
table9.1_df <- read.spss("http://davidakenny.net/kkc/c9/4person.sav",
               to.data.frame = TRUE, use.value.labels = FALSE)

# Read same data from GitHub
# table9.1_df <- read.spss(paste0("https://github.com/avi-kluger/",
#                 "RCompanion4DDABook/blob/master/4person.sav?raw=true"),
#                 to.data.frame = TRUE, use.value.labels = FALSE)

head(table9.1_df)
```

```
##   id2    mfanx    mcanx    myanx    fmanx    fcanx    fyanx    cmanx
## 1   1 1.000000 1.500000 1.500000 1.000000 1.500000 1.833333 1.000000
## 2   2 2.166667 2.000000 2.166667 2.333333 2.166667 2.166667 3.500000
## 3   3 1.166667 1.333333 1.000000 1.000000 1.333333 2.000000 1.166667
## 4   4 3.333333 2.166667 3.166667 1.166667 1.333333 2.000000 2.333333
```

```
## 5   5 1.166667 1.000000 1.000000 1.333333 1.333333 1.333333 1.000000
## 6   6 1.500000 1.833333 2.000000 2.833333 2.000000 1.833333 1.333333
##      cfanx    cyanx    ymanx    yfanx    ycanx
## 1 1.000000 1.166667 3.833333 3.500000 1.500000
## 2 2.333333 3.666667 2.000000 2.000000 2.000000
## 3 1.000000 1.930556 3.666667 3.333333 1.833333
## 4 2.500000 3.000000 2.329596 2.000000 1.000000
## 5 1.000000 1.000000 2.000000 1.333333 1.666667
## 6 2.000000 2.166667 2.000000 2.833333 2.500000
```

```
if (!require("weights")) install.packages("weights");
suppressPackageStartupMessages(library(weights))
```

```r
round(as.dist(cor(table9.1_df[, -1])), 2)
```

```
##       mfanx mcanx myanx fmanx fcanx fyanx cmanx cfanx cyanx ymanx yfanx
## mcanx  0.36
## myanx  0.31  0.35
## fmanx  0.32  0.08  0.08
## fcanx  0.08  0.24  0.07  0.32
## fyanx  0.05  0.04  0.25  0.28  0.50
## cmanx  0.26  0.38  0.08  0.14  0.17  0.03
## cfanx  0.30  0.25  0.11  0.08  0.27  0.09  0.56
## cyanx  0.19  0.24  0.13  0.05  0.15  0.20  0.54  0.55
## ymanx  0.20  0.14  0.24  0.10  0.00  0.15  0.16  0.07  0.22
## yfanx  0.14  0.08  0.20  0.05  0.02  0.26  0.10  0.16  0.28  0.53
## ycanx  0.05  0.16  0.07  0.04  0.01  0.06  0.14  0.15  0.29  0.40  0.35
```

```r
if (!require("lavaan")) install.packages("lavaan");
suppressPackageStartupMessages(library(lavaan))

OneWithManyDistinguishable <- '
# Actor effects:
        Actor   =~ 1*mfanx + 1*mcanx + 1*myanx

# Partner effects:
        Partner =~ 1*fmanx + 1*cmanx + 1*ymanx

# Generalized reciprocity:
        Actor   ~~ Partner

# Dyadic reciprocity:
        mfanx  ~~  fmanx
        mcanx  ~~  cmanx
        myanx  ~~  ymanx
'
# Estimate the model
fitOneWithManyDistinguishable <- sem(OneWithManyDistinguishable,
                 data = table9.1_df,
                 orthogonal = TRUE,
                 mimic = "EQS")
# Examine the model.
summary(fitOneWithManyDistinguishable,
        fit.measures = TRUE, standardized = TRUE)
```

```
## lavaan (0.6-1) converged normally after  20 iterations
##
##   Number of observations                           208
##
##   Estimator                                         ML
##   Model Fit Test Statistic                       8.184
##   Degrees of freedom                                 9
##   P-value (Chi-square)                           0.516
##
## Model test baseline model:
##
##   Minimum Function Test Statistic              145.879
```

