Group Search pt. 2

Strucchange

Easier to understand article:
http://journal.frontiersin.org/article/10.3389/fpsyg.2014.00438/abstract

More papers at Edgar Merkle's site:
http://semtools.r-forge.r-project.org/
estimation method depends on what type of covariate you have.

Need two new packages

```
library(psychotools)
library(OpenMx)
library(strucchange)
library(lavaan)
HS <- HolzingerSwineford1939
```

The One Factor Model from before

```
model1.lav <- '
F1 =~ x1 + x2 + x3 + x4 + x5 + x6 + x7 + x8 + x9
'
lav.fit <- cfa(model1.lav, HS,meanstructure=T)
#summary(lav.fit,fit=T)
```

Has problems with missing data, so can only use complete cases (BTW, no missing in HS, but for demo purposes)

```
comp <- complete.cases(HS)
HS.comp <- HS[comp,]
```

Test for continuous covariates:
1. "DM" 2. "CvM" 3. "maxLM"

Test for ordinal covariates: (note, takes much much longer – similar to semtree with ordinal)
1. "maxLMo" 2. "WDMo"

Lets run it – starting with no restrictions and searching

```
# can't run, not large enough sample size for 6 groups
#lav.fitGroup1 <- cfa(model1.lav, HS,meanstructure=T,group="ageyr",
#                    group.equal="loadings")


sctest(lav.fit, order.by = HS.comp$ageyr, parm = 1:8,
         vcov = "info", functional = "maxLMo",plot=T)


sctest(lav.fit, order.by = HS.comp$ageyr,
                            parm = 1:8, vcov = "info",
                            functional = "WDMo",plot=T)
```

```r
lav.fitGroup2 <- cfa(model1.lav, HS.comp,meanstructure=T,group="school",
                    group.equal=c("loadings","intercepts"))

#summary(lav.fitGroup2,fit=T)
anova(lav.fit,lav.fitGroup2)
```

```
## Chi Square Difference Test
##
##               Df    AIC    BIC  Chisq Chisq diff Df diff Pr(>Chisq)
## lav.fit       27 7756.4 7856.5 312.26
## lav.fitGroup2 70 7723.4 7864.2 422.00     109.73      43  9.536e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

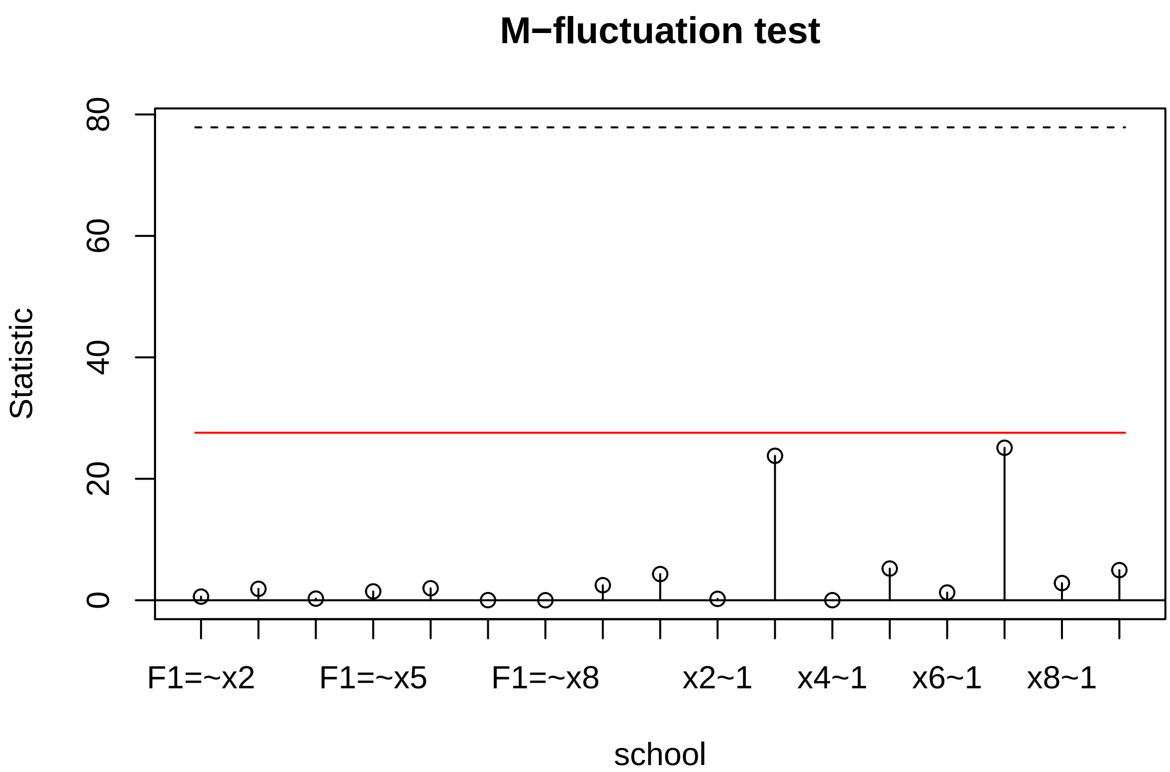
```r
# what are offending parameters
mod <- modindices(lav.fitGroup2)
mod[mod$op == "~1" & mod$mi > 10,]
```

