Appendix S1 Implementing the multigroup piecewise path model in R

Supplement to the publication: “A multigroup extension to piecewise path-analysis” published in Ecosphere

Bob Douma & Bill Shipley

24 Februari 2021

Contents

1 Preface

In this document we explain how multi-group testing can be combined with path-models that are expressed as Direct Acyclic Graphs (DAGs). First, we illustrate two ways to perform a multigroup analysis in R: by using the d-sep test and the χ^2 method. Next we show how different constraints can be applied to the multi-group models.

1.1 load libraries

The following libraries are needed to run the syntax below.

```r
library(ggm) # for DAGs
library(nlme) # mixed effect models
library(lavaan) # classic structural equation modelling
library(glmmTMB) # generalized linear mixed effect models
library(boot) # for the inverse logit
library(betareg) # beta regression
```

2 Illustration of the multigroup d-sep method

We assume the reader that he/she is familiar with the d-sep test. We refer to the main text or to Shipley (2009) - Confimatory path analysis in a gereralized multilevel context (Ecology) for more details on the d-sep test. Take the following steps:

1. Specify a causal hypothesis for each group in the form of a DAG.
2. Define the basis-set
3. For each group, calculate the null probability associated with the predicted independence claims. The predicted independence is calculated based on a model that assumes path coefficients equal/non-equal among groups.
4. Combine these null probability in a Fisher C statistic per group. Then add up the Fischer's C values and the k probabilities and test with Chi-square statistic.
5. From the models that are accepted, use AIC statistic to choose among models

2.1 Generate dataset

```r
# set random number generator to specific seed
set.seed(11)
```

```r
# number of observations
n=200
# make groups
group = as.factor(rep(c(1,2),n/2))
# generate variables
X1 = runif(n,0,100)
X2 = runif(n,0,100)
X3 = c(0.5,-0.5)[group]*X1  -0.2*X2 + rnorm(n,0,1)
X4 = c(2,4)[group]*X3 + rnorm(n,0,1)
X5 = c(-2)*X3 + rnorm(n,0,1)
dat = data.frame(X1,X2,X3,X4,X5,group)
```

2.2 Step 1: specify causal hypothesis

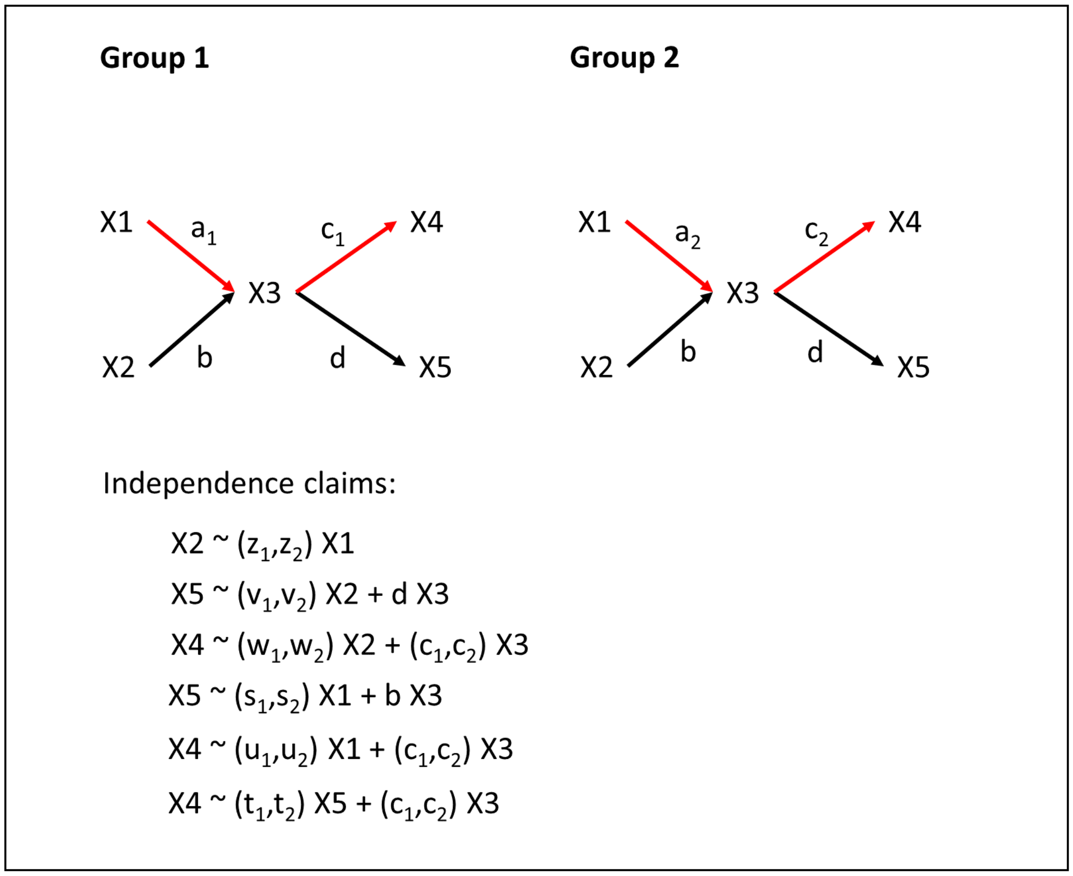
```r
  # specify DAG
  Z = DAG(X4~ X3,
          X5 ~ X3,
          X3 ~ X1+X2)
```

2.3 Step 2: construct a series of independence claims

Construct the basis-set.

```r
  bas = basiSet(Z)
```

The multigroup model that was used to generate the data is presented in the figure below, including their independence claims. Note that some paths have different path coefficients per group which is reflected in the independence claims.


Group 1
Group 2
X1
a_1
c_1
X4
X3
X2
b
d
X5
X1
a_2
c_2
X4
X3
X2
b
d
X5
Independence claims:
X2 ~ (z_1,z_2) X1
X5 ~ (v_1,v_2) X2 + d X3
X4 ~ (w_1,w_2) X2 + (c_1,c_2) X3
X5 ~ (s_1,s_2) X1 + b X3
X4 ~ (u_1,u_2) X1 + (c_1,c_2) X3
X4 ~ (t_1,t_2) X5 + (c_1,c_2) X3

To test the multigroup model in R, we need a slightly different syntax for the structural equations (aka regressions) for difference between groups compared to the conventional approach.