```
##   Degrees of freedom                                15
##   P-value                                        0.000
##
## User model versus baseline model:
##
##   Comparative Fit Index (CFI)                    1.000
##   Tucker-Lewis Index (TLI)                       1.010
##
## Loglikelihood and Information Criteria:
##
##   Loglikelihood user model (H0)              -1369.971
##   Loglikelihood unrestricted model (H1)      -1365.859
##
##   Number of free parameters                         12
##   Akaike (AIC)                                2763.942
##   Bayesian (BIC)                              2803.992
##   Sample-size adjusted Bayesian (BIC)         2765.971
##
## Root Mean Square Error of Approximation:
##
##   RMSEA                                          0.000
##   90 Percent Confidence Interval          0.000  0.073
##   P-value RMSEA <= 0.05                          0.817
##
## Standardized Root Mean Square Residual:
##
##   SRMR                                           0.040
##
## Parameter Estimates:
##
##   Information                                 Expected
##   Information saturated (h1) model          Structured
##   Standard Errors                             Standard
##
## Latent Variables:
##                    Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##   Actor =~
##     mfanx             1.000                               0.455    0.523
##     mcanx             1.000                               0.455    0.635
##     myanx             1.000                               0.455    0.579
##   Partner =~
##     fmanx             1.000                               0.271    0.298
##     cmanx             1.000                               0.271    0.435
##     ymanx             1.000                               0.271    0.372
##
## Covariances:
##                    Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##   Actor ~~
##     Partner           0.078    0.022    3.563    0.000    0.636    0.636
##  .mfanx ~~
##    .fmanx             0.175    0.052    3.335    0.001    0.175    0.271
##  .mcanx ~~
##    .cmanx             0.092    0.030    3.076    0.002    0.092    0.296
##  .myanx ~~
```

```
##    .ymanx             0.062    0.037    1.669    0.095    0.062    0.142
##
## Variances:
##                    Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##    .mfanx             0.550    0.064    8.626    0.000    0.550    0.726
##    .mcanx             0.306    0.042    7.286    0.000    0.306    0.596
##    .myanx             0.411    0.051    8.054    0.000    0.411    0.665
##    .fmanx             0.757    0.080    9.457    0.000    0.757    0.911
##    .cmanx             0.315    0.039    8.030    0.000    0.315    0.811
##    .ymanx             0.458    0.052    8.826    0.000    0.458    0.862
##     Actor             0.207    0.035    5.956    0.000    1.000    1.000
##     Partner           0.074    0.025    2.979    0.003    1.000    1.000
```

```
OneWithManyIndistinguisableReplicateSPSS <- '
# Actor effects:
    mActor   =~ 1*mfanx + 1*mcanx + 1*myanx

# Partner effects:
    mPartner =~ 1*fmanx + 1*cmanx + 1*ymanx

# Fix variances to equality with focal person (a) and partner (p)
    mcanx ~~ a* mcanx
    mfanx ~~ a* mfanx
    myanx ~~ a* myanx

    cmanx ~~ p* cmanx
    fmanx ~~ p* fmanx
    ymanx ~~ p* ymanx

# Generalized reciprocity:
    mActor   ~~ gr*mPartner

# Dyadic reciprocity:
    mfanx  ~~  dr*fmanx
    mcanx  ~~  dr*cmanx
    myanx  ~~  dr*ymanx

# Variance labels

    mActor   ~~ Actor  *mActor
    mPartner ~~ Partner*mPartner

# Fix intercepts to equality within focal person and partner
    mcanx ~ ia * 1
    mfanx ~ ia * 1
    myanx ~ ia * 1

    cmanx ~ ip * 1
    fmanx ~ ip * 1
    ymanx ~ ip * 1
'

# Estimate the model
fitOneWithManyIndistinguisableReplicateSPSS <-
```

```r
                 sem(OneWithManyIndistinguisableReplicateSPSS,
                 data = table9.1_df,
                 orthogonal = TRUE,
                 mimic = "EQS",
                 estimator ="MLM")
# Examine the model
summary(fitOneWithManyIndistinguisableReplicateSPSS,
        fit.measures = TRUE, standardized=TRUE)
```

```
## lavaan (0.6-1) converged normally after  22 iterations
##
##   Number of observations                           208
##
##   Estimator                                         ML      Robust
##   Model Fit Test Statistic                      90.264      79.481
##   Degrees of freedom                                19          19
##   P-value (Chi-square)                           0.000       0.000
##   Scaling correction factor                                  1.136
##     for the Satorra-Bentler correction
##
## Model test baseline model:
##
##   Minimum Function Test Statistic              145.879     125.301
##   Degrees of freedom                                15          15
##   P-value                                        0.000       0.000
##
## User model versus baseline model:
##
##   Comparative Fit Index (CFI)                    0.455       0.452
##   Tucker-Lewis Index (TLI)                       0.570       0.567
##
##   Robust Comparative Fit Index (CFI)                         0.465
##   Robust Tucker-Lewis Index (TLI)                            0.578
##
## Loglikelihood and Information Criteria:
##
##   Loglikelihood user model (H0)              -1411.209   -1411.209
##   Loglikelihood unrestricted model (H1)      -1365.859   -1365.859
##
##   Number of free parameters                          8           8
##   Akaike (AIC)                                2838.419    2838.419
##   Bayesian (BIC)                              2865.119    2865.119
##   Sample-size adjusted Bayesian (BIC)         2839.771    2839.771
##
## Root Mean Square Error of Approximation:
##
##   RMSEA                                          0.135       0.124
##   90 Percent Confidence Interval          0.107  0.163       0.098  0.151
##   P-value RMSEA <= 0.05                          0.000       0.000
##
##   Robust RMSEA                                               0.132
##   90 Percent Confidence Interval                             0.103  0.163
##
## Standardized Root Mean Square Residual:
```