```
##   lhs op rhs group     mi    epc sepc.lv sepc.all sepc.nox
## 1  x3 ~1         1 25.389  0.320   0.320    0.263    0.263
## 2  x7 ~1         1 24.437  0.306   0.306    0.275    0.275
## 3  x3 ~1         2 25.389 -0.320  -0.320   -0.311   -0.311
## 4  x7 ~1         2 24.437 -0.306  -0.306   -0.289   -0.289
```

2 Intercepts seem to be big problem

Do we get same answer with strucchange

```r
# get which parameters to fix:
# coef(lav.fitGroup2)
# intercepts 19:27
sctest(lav.fitGroup2, order.by = HS.comp$school,
                            parm = c(1:8,19:27), vcov = "info",
                            functional = "LMuo",plot=T)
```

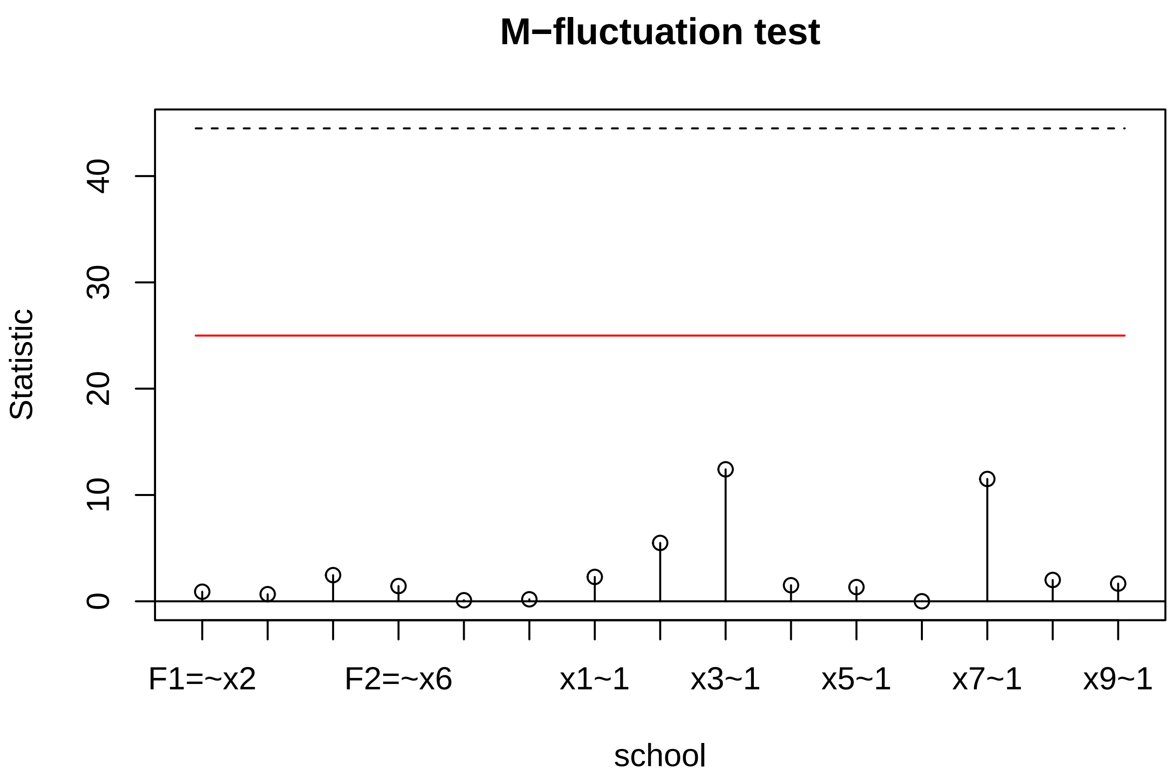