The default in R to for difference between groups is by setting one group as baseline, and the other group as difference with respect to the baseline. See example below:

```
lm.11 = lm(X2 ~ X1*group,data=dat)
summary(lm.11)
```

```
##
## Call:
## lm(formula = X2 ~ X1 * group, data = dat)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -53.329 -22.953   1.301  22.555  51.865
##
## Coefficients:
```

```
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept) 52.62966    5.41713   9.715   <2e-16 ***
## X1           0.02019    0.10147   0.199    0.842    
## group2      -5.45291    7.87240  -0.693    0.489    
## X1:group2    0.02319    0.14819   0.157    0.876    
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 28.13 on 196 degrees of freedom
## Multiple R-squared:  0.00715,    Adjusted R-squared:  -0.008047 
## F-statistic: 0.4705 on 3 and 196 DF,  p-value: 0.7032
```

The conclusion of this regression is that for group 1 the slope between X1 and X2 is not different from zero and that the slopes of group 2 and group 1 are not significantly different. However, we are not interested in the difference in the slope of group 2 with respect to group 1. We are interested whether the slope between X1 and X2 for group 2 is different from zero.

Whether the slope between X1 and X2 of group 2 is different from zero can be calculated by summing the baseline slope (X1) and the interaction between X1:group2, and calcuating its standard error.

```
# estimate of slope of group 2
m = sum(coef(lm.11)[c(2,4)])
m  # this adds up the slope for group 1 and for group 2, i.e.
```

```
## [1] 0.04338179
```

Next, extract the variance-covariance matrix of the model through `vcov()` to compute the associated significance:

```
vc <- vcov(lm.11)
vc
```

```
##              (Intercept)          X1      group2   X1:group2
## (Intercept)   29.3453139 -0.46976765 -29.3453139  0.46976765
## X1            -0.4697676  0.01029692   0.4697676 -0.01029692
## group2       -29.3453139  0.46976765  61.9746252 -1.00667085
## X1:group2      0.4697676 -0.01029692  -1.0066708  0.02196010
```

Calculate the variance for the newly calculated variable through: $Var(X+Y) = Var(X) + Var(Y) + 2 * Cov(X,Y)$. The variance of two random variables is the sum of their variances and 2* their covariance. Sqrt to get to standard error:

```
se = (sqrt(sum(c(vc[2,2], vc[4,4], 2*vc[2,4]))))
se
```

```
## [1] 0.1079962
```

```
tval  = m/se # t-value
# significance of slope beteen X1 and X2 for group 2
2 * pt(abs(tval),196,lower.tail = FALSE)
```

```
## [1] 0.6883445
```

Thus the relationship between X1 and X2 for group 2 is not significantly different from zero.

This result can be directly obtained by using the syntax `group/x1` instead of `group*x1`:

```
lm.22 = lm(X2 ~group/X1)
summary(lm.22)
```

```
##
## Call:
## lm(formula = X2 ~ group/X1)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -53.329 -22.953   1.301  22.555  51.865
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 52.62966    5.41713   9.715   <2e-16 ***
## group2      -5.45291    7.87240  -0.693    0.489
## group1:X1    0.02019    0.10147   0.199    0.842
## group2:X1    0.04338    0.10800   0.402    0.688
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 28.13 on 196 degrees of freedom
## Multiple R-squared:  0.00715,    Adjusted R-squared:  -0.008047
## F-statistic: 0.4705 on 3 and 196 DF,  p-value: 0.7032
```

This syntax is most convenient to use for the multigroup d-sep, because the last two terms represent the probability associated with the independence claim for both groups. For example, to test the fourth independence claim:

```r
# fourth independence claim
lm.4 = lm(X4 ~ group/X3+group/X2-1,data=dat)
summary(lm.4)
```

```
##
## Call:
## lm(formula = X4 ~ group/X3 + group/X2 - 1, data = dat)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.96267 -0.71921  0.00064  0.73491  2.62699
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## group1      0.003586   0.285829   0.013    0.990
## group2      0.146058   0.277070   0.527    0.599
## group1:X3   2.012252   0.007664 262.544   <2e-16 ***
## group2:X3   4.004379   0.007991 501.086   <2e-16 ***
## group1:X2 -0.001534   0.004087  -0.375    0.708
## group2:X2   0.001556   0.004156   0.374    0.709
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.059 on 194 degrees of freedom
## Multiple R-squared:  0.9999, Adjusted R-squared:  0.9999
## F-statistic: 3.286e+05 on 6 and 194 DF,  p-value: < 2.2e-16
```

The path of X2 on X4 is not significant for group 1 and just significant for group 2. Note that the `-1`makes the intercept relative to zero and not to the baseline.

If you want to constrain the path from `X3` to `X4` one tests:

```r
# test for no difference in path coefficients between groups for effect X3 on X4
lm.4a = lm(X4 ~ X3+group/X2-1,data=dat)
summary(lm.4a)
```

```
##
## Call:
## lm(formula = X4 ~ X3 + group/X2 - 1, data = dat)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -28.8575 -11.2336   0.5081  11.3039  25.4390
##
```

```
## Coefficients:
##            Estimate Std. Error t value Pr(>|t|)
## X3          2.96673    0.07148  41.504  < 2e-16 ***
## group1    -21.01153    3.37108  -6.233 2.77e-09 ***
## group2    -22.77368    3.17949  -7.163 1.58e-11 ***
## group1:X2   0.17689    0.05124   3.453 0.000681 ***
## group2:X2  -0.22502    0.05118  -4.397 1.81e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 13.68 on 195 degrees of freedom
## Multiple R-squared:  0.9835, Adjusted R-squared:  0.9831
## F-statistic:  2323 on 5 and 195 DF,  p-value: < 2.2e-16
```

Now the `X2` and `X4` are not independent anymore, and thus this independence claim is rejected. This was not the case when allowing the path coefficient of X3 to X4 to be group specific.

Using this syntax all independence claims can be tested, and compared to a set of independence claims in which we assume no difference between groups.

