Dyadic Incongruence in R using OLS, HLM, and SEM

Joel S Steele

Data preparation

In order to demonstrate the outcome of a dyadic incongruence approach we begin by simulating some data. It is always best to check your work when you know the answer ahead of time. Here we simulate 50 observations from two groups, g1 and g2, that will later serve as our dyadic partners.

```r
# set the same seed to get the same results. A good step in any simulation.
set.seed(42) # why not
# create a data frame with columns 'grp' and 'out' for group and outcome respectively.
ex.df = data.frame(
    'grp'=factor(
        c(rep('g1',25),
          rep('g2',25))),
    'out'=c(
        rnorm(25,mean=10,sd=15),
        rnorm(25,mean=25,sd=7)))
# quick peek
head(ex.df)
```

```
  grp       out
1  g1 30.564377
2  g1  1.529527
3  g1 15.446926
4  g1 19.492939
5  g1 16.064025
6  g1  8.408132
```

Packages needed

For the following demonstration we will be using a number of packages in R. Namely, **psych**, **nlme**, **lavaan**, and **emmeans**. Let's load them into our session. If these are not installed already run the following,

```r
install.packages(c("psych","lavaan","emmeans"))
```

Note that **nlme** should be installed as part of R by default.

```r
# load the libraries
library(psych)
library(nlme)
library(lavaan)
```

```
This is lavaan 0.6-5

lavaan is BETA software! Please report any bugs.


Attaching package: 'lavaan'
```

```
The following object is masked from 'package:psych':

    cor2cov
```

```r
library(emmeans)
```

```r
# entire sample
describe(ex.df$out,skew=F)
```

Quick description to check our simulation

```
   vars  n mean   sd    min  max range   se
X1    1 50   18 15.4 -29.85 44.3 74.15 2.18
```

```r
# per group statistics
describeBy(ex.df$out,group=ex.df$grp,skew=F)
```

```

 Descriptive statistics by group
group: g1
   vars  n  mean   sd    min  max range   se
X1    1 25 12.81 19.6 -29.85 44.3 74.15 3.92
------------------------------------------------------------
group: g2
   vars  n  mean   sd min   max range   se
X1    1 25 23.19 6.63 8.1 35.11 27.01 1.33
```

```r
# balanced data here
mean(ex.df$out) # grand mean on outcome
```

```
[1] 18.00044
```

```r
with(ex.df, aggregate(out~grp,FUN=mean)) # group means on outcome
```

```
  grp      out
1  g1 12.81304
2  g2 23.18784
```

```r
# group means differences from grand mean on outcome
with(ex.df, aggregate(out~grp,FUN=mean))[,"out"]-mean(ex.df$out)
```

```
[1] -5.1874  5.1874
```

```r
# difference in group means on outcome
diff(with(ex.df, aggregate(out~grp,FUN=mean))[,"out"])
```

```
[1] 10.3748
```

OLS

Let's start by examining the estimates under Treatment coding (default). This equates to difference between group1 (intercept) and group 2 means.

```r
summary(lm(out~grp,ex.df))
```

```

Call:
lm(formula = out ~ grp, data = ex.df)
```

```
Residuals:
    Min      1Q  Median      3Q     Max 
-42.660  -5.247  -0.958   6.715  31.487 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)   12.813      2.925   4.380 6.42e-05 ***
grpg2         10.375      4.137   2.508   0.0156 *  
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 14.63 on 48 degrees of freedom
Multiple R-squared:  0.1158,    Adjusted R-squared:  0.09741 
F-statistic: 6.288 on 1 and 48 DF,  p-value: 0.01559
```

Next we can see the difference using effect coding. Here the (intercept) is now grand mean (see above) and the group coefficient is group difference from grand mean

```
summary(lm(out~grp,ex.df,contrasts=list('grp'=contr.sum))->lm1)
```

```

Call:
lm(formula = out ~ grp, data = ex.df, contrasts = list(grp = contr.sum))

Residuals:
    Min      1Q  Median      3Q     Max 
-42.660  -5.247  -0.958   6.715  31.487 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)   18.000      2.069   8.702 1.96e-11 ***
grp1          -5.187      2.069  -2.508   0.0156 *  
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 14.63 on 48 degrees of freedom
Multiple R-squared:  0.1158,    Adjusted R-squared:  0.09741 
F-statistic: 6.288 on 1 and 48 DF,  p-value: 0.01559
```