```
##
##  M-fluctuation test
##
## data:  lav.fitGroup2
## f(efp) = 77.877, p-value = 9.115e-10
```

```
# what if we change the model
model5.lav <- '
F1 =~ x1 + x2 + x3 # + x7 + x9
F2 =~ x4 + x5 + x6 # + x1
F3 =~ x7 + x8 + x9
'
lav.fitGroup5 <- cfa(model5.lav, HS,meanstructure=T,group="school",group.equal=c("loadings","intercepts
coef(lav.fitGroup5)
```

```
##     F1=~x2     F1=~x3     F2=~x5     F2=~x6     F3=~x8     F3=~x9     x1~~x1
##      0.576      0.798      1.120      0.932      1.130      1.009      0.555
##     x2~~x2     x3~~x3     x4~~x4     x5~~x5     x6~~x6     x7~~x7     x8~~x8
##      1.296      0.944      0.445      0.502      0.263      0.888      0.541
##     x9~~x9     F1~~F1     F2~~F2     F3~~F3     F1~~F2     F1~~F3     F2~~F3
##      0.654      0.796      0.879      0.322      0.410      0.178      0.180
##       x1~1       x2~1       x3~1       x4~1       x5~1       x6~1       x7~1
##      5.001      6.151      2.271      2.778      4.035      1.926      4.242
##       x8~1       x9~1 x1~~x1.g2 x2~~x2.g2 x3~~x3.g2 x4~~x4.g2 x5~~x5.g2
##      5.630      5.465      0.654      0.964      0.641      0.343      0.376
## x6~~x6.g2 x7~~x7.g2 x8~~x8.g2 x9~~x9.g2 F1~~F1.g2 F2~~F2.g2 F3~~F3.g2
##      0.437      0.625      0.434      0.522      0.708      0.870      0.505
## F1~~F2.g2 F1~~F3.g2 F2~~F3.g2   F1~1.g2   F2~1.g2   F3~1.g2
##      0.427      0.329      0.236     -0.148      0.576     -0.177
```

```r
sctest(lav.fitGroup5, order.by = HS.comp$school,
                            parm = c(1:6,22:30), vcov = "info",
                            functional = "LMuo",plot=T)
```




```
##
##  M-fluctuation test
##
## data:  lav.fitGroup5
## f(efp) = 44.4879, p-value = 9.22e-05
```

```r
#sctest(lav.fitGroup5, order.by = HS.comp$school,
#                             parm = c(1:9,25:33), vcov = "info",
#                             functional = "LMuo",plot=T)

lav.fitGroup55 <- cfa(model5.lav, HS,meanstructure=T,group="school",group.equal=c("loadings"))
lav.fitGroup555 <- cfa(model5.lav, HS,meanstructure=T,group="school")
lavTestLRT(lav.fitGroup555,lav.fitGroup55,lav.fitGroup5) # intercepts are prob
```

```
## Chi Square Difference Test
##
##                 Df    AIC    BIC  Chisq Chisq diff Df diff Pr(>Chisq)
## lav.fitGroup555 48 7484.4 7706.8 115.85
## lav.fitGroup55  54 7480.6 7680.8 124.04      8.192       6     0.2244
## lav.fitGroup5   60 7508.6 7686.6 164.10     40.059       6  4.435e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
mod2 <- modificationIndices(lav.fitGroup5)
mod2[mod2$op == "~1" & mod2$mi > 10,]
```

```
##   lhs op rhs group     mi    epc sepc.lv sepc.all sepc.nox
## 1  x3 ~1         1 17.717  0.248   0.248    0.206    0.206
## 2  x7 ~1         1 13.681  0.205   0.205    0.186    0.186
## 3  x3 ~1         2 17.717 -0.248  -0.248   -0.238   -0.238
## 4  x7 ~1         2 13.681 -0.205  -0.205   -0.193   -0.193
```

Rasch Trees

SEM Trees, but for binary variables (rasch model)
http://cran.r-project.org/web/packages/psychotree/vignettes/raschtree.pdf

You have to set up the dataset a certain way – its a little wonky