2.4 Step 3: test independence claims

For convenience we have assumed that the residual variance is equal among groups. However, to accomdate the fact that different groups may have different variance, one should use `gls` from the package `nlme` and specify a variance per group. This can be done through setting the `weights` argument to `varIdent(form = ~1|group)`. The p-values of the coefficients can be extracted with `summary(gls.1)$tTable`

```r
# first independence claim
gls.1 = gls(X2~group/X1-1,weights=varIdent(form=~1|group),data=dat)
summary(gls.1)$tTable
```

```
##                 Value Std.Error   t-value      p-value
## group1    52.62965615 5.3788883 9.7844858 1.161136e-18
## group2    47.17674295 5.7522518 8.2014391 3.077400e-14
## group1:X1  0.02018951 0.1007574 0.2003775 8.413931e-01
## group2:X1  0.04338179 0.1087533 0.3989009 6.904005e-01
```

```r
# second independence claim
gls.2 = gls(X5~group/X3+group/X2-1,weights=varIdent(form=~1|group),data=dat)
summary(gls.2)$tTable
```

```
##                  Value   Std.Error      t-value       p-value
## group1    -0.143435012 0.261417907   -0.5486809  5.838554e-01
## group2      0.064818727 0.250065790    0.2592067  7.957506e-01
## group1:X3 -1.992235267 0.007009855 -284.2049226 3.708299e-256
## group2:X3 -1.990796675 0.007212536 -276.0189634 1.060497e-253
## group1:X2  0.002006460 0.003737975    0.5367771  5.920366e-01
## group2:X2  0.006763608 0.003750948    1.8031729  7.291280e-02
```

```
# no difference in path coefficients between groups for effect of x3 on x5
gls.2a = gls(X5~X3+group/X2-1,weights=varIdent(form=~1|group),data=dat)
summary(gls.2a)$tTable
```

```
##                  Value   Std.Error      t-value       p-value
## X3        -1.991536366 0.005014212 -397.1783728 1.515325e-285
## group1    -0.158822999 0.237675582   -0.6682344  5.047742e-01
## group2     0.048480345 0.221872583    0.2185053  8.272639e-01
## group1:X2  0.002137110 0.003615812    0.5910456  5.551743e-01
## group2:X2  0.006602094 0.003567667    1.8505357  6.574910e-02
```

```
# third independence claim
gls.3 = gls(X5 ~ group/X3+group/X1-1,weights=varIdent(form=~1|group),data=dat)
summary(gls.3)$tTable
```

```
##                  Value   Std.Error      t-value       p-value
## group1    -0.169108097 0.261194203   -0.6474420  5.181113e-01
## group2     0.058806294 0.245295401    0.2397366  8.107875e-01
## group1:X3 -2.004056474 0.017457989 -114.7930889 2.741972e-180
## group2:X3 -2.029209739 0.016757282 -121.0942048 9.941834e-185
## group1:X1  0.006028989 0.009287589    0.6491447  5.170126e-01
## group2:X1 -0.020035592 0.009389920   -2.1337341  3.412045e-02
```

```
# no diffs between groups in effect of x3 on x5
gls.3a = gls(X5 ~ X3+group/X1-1,weights=varIdent(form=~1|group),data=dat)
summary(gls.3a)$tTable
```

```
##                Value   Std.Error      t-value       p-value
## X3        -2.01714823 0.012091805 -166.8194419 2.526202e-212
## group1    -0.30637641 0.225398762   -1.3592639  1.756328e-01
## group2     0.16799256 0.221709073    0.7577162  4.495356e-01
## group1:X1  0.01248328 0.006908236    1.8070137  7.230158e-02
## group2:X1 -0.01380677 0.007230540   -1.9095081  5.766451e-02
```

```
# fourth independence claim
gls.4 = gls(X4 ~ group/X3+group/X2-1,weights=varIdent(form=~1|group),data=dat)
summary(gls.4)$tTable
```

```
##                   Value   Std.Error     t-value       p-value
## group1      0.003585724 0.270573687   0.0132523  9.894401e-01
## group2      0.146057604 0.291107626   0.5017306  6.164260e-01
## group1:X3   2.012252143 0.007255365 277.3467899 4.189738e-254
## group2:X3   4.004379255 0.008396287 476.9226149 1.046212e-299
## group1:X2  -0.001534496 0.003868892  -0.3966240  6.920807e-01
## group2:X2   0.001555945 0.004366569   0.3563313  7.219797e-01
```

```
# no difference in path coefficients between groups for effect X3 on X4
gls.4a = gls(X4 ~ X3+group/X2-1,weights=varIdent(form=~1|group),data=dat)
summary(gls.4a)$tTable
```

```
##                    Value   Std.Error    t-value       p-value
## X3            2.014843947 0.007255359 277.704245 2.690046e-255
## group1       -0.053479062 0.270688249  -0.197567  8.435897e-01
## group2      -43.799061171 5.377950372  -8.144192  4.473214e-14
## group1:X2    -0.001049993 0.003871099  -0.271239  7.864942e-01
## group2:X2    -0.432865138 0.094955173  -4.558626  9.076781e-06
```

```
# fifth independence claim
gls.5 = gls(X4 ~ group/X3+group/X1-1,weights=varIdent(form=~1|group),data=dat)
summary(gls.5)$tTable
```

```
##                   Value   Std.Error      t-value       p-value
## group1      0.005115538 0.270519959   0.01891002  9.849323e-01
## group2      0.162209422 0.287519060   0.56416928  5.732906e-01
## group1:X3   2.020016311 0.018081314 111.71844563 4.915554e-178
## group2:X3   3.999168305 0.019641779 203.60520209 3.745652e-228
## group1:X1  -0.003878074 0.009619196  -0.40315989  6.872746e-01
## group2:X1  -0.002404559 0.011006244  -0.21847229  8.272908e-01
```

```
# no diffs between groups on effect of X3 on X4
gls.5a = gls(X4 ~ X3+group/X1-1,weights=varIdent(form=~1|group),data=dat)
summary(gls.5a)$tTable
```

```
##                 Value  Std.Error      t-value       p-value
## X3         3.97984759 0.01964195 202.61979971 1.076828e-228
## group1    20.55414035 2.13165478   9.64234009  3.080892e-18
## group2    -0.01269044 0.28839545  -0.04400361  9.649466e-01
## group1:X1 -0.97008291 0.04090608 -23.71488524  2.315469e-59
## group2:X1 -0.01238218 0.01101451  -1.12417015  2.623228e-01
```

```
# sixth independence claim
gls.6 = gls(X5 ~ group/X3+group/X4-1,weights=varIdent(form=~1|group),data=dat)
summary(gls.6)$tTable
```

```
##                  Value  Std.Error    t-value      p-value
## group1    -0.02637182 0.12503732 -0.2109116 8.331776e-01
## group2     0.23974166 0.23737156  1.0099847 3.137606e-01
## group1:X3 -1.85544292 0.19724264 -9.4069060 1.493795e-17
## group2:X3 -1.52000716 0.35147288 -4.3246783 2.441289e-05
## group1:X4 -0.06860086 0.09791755 -0.7005982 4.843932e-01
## group2:X4 -0.11896961 0.08778422 -1.3552505 1.769136e-01
```

```
# no diffs between groups on effect of X3 on X5
gls.6a = gls(X5 ~ X3+group/X4-1,weights=varIdent(form=~1|group),data=dat)
summary(gls.6a)$tTable
```

```
##                  Value  Std.Error     t-value      p-value
## X3        -1.77500401 0.17190946 -10.3252257 3.240566e-20
## group1    -0.03105942 0.12486348  -0.2487470 8.038183e-01
## group2     0.22524976 0.23644828   0.9526386 3.419526e-01
## group1:X4 -0.10851122 0.08535623  -1.2712747 2.051452e-01
## group2:X4 -0.05529255 0.04296022  -1.2870639 1.995977e-01
```