In order to understand a bit more let's have a look at the coding

```
contr.sum(2) # scaled as 1,-1
```

```
  [,1]
1    1
2   -1
```

Below we creat a new contrast, rescaled for dyadic incongruence

```
newcontr = contr.sum(2)*.5
newcontr # see the difference
```

```
  [,1]
1  0.5
2 -0.5
```

This new contrast will scale our effects by a factor of 2!

```r
summary(lm(out~grp,ex.df,contrasts=list('grp'=newcontr))->lm2)
```

```

Call:
lm(formula = out ~ grp, data = ex.df, contrasts = list(grp = newcontr))

Residuals:
    Min      1Q  Median      3Q     Max 
-42.660  -5.247  -0.958   6.715  31.487 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)   18.000      2.069   8.702 1.96e-11 ***
grp1         -10.375      4.137  -2.508   0.0156 *  
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 14.63 on 48 degrees of freedom
Multiple R-squared:  0.1158,    Adjusted R-squared:  0.09741 
F-statistic: 6.288 on 1 and 48 DF,  p-value: 0.01559
```

Mixed-Effects or HLM

Next we will extend this idea to include dyadic nesting through HLM. In order to do so we need to add in a dyadic indicator.

```r
# copy the data to a new data frame
ml.df = ex.df
# add in dyad id for nesting
ml.df$did = rep(1:25,times=2)
# quick peek
head(ml.df[order(ml.df$did),])
```

```
   grp       out did
1   g1 30.564377   1
26  g2 21.986716   1
2   g1  1.529527   2
27  g2 23.199114   2
3   g1 15.446926   3
28  g2 12.657858   3
```

```r
# dyad indicator set to 1,-1 via contr.sum
lme1 = lme(out~grp,data=ml.df,
           random=~grp|did,
           contrasts=list('grp'=contr.sum))
summary(lme1)
```

```
Linear mixed-effects model fit by REML
 Data: ml.df 
       AIC      BIC    logLik
  389.4855 400.7127 -188.7427

Random effects:
 Formula: ~grp | did
 Structure: General positive-definite, Log-Cholesky parametrization
            StdDev    Corr  
```

```
(Intercept)  9.634065 (Intr)
grp1        10.163366 0.868
Residual     4.224940

Fixed effects: out ~ grp
               Value Std.Error DF   t-value p-value
(Intercept) 18.00044  2.017328 24  8.922915   0.000
grp1        -5.18740  2.118670 24 -2.448423   0.022
 Correlation:
     (Intr)
grp1 0.796

Standardized Within-Group Residuals:
        Min          Q1         Med          Q3         Max
-1.47878710 -0.22838070 -0.02783819  0.28270531  1.13958788

Number of Observations: 50
Number of Groups: 25
```

```
# dyadic incongruence contrast
lme2 = lme(out~grp,data=ml.df,
    random=~grp|did,
    contrasts=list('grp'=newcontr))
summary(lme2)
```

```
Linear mixed-effects model fit by REML
 Data: ml.df
       AIC      BIC    logLik
  388.0992 399.3264 -188.0496

Random effects:
 Formula: ~grp | did
 Structure: General positive-definite, Log-Cholesky parametrization
            StdDev    Corr
(Intercept)  9.634065 (Intr)
grp1        20.326733 0.868
Residual     4.224940

Fixed effects: out ~ grp
                Value Std.Error DF   t-value p-value
(Intercept)  18.00044  2.017328 24  8.922915   0.000
grp1        -10.37480  4.237340 24 -2.448423   0.022
 Correlation:
     (Intr)
grp1 0.796

Standardized Within-Group Residuals:
        Min          Q1         Med          Q3         Max
-1.47878710 -0.22838070 -0.02783819  0.28270531  1.13958788

Number of Observations: 50
Number of Groups: 25
```

Examining estimated marginal means

```r
emmeans(lm1,list(pairwise~grp))# same P(>|z|)
```