```r
library(psychotree)
```

```
## Loading required package: partykit
## Loading required package: grid
```

```r
library(colorspace)

HS.xs <- HS[,7:15]
for(i in 1:9){
HS.xs[,i] = ifelse(HS.xs[,i] < mean(HS.xs[,i]), 0, 1)
}
summary(HS.xs)
```

```
##        x1               x2               x3              x4
##  Min.   :0.0000   Min.   :0.0000   Min.   :0.000   Min.   :0.0000
##  1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.000   1st Qu.:0.0000
##  Median :1.0000   Median :0.0000   Median :0.000   Median :0.0000
##  Mean   :0.5282   Mean   :0.4286   Mean   :0.412   Mean   :0.4286
##  3rd Qu.:1.0000   3rd Qu.:1.0000   3rd Qu.:1.000   3rd Qu.:1.0000
##  Max.   :1.0000   Max.   :1.0000   Max.   :1.000   Max.   :1.0000
##        x5               x6               x7               x8
##  Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :0.000
##  1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.000
##  Median :1.0000   Median :0.0000   Median :0.0000   Median :0.000
##  Mean   :0.5216   Mean   :0.4252   Mean   :0.4385   Mean   :0.495
##  3rd Qu.:1.0000   3rd Qu.:1.0000   3rd Qu.:1.0000   3rd Qu.:1.000
##  Max.   :1.0000   Max.   :1.0000   Max.   :1.0000   Max.   :1.000
##        x9
##  Min.   :0.0000
##  1st Qu.:0.0000
##  Median :1.0000
##  Mean   :0.5083
##  3rd Qu.:1.0000
##  Max.   :1.0000
```

```r
mydata = data.frame(HS[,1:5])
mydata$scale <- as.matrix(HS.xs)
```

Run it

```r
RT_out <- raschtree(scale ~ sex + ageyr + school, data = mydata)
RT_out
```

```
## Rasch tree
##
## Model formula:
## scale ~ sex + ageyr + school
##
## Fitted party:
## [1] root
## |   [2] school in Grant-White: n = 145
## |             scalex2       scalex3       scalex4       scalex5       scalex6
## |        1.044759e-01  1.129595e+00  1.392938e-01 -5.345298e-01 -8.612114e-06
## |             scalex7       scalex8       scalex9
## |        8.550781e-01  1.392938e-01  1.740704e-01
## |   [3] school in Pasteur: n = 156
## |          scalex2    scalex3    scalex4    scalex5    scalex6    scalex7
## |       0.84676450 0.12235959 0.81343399 0.52340945 0.98301944 0.09179353
## |          scalex8    scalex9
## |       0.18349732 0.03062488
##
## Number of inner nodes:    1
## Number of terminal nodes: 2
## Number of parameters per node: 8
## Objective function (negative log-likelihood): 1060.886
```

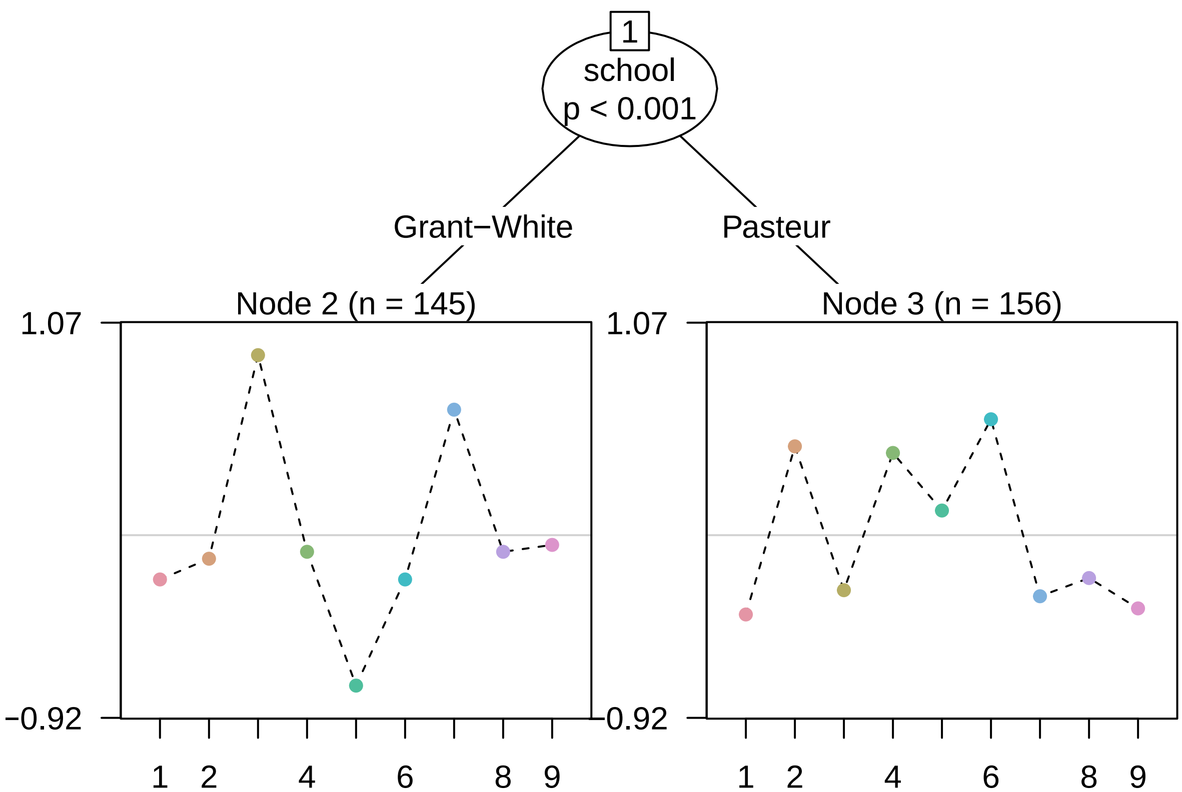
```r
summary(RT_out)
```