2.5 Step 4 combine null probabilities

```
  # make function to extract the p-values of the
extract.p = function(x,mod,group){
   spec = group-1
    if (mod =="gls"){
      pvals <- summary(x)$tTable
      n = nrow(pvals)
      ps <- pvals[(n-spec):n,4]
    } else if (mod=="lm") {
      pvals = summary(x)$coefficients
      n = nrow(pvals)
      ps = pvals[(n-spec):n,4]
    }
    return(ps)
  }

 extract.p(lm.11,mod="lm",group=2)
```

```
##    group2 X1:group2
## 0.4893413 0.8757966
```

```
##### different model for different groups; no constraints on
#path coefficients between groups  collect p-vales for C statistic;

  C1=-2*sum(log(c(extract.p(gls.1,mod="gls",group=2),
                  extract.p(gls.2,mod="gls",group=2),
                  extract.p(gls.3,mod="gls",group=2),
                  extract.p(gls.4,mod="gls",group=2),
                  extract.p(gls.5,mod="gls",group=2),
                  extract.p(gls.6,mod="gls",group=2))))

  C2=-2*sum(log(c(extract.p(gls.1,mod="gls",group=2)[2],
                  extract.p(gls.2,mod="gls",group=2)[2],
                  extract.p(gls.3,mod="gls",group=2)[2],
                  extract.p(gls.4,mod="gls",group=2)[2],
                  extract.p(gls.5,mod="gls",group=2)[2],
                  extract.p(gls.6,mod="gls",group=2)[2])))

  # accept the model with differences between groups
  p.diff <- 1-pchisq(C1+C2,2*2*6)
  p.diff
```

```
## [1] 0.02083224
```

All path coefficients equal among groups

```
C1=-2*sum(log(c(extract.p(gls.1,mod="gls",group=2)[1],
                extract.p(gls.2a,mod="gls",group=2)[1],
                extract.p(gls.3a,mod="gls",group=2)[1],
                extract.p(gls.4a,mod="gls",group=2)[1],
                extract.p(gls.5a,mod="gls",group=2)[1],
                extract.p(gls.6a,mod="gls",group=2)[1])))

C2=-2*sum(log(c(extract.p(gls.1,mod="gls",group=2)[2],
                extract.p(gls.2a,mod="gls",group=2)[2],
                extract.p(gls.3a,mod="gls",group=2)[2],
                extract.p(gls.4a,mod="gls",group=2)[2],
                extract.p(gls.5a,mod="gls",group=2)[2],
                extract.p(gls.6a,mod="gls",group=2)[2])))

# model with one path coefficients for all paths not accepted
p.constrained <- 1-pchisq(C1+C2,2*12)
p.constrained
```

Table 1: Comparing the p-values with the path models with varying constraints

path coeff different	path coeff equal	path coeff partially equal
0.0208322	0	0.3482143

```
## [1] 0
```

Only the path coefficients of `X2` to `X3` and `X3` to `X5` constrained to be equal among groups

```
    C1=-2*sum(log(c(extract.p(gls.1, mod="gls",group=2)[1],
                    extract.p(gls.2a, mod="gls",group=2)[1],
                    extract.p(gls.3a, mod="gls",group=2)[1],
                    extract.p(gls.4, mod="gls",group=2)[1],
                    extract.p(gls.5, mod="gls",group=2)[1],
                    extract.p(gls.6, mod="gls",group=2)[1])))

  C2=-2*sum(log(c(extract.p(gls.1, mod="gls",group=2)[2],
                  extract.p(gls.2a, mod="gls",group=2)[2],
                  extract.p(gls.3a, mod="gls",group=2)[2],
                  extract.p(gls.4, mod="gls",group=2)[2],
                  extract.p(gls.5, mod="gls",group=2)[2],
                  extract.p(gls.6, mod="gls",group=2)[2])))

# model accepted
p.partial <- 1-pchisq(C1+C2,2*12)
p.partial
```

```
## [1] 0.3482143
```

2.6 Step 5 compare models

```
  tab <- data.frame(t(c(p.diff,p.constrained,p.partial)))
  colnames(tab) <- c("path coeff different","path coeff equal","path coeff partially equal")
  knitr::kable(tab, booktabs = TRUE,
      caption = 'Comparing the p-values with the path models with varying constraints'
  )
```

Conclusion; model with parameters free for each group is accepted, like the model with constraints on path `b` and `d`. Note that path `a` never occurs in one of the independence claims. Thus the model with constraints is purely rejected based on path `c`.

3 Multigroup modelling with the Likellihood Ratio Test

Shipley & Douma (2020) Generalized AIC and chi-squared statistics for path models consistent with directed acyclic graphs (Ecology) presented an alternative way to test path models expressed as DAGs. We refer to the main body of this paper, and the above mentioned paper for a detailed justification of this methodology.

3.1 Step 1: Fit the regressions

In the first step the individual regressions models are fitted. Each variable is regressed against its parents (when available).

```
# group specific path coefficients
gls.X1.diff <- gls(X1 ~ group,weights=varIdent(form=~1|group),method="ML",data=dat)
gls.X2.diff <- gls(X2 ~ group,weights=varIdent(form=~1|group),method="ML",data=dat)
gls.X4.diff <- gls(X4 ~ group*X3 + group,
                   weights=varIdent(form=~1|group),method="ML",data=dat)
gls.X5.diff <- gls(X5 ~ group*X3 + group,
                   weights=varIdent(form=~1|group),method="ML",data=dat)
gls.X3.diff <- gls(X3 ~ group*X1+group*X2+group,
                   weights=varIdent(form=~1|group),
                   method="ML",data=dat)

logLik.diff<- logLik(gls.X1.diff) + logLik(gls.X2.diff) +
  logLik(gls.X3.diff) + logLik(gls.X4.diff) + logLik(gls.X5.diff)

AIC.diff<- AIC(gls.X1.diff) + AIC(gls.X2.diff) +
  AIC(gls.X3.diff) + AIC(gls.X4.diff) + AIC(gls.X5.diff)


# no difference between groups
gls.X1.cons <- gls(X1 ~ group,weights=varIdent(form=~1|group),
                   method="ML",data=dat)
gls.X2.cons <- gls(X2 ~ group,weights=varIdent(form=~1|group),
                   method="ML",data=dat)
gls.X4.cons <- gls(X4 ~ X3+group,weights=varIdent(form=~1|group),
                   method="ML",data=dat)
gls.X5.cons <- gls(X5 ~ X3+group,weights=varIdent(form=~1|group),
                   method="ML",data=dat)
gls.X3.cons <- gls(X3 ~ X1+X2+group,weights=varIdent(form=~1|group),
                   method="ML",data=dat)

logLik.cons<- logLik(gls.X1.cons) + logLik(gls.X2.cons) +
 logLik(gls.X3.cons) + logLik(gls.X4.cons) + logLik(gls.X5.cons)
```

```r
AIC.cons<- AIC(gls.X1.cons) + AIC(gls.X2.cons) +
  AIC(gls.X3.cons) + AIC(gls.X4.cons) + AIC(gls.X5.cons)

# path a and c group specific
gls.X1.partial <- gls(X1 ~ group,weights=varIdent(form=~1|group),
                      method="ML",data=dat)
gls.X2.partial <- gls(X2 ~ group,weights=varIdent(form=~1|group),
                      method="ML",data=dat)
gls.X4.partial <- gls(X4 ~ group*X3,weights=varIdent(form=~1|group),
                      method="ML",data=dat)
gls.X5.partial <- gls(X5 ~ X3+group,weights=varIdent(form=~1|group),
                      method="ML",data=dat)
gls.X3.partial <- gls(X3 ~ group*X1+X2+group,weights=varIdent(form=~1|group),
                      method="ML",data=dat)

logLik.partial<- logLik(gls.X1.partial) + logLik(gls.X2.partial) +
  logLik(gls.X3.partial) + logLik(gls.X4.partial) + logLik(gls.X5.partial)

AIC.partial<- AIC(gls.X1.partial) + AIC(gls.X2.partial) +
  AIC(gls.X3.partial) + AIC(gls.X4.partial) + AIC(gls.X5.partial)
```