OLS models

```
$`emmeans of grp`
 grp emmean   SE df lower.CL upper.CL
 g1    12.8 2.93 48     6.93     18.7
 g2    23.2 2.93 48    17.31     29.1

Confidence level used: 0.95

$`pairwise differences of grp`
 1        estimate   SE df t.ratio p.value
 g1 - g2     -10.4 4.14 48 -2.508  0.0156
```

```r
emmeans(lm2,list(pairwise~grp))# same
```

```
$`emmeans of grp`
 grp emmean   SE df lower.CL upper.CL
 g1    12.8 2.93 48     6.93     18.7
 g2    23.2 2.93 48    17.31     29.1

Confidence level used: 0.95

$`pairwise differences of grp`
 1        estimate   SE df t.ratio p.value
 g1 - g2     -10.4 4.14 48 -2.508  0.0156
```

```r
emmeans(lme1,list(pairwise~grp))# same
```

LME models

```
$`emmeans of grp`
 grp emmean   SE df lower.CL upper.CL
 g1    12.8 3.92 24     4.72     20.9
 g2    23.2 1.33 24    20.45     25.9

Degrees-of-freedom method: containment
Confidence level used: 0.95

$`pairwise differences of grp`
 1        estimate   SE df t.ratio p.value
 g1 - g2     -10.4 4.24 24 -2.448  0.0220

Degrees-of-freedom method: containment
```

```r
emmeans(lme2,list(pairwise~grp))# yawn...same
```

```
$`emmeans of grp`
 grp emmean   SE df lower.CL upper.CL
 g1    12.8 3.92 24     4.72     20.9
 g2    23.2 1.33 24    20.45     25.9

Degrees-of-freedom method: containment
```

```
Confidence level used: 0.95

$`pairwise differences of grp`
 1        estimate   SE df t.ratio p.value
 g1 - g2     -10.4 4.24 24 -2.448  0.0220

Degrees-of-freedom method: containment
```

Structural Equation Modeling

To begin we include some hard coded contrast coefficients in order to estimate the incongruence between partners.

```r
# group coding
lv.df = ex.df
# hard code the contrast!
lv.df$gi = ifelse(lv.df$grp=='g1',.5,-.5)
# quick peek
head(lv.df)
```

```
  grp       out  gi
1  g1 30.564377 0.5
2  g1  1.529527 0.5
3  g1 15.446926 0.5
4  g1 19.492939 0.5
5  g1 16.064025 0.5
6  g1  8.408132 0.5
```

```r
eff_code_mod = '
out ~ a*gi
out ~ 1
out~~out
incong := .5*a #effect code estimate
'
fit = lavaan(eff_code_mod,data=lv.df,effect.coding=c('intercepts'))
summary(fit)
```

```
lavaan 0.6-5 ended normally after 19 iterations

  Estimator                                         ML
  Optimization method                           NLMINB
  Number of free parameters                          3

  Number of observations                            50

Model Test User Model:

  Test statistic                                 0.000
  Degrees of freedom                                 0

Parameter Estimates:

  Information                                 Expected
  Information saturated (h1) model          Structured
  Standard errors                             Standard
```

```
Regressions:
                   Estimate  Std.Err  z-value  P(>|z|)
  out ~
    gi         (a)  -10.375    4.054   -2.559    0.010

Intercepts:
                   Estimate  Std.Err  z-value  P(>|z|)
   .out              18.000    2.027    8.881    0.000

Variances:
                   Estimate  Std.Err  z-value  P(>|z|)
   .out             205.401   41.080    5.000    0.000

Defined Parameters:
                   Estimate  Std.Err  z-value  P(>|z|)
    incong           -5.187    2.027   -2.559    0.010
```

```
# just to prove it we will hard code an effect coding contrast
lv.df = ex.df
lv.df$gi2 = ifelse(lv.df$grp=='g1',1,-1)
# quick peek
head(lv.df)
```

```
  grp       out gi2
1  g1 30.564377   1
2  g1  1.529527   1
3  g1 15.446926   1
4  g1 19.492939   1
5  g1 16.064025   1
6  g1  8.408132   1
```

```
eff_code_mod2 = '
out ~ a*gi2
out ~ 1
out~~out
'
fit2 = lavaan(eff_code_mod2,data=lv.df,effect.coding=c('intercepts'))
summary(fit2)
```

```
lavaan 0.6-5 ended normally after 19 iterations

  Estimator                                         ML
  Optimization method                           NLMINB
  Number of free parameters                          3

  Number of observations                            50

Model Test User Model:

  Test statistic                                 0.000
  Degrees of freedom                                 0

Parameter Estimates:

  Information                                 Expected
  Information saturated (h1) model          Structured
```

```
  Standard errors                             Standard

Regressions:
                   Estimate  Std.Err  z-value  P(>|z|)
  out ~
    gi2        (a)   -5.187    2.027   -2.559    0.010

Intercepts:
                   Estimate  Std.Err  z-value  P(>|z|)
   .out              18.000    2.027    8.881    0.000

Variances:
                   Estimate  Std.Err  z-value  P(>|z|)
   .out             205.401   41.080    5.000    0.000
```