```
## $`2`
##
## Rasch model
##
## Difficulty parameters:
##           Estimate Std. Error z value Pr(>|z|)
## scalex2  1.045e-01  2.640e-01   0.396  0.69226
## scalex3  1.130e+00  2.767e-01   4.082 4.47e-05 ***
## scalex4  1.393e-01  2.640e-01   0.528  0.59773
## scalex5 -5.345e-01  2.684e-01  -1.992  0.04640 *
## scalex6 -8.612e-06  2.641e-01   0.000  0.99997
## scalex7  8.551e-01  2.706e-01   3.160  0.00158 **
## scalex8  1.393e-01  2.640e-01   0.528  0.59773
## scalex9  1.741e-01  2.640e-01   0.659  0.50971
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Log-likelihood: -497.3 (df = 8)
## Number of iterations in BFGS optimization: 11
```

```
##
##
## $`3`
##
## Rasch model
##
## Difficulty parameters:
##         Estimate Std. Error z value Pr(>|z|)
## scalex2  0.84676    0.25397   3.334 0.000856 ***
## scalex3  0.12236    0.24742   0.495 0.620919
## scalex4  0.81343    0.25338   3.210 0.001326 **
## scalex5  0.52341    0.24946   2.098 0.035888 *
## scalex6  0.98302    0.25668   3.830 0.000128 ***
## scalex7  0.09179    0.24742   0.371 0.710635
## scalex8  0.18350    0.24748   0.741 0.458410
## scalex9  0.03062    0.24749   0.124 0.901521
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Log-likelihood: -563.6 (df = 8)
## Number of iterations in BFGS optimization: 13
```

Plot it

```
plot(RT_out,col = rainbow_hcl(9))
```



More output from raschtree

```r
itempar(RT_out,node=1) # what would it be if no split
```

```
##    scalex1    scalex2    scalex3    scalex4    scalex5    scalex6
## -0.3082635  0.1784891  0.2614789  0.1784891 -0.2759642  0.1950127
##    scalex7    scalex8    scalex9
##  0.1291150 -0.1469228 -0.2114342
```

```r
itempar(RT_out, node = 2)
```

```
##     scalex1     scalex2     scalex3     scalex4     scalex5     scalex6
## -0.22302987 -0.11855399  0.90656550 -0.08373608 -0.75755971 -0.22303849
##     scalex7     scalex8     scalex9
##  0.63204820 -0.08373608 -0.04895948
```