3.2 Step 2: Fit the saturated model

In the second step a saturated path-model is fitted. This is a path model in which all variables are linked to other variables.

```r
gls.X1.sat <- gls(X1 ~ group,weights=varIdent(form=~1|group),
                  method="ML", data=dat)
gls.X2.sat <- gls(X2 ~ group*X1,weights=varIdent(form=~1|group),
                  method="ML", data=dat)
gls.X3.sat <- gls(X3 ~ group*X1 + group*X2,
                  weights=varIdent(form=~1|group),method="ML",data=dat)
gls.X4.sat <- gls(X4 ~ group*X3 + group*X2 + group*X1 + group*X5,
                  weights=varIdent(form=~1|group),method="ML",data=dat)
gls.X5.sat <- gls(X5 ~ group*X3 + group*X2 + group*X1,
                  weights=varIdent(form=~1|group),method="ML", data=dat)

logLik.sat<- logLik(gls.X1.sat) + logLik(gls.X2.sat) + logLik(gls.X3.sat) +
  logLik(gls.X4.sat) + logLik(gls.X5.sat)
```

3.3 Calculate the Chi-squre value and the associated p-value

In the third step the logLikelihood of the hypothesized model is compared to the logLikelihood of the saturated model and with a Likelihood Ratio Test (LRT) one can test if any deviations in the logLikelihood are due to chance. This is expected when the hypothesized model was underlying the data.

```
Chi.diff <- -2*(logLik.diff - logLik.sat)
Chi.cons <- -2*(logLik.cons - logLik.sat)
Chi.partial <- -2*(logLik.partial - logLik.sat)

p.chi.diff <- 1-pchisq(Chi.diff,df=12)
p.chi.cons <- 1-pchisq(Chi.cons,df=16)
p.chi.partial <- 1-pchisq(Chi.partial,df=14)

p.chi.diff ; p.chi.cons ;p.chi.partial
```

```
## 'log Lik.' 0.5006126 (df=4)

## 'log Lik.' 0 (df=4)

## 'log Lik.' 0.6343851 (df=4)
```

4 Comparison of the results with lavaan

Fit path model and check coefficients to compare with lavaan

```
groupsem = "
  X4 ~ X3
  X5 ~ X3
  X3 ~ X1+X2

  #
  X1~~ 0*X2
  X5 ~~0*X1
  X5 ~~0*X2
  X4 ~~0*X1
  X4 ~~0*X2
  X4 ~~0*X5
"
# all path coefficients different between groups
sem.1 = sem(groupsem,data=dat,group="group",fixed.x=F)
# AIC value of the model
logLik(sem.1) ; logLik.diff
```

```
## 'log Lik.' -2734.083 (df=28)
## 'log Lik.' -2734.083 (df=4)
```

```
AIC(sem.1); AIC.diff
```

```
## [1] 5524.166
```

```
## [1] 5524.166
```

```
#parTable(sem.1)
# Chi square value
lavInspect(sem.1,"fit")["chisq"]; Chi.diff
```

```
##    chisq
## 11.33305
```

```
## 'log Lik.' 11.33305 (df=4)
```

```
lavInspect(sem.1,"fit")["pvalue"]; p.chi.diff
```

```
##    pvalue
## 0.5006126
```

```
## 'log Lik.' 0.5006126 (df=4)
```

```
# all path coefficients equal between groups
groupsem.cons = "
  X4 ~ X3
  X5 ~ X3
  X3 ~ X1 + X2

  #
  X1 ~~0*X2
  X5 ~~0*X1
  X5 ~~0*X2
  X4 ~~0*X1
  X4 ~~0*X2
  X4 ~~0*X5
"
sem.2 = sem(groupsem.cons,data=dat,group="group",group.equal="regressions",fixed.x=F)
#summary(sem.2)
logLik(sem.2) ; logLik.cons
```

```
## 'log Lik.' -3397.549 (df=24)

## 'log Lik.' -3397.549 (df=4)
```

```
AIC(sem.2); AIC.cons
```

```
## [1] 6843.098

## [1] 6843.098
```

```
# Chi square value
lavInspect(sem.2,"fit")["chisq"]; Chi.cons
```

```
##    chisq
## 1338.265

## 'log Lik.' 1338.265 (df=4)
```

```
lavInspect(sem.2,"fit")["pvalue"]; p.chi.cons
```

```
## pvalue
##      0

## 'log Lik.' 0 (df=4)
```

```
# partial constraints
groupsem_coef = "
  X4 ~ c(d1,d2)*X3
  X5 ~ c(e,e)*X3
  X3 ~ c(a1,a2)*X1+c(b,b)*X2

  X1~~0.*X2
  X5 ~~0*X1
  X5 ~~0*X2
  X4 ~~0*X1
  X4 ~~0*X2
  X4 ~~0*X5
"
sem.3 = sem(groupsem_coef,data=dat,group="group",fixed.x=F)
logLik(sem.3) ; logLik.partial
```

```
## 'log Lik.' -2734.241 (df=26)

## 'log Lik.' -2734.241 (df=4)
```

```
AIC(sem.3); AIC.partial
```

```
## [1] 5520.483
```

```
## [1] 5520.483
```

```
# Chi square value
lavInspect(sem.3,"fit")["chisq"]; Chi.partial
```

```
##    chisq
## 11.65003
```

```
## 'log Lik.' 11.65003 (df=4)
```

```
lavInspect(sem.3,"fit")["pvalue"]; p.chi.partial
```

```
##    pvalue
## 0.6343851
```

```
## 'log Lik.' 0.6343851 (df=4)
```

```
#parTable(sem.3)

# saturated model
groupsem_sat = "
  X4 ~ c(d1,d2)*X3 + c(e1,e2)*X5 + c(f1,f2)*X1 + c(g1,g2)*X2
  X5 ~ c(m1,m2)*X3  + c(h1,h2)*X1 + c(i1,i2)*X2
  X3 ~ c(a1,a2)*X1+c(b1,b2)*X2
  X2 ~ c(z1,z2)*X1
"
sem.sat = sem(groupsem_sat,data=dat,group="group",fixed.x=F)
logLik(sem.sat) ; logLik.sat
```

```
## 'log Lik.' -2728.416 (df=40)
```

```
## 'log Lik.' -2728.416 (df=4)
```

Comparing the local estimation method, X^2_{LML}, with the global estimation as done in classical SEM X^2_{GML}. Also qualitatively, the results are similar to the d-sep test. The ΔX^2 is calculated by subtracting the logLikelihood of the hyothesized model from the logLikelihood of the saturated model.