```
##
##   SRMR                                           0.166       0.166
##
## Parameter Estimates:
##
##   Information                                 Expected
##   Information saturated (h1) model          Structured
##   Standard Errors                           Robust.sem
##
## Latent Variables:
##                    Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##   mActor =~
##     mfanx             1.000                               0.456    0.573
##     mcanx             1.000                               0.456    0.573
##     myanx             1.000                               0.456    0.573
##   mPartner =~
##     fmanx             1.000                               0.245    0.313
##     cmanx             1.000                               0.245    0.313
##     ymanx             1.000                               0.245    0.313
##
## Covariances:
##                    Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##   mActor ~~
##     mPartner  (gr)    0.079    0.022    3.572    0.000    0.703    0.703
##  .mfanx ~~
##    .fmanx     (dr)    0.116    0.025    4.640    0.000    0.116    0.239
##  .mcanx ~~
##    .cmanx     (dr)    0.116    0.025    4.640    0.000    0.116    0.239
##  .myanx ~~
##    .ymanx     (dr)    0.116    0.025    4.640    0.000    0.116    0.239
##
## Intercepts:
##                    Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##    .mcanx     (ia)    1.808    0.041   44.102    0.000    1.808    2.272
##    .mfanx     (ia)    1.808    0.041   44.102    0.000    1.808    2.272
##    .myanx     (ia)    1.808    0.041   44.102    0.000    1.808    2.272
##    .cmanx     (ip)    1.698    0.034   49.587    0.000    1.698    2.171
##    .fmanx     (ip)    1.698    0.034   49.587    0.000    1.698    2.171
##    .ymanx     (ip)    1.698    0.034   49.587    0.000    1.698    2.171
##     mActor            0.000                               0.000    0.000
##     mPartner          0.000                               0.000    0.000
##
## Variances:
##                    Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##    .mcanx      (a)    0.425    0.044    9.705    0.000    0.425    0.672
##    .mfanx      (a)    0.425    0.044    9.705    0.000    0.425    0.672
##    .myanx      (a)    0.425    0.044    9.705    0.000    0.425    0.672
##    .cmanx      (p)    0.552    0.047   11.759    0.000    0.552    0.902
##    .fmanx      (p)    0.552    0.047   11.759    0.000    0.552    0.902
##    .ymanx      (p)    0.552    0.047   11.759    0.000    0.552    0.902
##     mActor  (Actr)    0.208    0.037    5.567    0.000    1.000    1.000
##     mPartnr (Prtn)    0.060    0.029    2.043    0.041    1.000    1.000
```

References

Kenny, D. A., Kashy, D. A., & Cook, W. L. (2006). *Dyadic data analysis.* New York: Guilford Press.

Appendix

The difference between *lme* function in *nlme* and SPSS

lme parameterizes this model differently than SPSS. In *lme*, the error variance Var(1) is:

```
sigma(mlm)^2
```

```
## [1] 0.423157
```

and the error variance Var(2) is given by this error variance times a multiplicative factor:

```
sigma(mlm)^2 * coef(mlm$modelStruct$varStruct, unconstrained=FALSE)^2
```

```
##         2
## 0.5492319
```

The CI for Var(1) is given by the function *intervals* (see defintion of of cis above):

```
cis$sigma^2
```

```
##     lower      est.     upper
## 0.3693293 0.4231570 0.4848298
## attr(,"label")
## [1] "Within-group standard error:"
```

But, *intervals* does not produce the CI for Var(2). Instead, it provides the CI for the multiplicative factor:

```
cis$varStruct^2
```

```
##      lower     est.    upper
## 2 1.076345 1.297939 1.565154
## attr(,"label")
## [1] "Variance function:"
```

To estimate this CI, there are at least 3 possibilities:

1. Take the CI for the multiplicative factor and multiply by Var(1). This is what is shown in the table "SRM estimates for the reciprocal one-with-many design for indistinguishable partners" above:

```
cis$varStruct[c(1,3)]^2 * sigma(mlm)^2
```

```
## [1] 0.4554629 0.6623059
```

This ignores the uncertainty in sigma(mlm)^2.

2. Take the CI for Var(1) and multiply by the multiplicative factor:

```
cis$sigma[c(1,3)]^2 * coef(mlm$modelStruct$varStruct, unconstrained=FALSE)^2
```

```
##     lower     upper
## 0.4793669 0.6292795
```

This ignores the uncertainty in coef(mlm*modelStruct*varStruct, unconstrained=FALSE)^2.

3. Take the CI for the multiplicative factor and multiply its bounds by those of the CI for Var(1):

```
cis$varStruct[c(1,3)]^2 * cis$sigma[c(1,3)]^2
```

```
##     lower     upper
## 0.3975257 0.7588333
```

But it isn't correct to 'combine' two CIs in this manner.

None of these are exactly the same as the CI for Var(2) in SPSS (approach 1 comes close, but this might not be true in general). SPSS parameterizes the model in terms of Var(1) and Var(2), so getting the CI for Var(1) and Var(2) is easy. But *lme* uses a different parameterization, so one can directly get the CI for the multiplicative factor, but not the CI for Var(2). The advantage of this parameterization is that it allows testing whether Var(1) and Var(2) are significantly different from each other: Because the CI for the multiplicative factor (1.077194 to 1.56392) excludes 1, the difference is significant (at $\alpha = .05$, two-tailed). Of course one can always do a likelihood ratio test to examine if H0: Var(1) = Var(2), but *lme* parameterization provides it automatically. So, both parameterizations are useful (one directly gives the CI for Var(2), the other for Var(2) / Var(1)).

If you want the same parameterization as SPSS, then you could use *metafor*. While it is slow (couple of hours!!!), it does give the same CI for Var(2) (and the other parameters) as SPSS.

metafor **solution**

```
if (!require('metafor')) devtools::install_github("wviechtb/metafor")
```

```
## Loading required package: metafor

## Loading required package: Matrix

## Loading 'metafor' package (version 2.1-0). For an overview
## and introduction to the package please type: help(metafor).
```

```
library(metafor)

Chapter10_df$dyadid.in.focalid <-
                          interaction(Chapter10_df$focalid, Chapter10_df$dyadid)

res <- rma.mv(outcome ~ 0 + focalcode + partcode,
              V = 0,
              random = list(~ 0 + focalcode + partcode | focalid,
                            ~ 0 + focalcode + partcode | dyadid.in.focalid),
              struct = "GEN",
              data = Chapter10_df,
              sparse = TRUE)
res
```

```
##
## Multivariate Meta-Analysis Model (k = 1248; method: REML)
##
## Variance Components:
##
## outer factor: focalid              (nlvls = 208)
## inner term:   ~0 + focalcode [...] (nlvls = 2)
##
##             estim    sqrt  fixed  rho:    fclc  prtc
## focalcode  0.2084  0.4565     no             -    no
## partcode   0.0609  0.2468     no        0.6988     -
```

```
##
## outer factor: dyadid.in.focalid    (nlvls = 624)
## inner term:   ~0 + focalcode [...] (nlvls = 2)
##
##              estim    sqrt  fixed  phi:    fclc  prtc
## focalcode  0.4232  0.6505     no              -    no
## partcode   0.5492  0.7411     no         0.2390     -
##
## Test of Moderators (coefficients 1:2):
## QM(df = 2) = 3157.3380, p-val < .0001
##
## Model Results:
##
##             estimate      se     zval    pval   ci.lb   ci.ub
## focalcode    1.8077  0.0410  44.1019  <.0001  1.7274  1.8880  ***
## partcode     1.6983  0.0342  49.5865  <.0001  1.6311  1.7654  ***
##
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The results above yield the same parameter estimates as SPSS and *nlme*. The *confint* functions below produce the same CI as SPSS, but can run up to 3 hours to converge!

```r
confint(res)
```

```
##
##         estimate  ci.lb  ci.ub
## tau^2.1   0.2084 0.1453 0.2874
## tau.1     0.4565 0.3812 0.5361
##
##         estimate  ci.lb  ci.ub
## tau^2.2   0.0609 0.0128 0.1192
## tau.2     0.2468 0.1133 0.3453
##
##     estimate  ci.lb  ci.ub
## rho   0.6988 0.3660 1.0000
##
##           estimate  ci.lb  ci.ub
## gamma^2.1   0.4232 0.3705 0.4863
## gamma.1     0.6505 0.6087 0.6973
##
##           estimate  ci.lb  ci.ub
## gamma^2.2   0.5492 0.4809 0.6312
## gamma.2     0.7411 0.6935 0.7945
##
##     estimate  ci.lb  ci.ub
## phi   0.2390 0.1465 0.3274
```

```r
# ci.rho <- confint(res, rho=1, verbose=TRUE)
# ci.rho
# ci.phi <- confint(res, phi=1, verbose=TRUE)
# ci.phi
```