```r
itempar(RT_out, node=3)
```

```
##    scalex1    scalex2    scalex3    scalex4    scalex5    scalex6
## -0.3994336  0.4473309 -0.2770740  0.4140004  0.1239758  0.5835858
##    scalex7    scalex8    scalex9
## -0.3076401 -0.2159363 -0.3688088
```

```r
coef(RT_out, node = 2)
```

```
##       scalex2       scalex3       scalex4       scalex5       scalex6
##  1.044759e-01  1.129595e+00  1.392938e-01 -5.345298e-01 -8.612114e-06
##       scalex7       scalex8       scalex9
##  8.550781e-01  1.392938e-01  1.740704e-01
```

DIFlasso

- this section causes problem

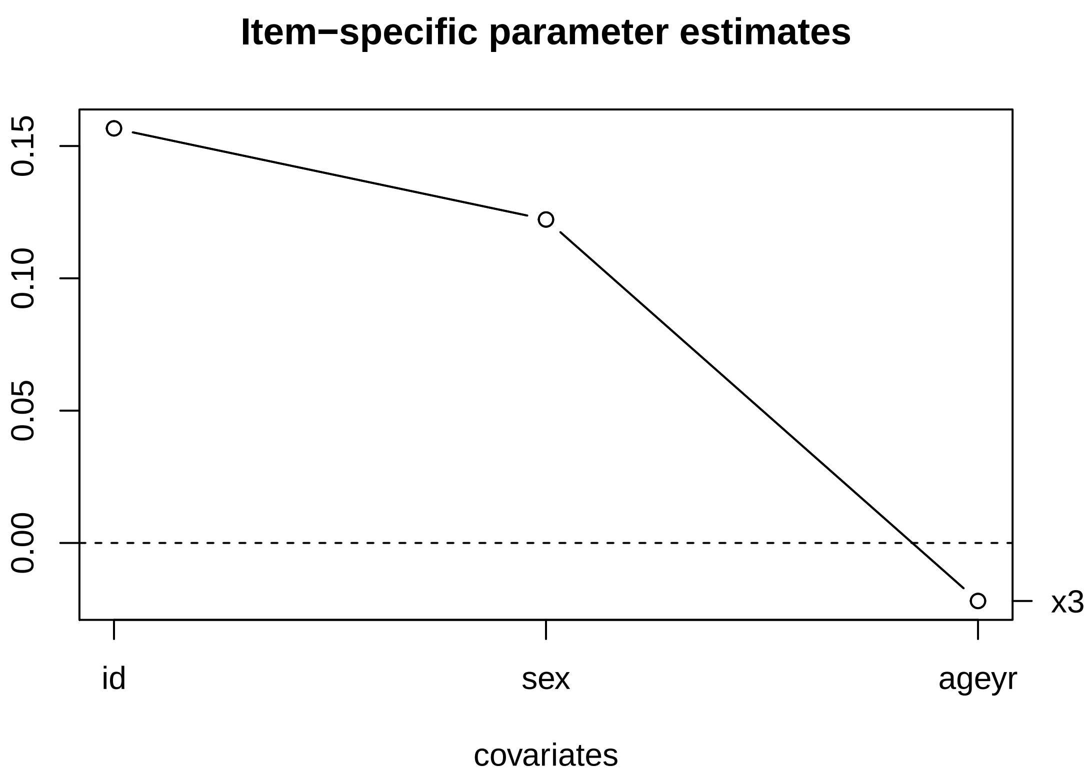
```r
library(DIFlasso)
```

```
## Loading required package: grplasso
## Loading required package: penalized
## Loading required package: survival
## Loading required package: splines
## Welcome to penalized. For extended examples, see vignette("penalized").
## Loading required package: miscTools
```

```r
Y <- data.frame(mydata$scale)
X <- sapply(HS[,1:3],as.numeric)
X.std <- data.frame(scale(X))
mlas1 <- DIFlasso(Y,X.std)
print(mlas1)
```

```
## Number of (valid) persons: P = 274
## Number of items: I = 9
## DIF Items: 3
##
## Matrix of estimated item-specific coefficients:
##       x1 x2          x3 x4 x5 x6 x7 x8 x9
## id     0  0  0.15666082  0  0  0  0  0  0
## sex    0  0  0.12222888  0  0  0  0  0  0
## ageyr  0  0 -0.02191697  0  0  0  0  0  0
```

```
plot(mlas1)
```



Can re-fit (Problem?)