Table 2: Comparing the statistics of the local Chi-square method X^2_{LML} and the global statistic X^2_{GML} as fitted by 'lavaan'. The ΔX^2 is calculated by subtracting the logL of the hypothesized model from the logL of the saturated model

	$\log L_{local}$	$\log L_{global}$	ΔX^2	Δdf	p_{ML}
no constraints	-2734.083	-2734.083	11.3330467272792	12	0.500612648540946
all constraints	-3397.549	-3397.549	1338.26511235971	16	0
partial constraints	-2734.241	-2734.241	11.6500305700598	14	0.634385135507488
saturated model	-2728.416	-2728.416	0	NA	NA

5 Model specification for various constraints (path coefficient, intercept, variance etc.)

For the d-sep test it is most convenient to write the group model syntax as: group/X, but for the χ^2_{LML} one can also use group*X, as you collect the likelihood of the fitted model and not the p-values in the first place. Always check whether all intercepts and interaction terms are present in the summary statement as you intended to be.

5.1 Fixing path coefficients across groups

Below we fix the path from `X3` to `X4` to be eqaul across groups. Thus we change the `group/X3` into `X3`.

```
gls.path.fixed = gls(X4 ~ X3+group/X2-1,weights=varIdent(form=~1|group),
                     data=dat)
summary(gls.path.fixed)$tTable
```

```
##                    Value   Std.Error    t-value       p-value
## X3           2.014843947 0.007255359 277.704245 2.690046e-255
## group1      -0.053479062 0.270688249  -0.197567  8.435897e-01
## group2     -43.799061171 5.377950372  -8.144192  4.473214e-14
## group1:X2  -0.001049993 0.003871099  -0.271239  7.864942e-01
## group2:X2  -0.432865138 0.094955173  -4.558626  9.076781e-06
```

5.2 Fixing path coefficients to a predefined value

To fix the path coefficient to a preset value, one use `offset`. One can fix different path coefficients for different groups.

```
gls.path.set = gls(X4 ~ offset(3*X3)+group/X2-1,weights=varIdent(form=~1|group),
                   data=dat)
summary(gls.path.set)$tTable
```

```
##                  Value  Std.Error   t-value      p-value
## group1     44.3081408  6.1221794  7.237315 1.012199e-11
## group2    -88.3032005 10.8124053 -8.166841 3.811995e-14
## group1:X2  -0.3776979  0.1015772 -3.718333 2.617108e-04
## group2:X2  -0.8728124  0.1909662 -4.570506 8.600711e-06
```

```
gls.path.set1 = gls(X4 ~ offset(c(2.0,4.0)[group]*X3)+group/X2-1,
                   weights=varIdent(form=~1|group),data=dat)
summary(gls.path.set1)$tTable
```

```
##                  Value  Std.Error   t-value      p-value
## group1     44.3081408  6.1221794  7.237315 1.012199e-11
## group2    -88.3032005 10.8124053 -8.166841 3.811995e-14
## group1:X2  -0.3776979  0.1015772 -3.718333 2.617108e-04
## group2:X2  -0.8728124  0.1909662 -4.570506 8.600711e-06
```

5.3 Fixing the intercept

One can also constrain the intercept to be equal among groups. First the unconstrained syntax is shown and below the constrained. In the first output you see the terms `group1` and `group2` which represent the group-specific intercepts. In the second `lm.4.int` you see `Intercept` reported which is the intercept across groups. To have a model without intercept add -1‘.

```
    gls.intercept.free = gls(X4 ~ group/X3+group/X2-1,weights=varIdent(form=~1|group),
                             data=dat)
    summary(gls.intercept.free)$tTable
```

```
##                  Value   Std.Error     t-value       p-value
## group1     0.003585724 0.270573687   0.0132523  9.894401e-01
## group2     0.146057604 0.291107626   0.5017306  6.164260e-01
## group1:X3  2.012252143 0.007255365 277.3467899 4.189738e-254
## group2:X3  4.004379255 0.008396287 476.9226149 1.046212e-299
## group1:X2 -0.001534496 0.003868892  -0.3966240  6.920807e-01
## group2:X2  0.001555945 0.004366569   0.3563313  7.219797e-01
```

```
    gls.intercept.fixed = gls(X4 ~ group/X3+group/X2-group,
                              weights=varIdent(form=~1|group),data=dat)
    summary(gls.intercept.fixed)$tTable
```

```
##                    Value   Std.Error     t-value       p-value
## (Intercept)  0.069643152 0.197745519   0.3521857  7.250793e-01
## group1:X3    2.011206376 0.006629486 303.3728907 9.122299e-263
## group2:X3    4.002975142 0.007408572 540.3167257 1.380925e-311
## group1:X2   -0.002364355 0.003093765  -0.7642323  4.456526e-01
## group2:X2    0.001945755 0.004218818   0.4612087  6.451629e-01
```