Below we code the loadings from our dyadic outcomes onto a latent variable that represents the incongruence between the partners. For the following examples in SEM, we will need wide data.

```r
g1dat = ml.df[ml.df$grp=='g1',] # all group one data
g2dat = ml.df[ml.df$grp=='g2',] # all group two data
lv.wide= merge(g1dat,g2dat,by=c('did'),all=T) # stick them together based on dyad id.
# quick peek
head(lv.wide)
```

```
  did grp.x     out.x grp.y    out.y
1   1    g1 30.564377    g2 21.98672
2   2    g1  1.529527    g2 23.19911
3   3    g1 15.446926    g2 12.65786
4   4    g1 19.492939    g2 28.22068
5   5    g1 16.064025    g2 20.52004
6   6    g1  8.408132    g2 28.18815
```

Our first model estimates the dyadic incongruence by creating a latent variable with constrained loadings from the outcomes or each partner. The loadings represent the incongruence coding seen above in the OLS and LME models.

```r
dlcm_mod.a = '
d =~ -.5*out.x + .5*out.y
d ~ 1
out.x ~ 0
out.y ~ 0
out.x ~~ 1*out.x
out.y ~~ 1*out.y
'
fit.dlcm.a = sem(dlcm_mod.a,data=lv.wide)
summary(fit.dlcm.a)
```

```
lavaan 0.6-5 ended normally after 36 iterations

  Estimator                                         ML
  Optimization method                           NLMINB
  Number of free parameters                          2

  Number of observations                            25

Model Test User Model:
```

```
  Test statistic                             20952.519
  Degrees of freedom                                 3
  P-value (Chi-square)                           0.000

Parameter Estimates:

  Information                                 Expected
  Information saturated (h1) model          Structured
  Standard errors                             Standard

Latent Variables:
                   Estimate  Std.Err  z-value  P(>|z|)
  d =~
    out.x            -0.500
    out.y             0.500

Intercepts:
                   Estimate  Std.Err  z-value  P(>|z|)
    d                10.375    4.152    2.499    0.012
   .out.x             0.000
   .out.y             0.000

Variances:
                   Estimate  Std.Err  z-value  P(>|z|)
   .out.x             1.000
   .out.y             1.000
    d               428.921  121.883    3.519    0.000
```

Next we add in the dyadic average to this model above. This way we estimate both the level of the dyadic outcome as well as the incongruence between partners.

```r
dlcm_mod.b = '
l =~ 1*out.x + 1*out.y
d =~ -.5*out.x + .5*out.y
d ~ 1
l ~ 1
d ~~ l
out.x ~ 0
out.y ~ 0
out.x ~~ 1*out.x
out.y ~~ 1*out.y
'
fit.dlcm.b = sem(dlcm_mod.b,data=lv.wide)
summary(fit.dlcm.b)
```

```
lavaan 0.6-5 ended normally after 90 iterations

  Estimator                                         ML
  Optimization method                           NLMINB
  Number of free parameters                          5

  Number of observations                            25

Model Test User Model:
```

```
  Test statistic                                 0.000
  Degrees of freedom                                 0

Parameter Estimates:

  Information                                 Expected
  Information saturated (h1) model          Structured
  Standard errors                             Standard

Latent Variables:
                   Estimate  Std.Err  z-value  P(>|z|)
  l =~
    out.x             1.000
    out.y             1.000
  d =~
    out.x            -0.500
    out.y             0.500

Covariances:
                   Estimate  Std.Err  z-value  P(>|z|)
  l ~~
    d              -163.222   52.433   -3.113    0.002

Intercepts:
                   Estimate  Std.Err  z-value  P(>|z|)
    d                10.375    4.152    2.499    0.012
    l                18.000    1.977    9.107    0.000
   .out.x             0.000
   .out.y             0.000

Variances:
                   Estimate  Std.Err  z-value  P(>|z|)
   .out.x             1.000
   .out.y             1.000
    l                97.171   27.625    3.517    0.000
    d               428.921  121.883    3.519    0.000
```