5.4 Fixing path coefficients for >2 groups

```
set.seed(101)
n=200
# make four groups
group4 = as.factor(rep(c(1,2,3,4),n/4))
X1_4 = runif(n,0,100)
X2_4 = runif(n,0,100)
X3_4 = c(0.5,0.5,-0.5,-0.5)[group4]*X1  -0.2*X2 + rnorm(n,0,1)
X4_4 = c(1,2,3,4)[group4]*X3 + rnorm(n,0,1)
X5_4 = c(-2,-2,-2,-2.)*X3 + rnorm(n,0,1)
dat4 = data.frame(X1_4,X2_4,X3_4,X4_4,X5_4,group4)

gls.2_4 = gls(X5_4~group4/X3_4+group4/X2_4,data=dat4,weights=varIdent(form=~1|group))
summary(gls.2_4)
```

```
## Generalized least squares fit by REML
##   Model: X5_4 ~ group4/X3_4 + group4/X2_4
##   Data: dat4
##        AIC      BIC    logLik
##   1653.082 1698.393 -812.5412
##
## Variance function:
##  Structure: Different standard deviations per stratum
##  Formula: ~1 | group
##  Parameter estimates:
##        1        2
## 1.000000 1.007767
##
## Coefficients:
##                 Value Std.Error    t-value p-value
## (Intercept)  -0.47506  4.612741  -0.102989  0.9181
## group42      60.62836  6.680473   9.075460  0.0000
## group43      19.07284  7.672564   2.485849  0.0138
## group44      -0.64533  7.254058  -0.088961  0.9292
## group41:X3_4 -1.98938  0.124024 -16.040299  0.0000
## group42:X3_4  1.51281  0.136426  11.088912  0.0000
## group43:X3_4  1.27487  0.136900   9.312414  0.0000
## group44:X3_4 -2.02143  0.146692 -13.780103  0.0000
## group41:X2_4  0.01642  0.075012   0.218883  0.8270
## group42:X2_4 -0.28456  0.075974  -3.745479  0.0002
## group43:X2_4  0.01316  0.066105   0.199128  0.8424
## group44:X2_4 -0.00500  0.071193  -0.070202  0.9441
##
##  Correlation:
##              (Intr) grop42 grop43 grop44 g41:X3 g42:X3 g43:X3 g44:X3 g41:X2
## group42      -0.690
## group43      -0.601  0.415
## group44      -0.636  0.439  0.382
## group41:X3_4 -0.405  0.279  0.243  0.257
## group42:X3_4  0.000 -0.246  0.000  0.000  0.000
## group43:X3_4  0.000  0.000  0.640  0.000  0.000  0.000
## group44:X3_4  0.000  0.000  0.000  0.604  0.000  0.000  0.000
## group41:X2_4 -0.836  0.577  0.502  0.531  0.080  0.000  0.000  0.000
## group42:X2_4  0.000 -0.592  0.000  0.000  0.000 -0.067  0.000  0.000  0.000
## group43:X2_4  0.000  0.000 -0.470  0.000  0.000  0.000 -0.111  0.000  0.000
## group44:X2_4  0.000  0.000  0.000 -0.302  0.000  0.000  0.000  0.143  0.000
##              g42:X2 g43:X2
## group42
## group43
## group44
## group41:X3_4
## group42:X3_4
## group43:X3_4
## group44:X3_4
## group41:X2_4
## group42:X2_4
## group43:X2_4  0.000
## group44:X2_4  0.000  0.000
##
## Standardized residuals:
##         Min          Q1         Med          Q3         Max
## -3.62438833 -0.26465108  0.04317097  0.33263604  2.46102612
##
## Residual standard error: 14.10092
## Degrees of freedom: 200 total; 188 residual
```

As you can see in the summary output, the last four coefficients and p-values represent the independence claims for each of the four groups.

5.5 Fixing the variance

```
gls.variance.free = gls(X4 ~ group/X3+group/X2-1,
                        weights=varIdent(form=~1|group),data=dat)
```

```r
summary(gls.variance.free)
```

```
## Generalized least squares fit by REML
##   Model: X4 ~ group/X3 + group/X2 - 1
##   Data: dat
##       AIC      BIC   logLik
##   638.962 665.1048 -311.481
##
## Variance function:
##  Structure: Different standard deviations per stratum
##  Formula: ~1 | group
##  Parameter estimates:
##        1        2
## 1.000000 1.109904
##
## Coefficients:
##                Value  Std.Error  t-value p-value
## group1      0.003586 0.27057369   0.0133  0.9894
## group2      0.146058 0.29110763   0.5017  0.6164
## group1:X3   2.012252 0.00725536 277.3468  0.0000
## group2:X3   4.004379 0.00839629 476.9226  0.0000
## group1:X2  -0.001534 0.00386889  -0.3966  0.6921
## group2:X2   0.001556 0.00436657   0.3563  0.7220
##
##  Correlation:
##           group1 group2 gr1:X3 gr2:X3 gr1:X2
## group2     0.000
## group1:X3 -0.590  0.000
## group2:X3  0.000  0.637  0.000
## group1:X2 -0.879  0.000  0.351  0.000
## group2:X2  0.000 -0.340  0.000  0.420  0.000
##
## Standardized residuals:
##           Min            Q1           Med            Q3           Max
## -2.6636260218 -0.6699152535  0.0005748431  0.7167050660  2.5362486360
##
## Residual standard error: 1.002131
## Degrees of freedom: 200 total; 194 residual
```

```r
gls.variance.fixed = gls(X4 ~ group/X3+group/X2-group,data=dat)
summary(gls.variance.fixed)
```

```
## Generalized least squares fit by REML
##   Model: X4 ~ group/X3 + group/X2 - group
##   Data: dat
##        AIC      BIC    logLik
##   636.1366 655.7746 -312.0683
##
## Coefficients:
##                Value  Std.Error  t-value p-value
## (Intercept)  0.077038 0.19849735   0.3881  0.6984
## group1:X3    2.011089 0.00692616 290.3612  0.0000
## group2:X3    4.003111 0.00714678 560.1280  0.0000
## group1:X2   -0.002457 0.00316408  -0.7766  0.4383
## group2:X2    0.001908 0.00402886   0.4736  0.6363
##
##  Correlation:
##           (Intr) gr1:X3 gr2:X3 gr1:X2
## group1:X3 -0.454
## group2:X3  0.510 -0.232
## group1:X2 -0.788  0.118 -0.402
## group2:X2 -0.251  0.114  0.603  0.198
##
## Standardized residuals:
##           Min            Q1           Med            Q3           Max
## -2.7508723060 -0.6722305217  0.0002347752  0.6845927003  2.4983906299
##
## Residual standard error: 1.056265
## Degrees of freedom: 200 total; 195 residual
```

```
#or
lm.variance.fixed = lm(X4 ~ group/X3+group/X2-group,data=dat)
summary(lm.variance.fixed)
```

```
##
## Call:
## lm(formula = X4 ~ group/X3 + group/X2 - group, data = dat)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.90565 -0.71005  0.00025  0.72311  2.63896
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.077038   0.198497   0.388    0.698
## group1:X3    2.011089   0.006926 290.361   <2e-16 ***
## group2:X3    4.003111   0.007147 560.128   <2e-16 ***
## group1:X2   -0.002457   0.003164  -0.777    0.438
## group2:X2    0.001908   0.004029   0.474    0.636
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.056 on 195 degrees of freedom
## Multiple R-squared:  0.9999, Adjusted R-squared:  0.9999
## F-statistic: 3.665e+05 on 4 and 195 DF,  p-value: < 2.2e-16
```

6 Applying constraints to non-normally distributed variables or nested designs

6.1 Generate data

```
set.seed(101)
n=200
# make groups
group = as.factor(rep(c(1,2),n/2))
# generate variables
X1 = runif(n,0,100)
X2 = runif(n,0,100)
X3 = c(0.5,-0.5)[group]*X1  -0.2*X2 + rnorm(n,0,1)
X4 = c(2,4)[group]*X3 + rnorm(n,0,1)
# Generate poisson distributed data
X5.p = rpois(n,exp(-0.02*X3))
# Generate beta distributed data
phi <- 200
prob <- inv.logit(c(-0.038)*X3)
X5.b = rbeta(n,shape1=prob*phi,shape2=(1-prob)*phi)
dat.non = data.frame(X1,X2,X3,X4,X5.p,X5.b,group)
```

6.2 Non-normal distributions

When modelling a non-normal distribution, one should consider to make the dispersion parameter group specific. This can for example be done in case of the beta distribution by making the ϕ group specific. For the negative binomial, the dispersion parameter k could be modelled group specific. However, there is no pre-canned solution in R to my knowledge. One could program this yourself using the approach advocated in Bolker's Ecological Models and Data in R. The application of other constraints (intercepts etc.) can be done in the same way as for the linear models with normal distribution.

```
# with glm
glm.p = glm(X5.p~group/X3+group/X4-1,family="poisson",data=dat.non)
summary(glm.p)
```

```
##
## Call:
## glm(formula = X5.p ~ group/X3 + group/X4 - 1, family = "poisson",
##     data = dat.non)
##
## Deviance Residuals:
##     Min        1Q    Median        3Q       Max
## -2.4269  -1.0247  -0.1141   0.5919   2.4015
##
## Coefficients:
##            Estimate Std. Error z value Pr(>|z|)
## group1    -0.302851   0.163203  -1.856   0.0635 .
## group2     0.388157   0.172709   2.247   0.0246 *
## group1:X3 -0.410457   0.269345  -1.524   0.1275
## group2:X3 -0.043657   0.285835  -0.153   0.8786
## group1:X4  0.196802   0.134912   1.459   0.1446
## group2:X4  0.007444   0.071519   0.104   0.9171
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 409.21  on 200  degrees of freedom
## Residual deviance: 207.25  on 194  degrees of freedom
## AIC: 569.88
##
## Number of Fisher Scoring iterations: 5
```

```
glm.beta = betareg(X5.b~group/X3+group/X4-1 | group,data=dat.non)
summary(glm.beta)
```

```
##
## Call:
## betareg(formula = X5.b ~ group/X3 + group/X4 - 1 | group, data = dat.non)
##
## Standardized weighted residuals 2:
##     Min      1Q  Median      3Q     Max
## -2.5013 -0.6493  0.0049  0.6794  2.7981
##
## Coefficients (mean model with logit link):
##            Estimate Std. Error z value Pr(>|z|)
## group1    -0.020318   0.020323  -1.000    0.317
## group2    -0.005487   0.034625  -0.158    0.874
## group1:X3 -0.007075   0.030824  -0.230    0.818
## group2:X3 -0.091786   0.070419  -1.303    0.192
## group1:X4 -0.015044   0.015461  -0.973    0.331
## group2:X4  0.013393   0.017620   0.760    0.447
##
## Phi coefficients (precision model with log link):
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)   5.3244     0.1411  37.729   <2e-16 ***
## group2        0.2422     0.1997   1.213    0.225
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Type of estimator: ML (maximum likelihood)
## Log-likelihood: 434.1 on 8 Df
## Pseudo R-squared: 0.9808
## Number of iterations: 15 (BFGS) + 1 (Fisher scoring)
```

6.3 Hierarchical designs

To both allow for a nested design and a group specific variance one should use the package `nlme`. If you want to make the dispersion parameter group specific, the package `glmmTMB` offers most opportunities to do so.

```
dat$mix = sample((rep(c(1:4),each=25)))

lm.w2 = lme(X4~group/X3+group/X1, random=~1|mix,
            weights=varIdent(form=~1|group),data=dat)

lm.w2.alt = lme(X4~group*X3+group*X1, random=~1|mix,
            weights=varIdent(form=~1|group),data=dat)



summary(lm.w2)
```

```
## Linear mixed-effects model fit by REML
##  Data: dat
##        AIC      BIC    logLik
##   636.9294 666.3401 -309.4647
##
## Random effects:
##  Formula: ~1 | mix
##         (Intercept) Residual
## StdDev:   0.1241354 1.115859
##
## Variance function:
##  Structure: Different standard deviations per stratum
##  Formula: ~1 | group
##  Parameter estimates:
##         2         1
## 1.0000000 0.8864091
## Fixed effects: X4 ~ group/X3 + group/X1
##                  Value Std.Error  DF   t-value p-value
## (Intercept)  0.019624 0.2753629 191   0.07127  0.9433
## group2       0.120976 0.3976944 191   0.30419  0.7613
## group1:X3    2.020741 0.0179005 191 112.88717  0.0000
## group2:X3    3.997708 0.0198167 191 201.73477  0.0000
## group1:X1   -0.004177 0.0095179 191  -0.43889  0.6612
## group2:X1   -0.003184 0.0110859 191  -0.28718  0.7743
##  Correlation:
##           (Intr) group2 gr1:X3 gr2:X3 gr1:X1
## group2    -0.665
## group1:X3  0.685 -0.481
## group2:X3 -0.008  0.460 -0.008
## group1:X1 -0.856  0.599 -0.927  0.007
## group2:X1 -0.006  0.232 -0.006  0.922  0.005
##
## Standardized Within-Group Residuals:
##         Min          Q1         Med          Q3         Max
## -2.67166258 -0.67009444 -0.04515523  0.68691292  2.46971392
##
## Number of Observations: 200
## Number of Groups: 4
```

```
dat.non$mix = as.factor(sample((rep(c(1:4),each=25))))

glmm.beta = glmmTMB(X5.b ~ group/X3-1  + (1|mix),dispformula = ~group,
                    family="beta_family" ,data=dat.non)
```

```
## Warning in (function (start, objective, gradient = NULL, hessian = NULL, : NA/
## NaN function evaluation
```

```
summary(glmm.beta)
```

```
##  Family: beta  ( logit )
## Formula:          X5.b ~ group/X3 - 1 + (1 | mix)
## Dispersion:            ~group
## Data: dat.non
##
##      AIC      BIC   logLik deviance df.resid
##   -852.7   -829.6    433.3   -866.7      193
##
```

```
## Random effects:
##
## Conditional model:
##  Groups Name        Variance  Std.Dev.
##  mix    (Intercept) 4.066e-11 6.377e-06
## Number of obs: 200, groups:  mix, 4
##
## Conditional model:
##            Estimate Std. Error z value Pr(>|z|)
## group1    -0.0215779  0.0203660   -1.06    0.289
## group2    -0.0094716  0.0345195   -0.27    0.784
## group1:X3 -0.0370476  0.0010273  -36.06   <2e-16 ***
## group2:X3 -0.0382743  0.0009804  -39.04   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Dispersion model:
##            Estimate Std. Error z value Pr(>|z|)
## (Intercept)   5.3149     0.1411   37.66   <2e-16 ***
## group2        0.2460     0.1997    1.23    0.218
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```