```
mlas2 <- refitDIFlasso(mlas1)
mlas2
plot(mlas2)
```

Magis procedure: first load lassoDIF.R script and run functions http://jeb.sagepub.com/content/early/2014/12/16/1076998614559747.abstract

Have to reformat dataset – kinda weird (this is common with some IRT packages – particularly using mixed models for IRT) * each row is one item response with columns corresponding to:
* item number
* id number
* values on covariates
First reshape data

```r
HS$grade[is.na(HS$grade)] <- 7
library(reshape2)
# have to add ID variable that is a factor
hs.wide <- data.frame(HS[,1:4],HS.xs)
hs.wide$sum <- rowSums(HS.xs)
hs.wide$id <- as.factor(1:301)
HS$id <- as.factor(1:301)
xs <- c("x1","x2","x3","x4","x5","x6","x7","x8","x9")
hs.long <- melt(hs.wide, id.vars=c("id"),measure.vars =xs,variable.name="ITEM")

hs.long$school <- NA
hs.long$ageyr <- NA
hs.long$grade <- NA
hs.long$sex <- NA
hs.long$SCORE <- NA
for(i in 1:301){
  hs.long[hs.long$id == i,]$school <- HS[HS$id == i,]$school
  hs.long[hs.long$id == i,]$ageyr <- HS[HS$id == i,]$ageyr
  hs.long[hs.long$id == i,]$grade <- HS[HS$id == i,]$grade
  hs.long[hs.long$id == i,]$sex <- HS[HS$id == i,]$sex
  hs.long[hs.long$id == i,]$SCORE <- hs.wide[hs.wide$id == i,]$sum
}
names(hs.long)[3] <- "Y"
```

Dataset is set up, now have to change variable names in lassoDIF.R

– right now, only can use one covariate at a time

Change lines 7 and 63 to reflect covariate names

```
## Loading required package: Matrix
##
## Attaching package: 'Matrix'
##
## The following object is masked from 'package:OpenMx':
##
##     expm
##
## Loaded glmnet 1.9-8
```

```
## $DIFitems
## [1] 3 4 7
##
## $DIFpars
##              [,1]
## Item1  0.0000000
## Item2  0.0000000
## Item3 -0.8685773
## Item4  0.3229705
## Item5  0.0000000
## Item6  0.0000000
## Item7  0.3584199
```

```
## Item8  0.0000000
## Item9  0.0000000
##
## $crit.value
##  [1] 2925.450 2923.210 2921.330 2919.765 2918.463 2917.385 2916.485
##  [8] 2915.736 2916.754 2917.343 2916.072 2915.015 2914.136 2913.405
## [15] 2912.803 2912.298 2911.878 2911.528 2911.238 2910.997 2910.797
## [22] 2910.630 2912.473 2912.321 2912.194 2912.088 2913.979 2919.852
## [29] 2919.733 2919.637 2921.549 2921.472 2921.408 2921.380 2921.094
##
## $crit.type
## [1] "AIC"
##
## $lambda
##  [1] 0.0100153500 0.0091256139 0.0083149195 0.0075762450 0.0069031922
##  [6] 0.0062899317 0.0057311515 0.0052220118 0.0047581026 0.0043354059
## [11] 0.0039502604 0.0035993302 0.0032795756 0.0029882271 0.0027227613
## [16] 0.0024808787 0.0022604842 0.0020596690 0.0018766937 0.0017099734
## [21] 0.0015580641 0.0014196500 0.0012935322 0.0011786184 0.0010739131
## [26] 0.0009785097 0.0008915816 0.0008123759 0.0007402067 0.0006744488
## [31] 0.0006145326 0.0005599392 0.0005101958 0.0004648714 0.0000000000
##
## $opt.lambda
## [1] 0.00141965

## $DIFitems
## [1] 3 4 7
##
## $DIFpars
##                    factor(data$ITEM)x1
##                             -2.2656246
##                    factor(data$ITEM)x2
##                             -2.8179508
##                    factor(data$ITEM)x3
##                             -2.5643464
##                    factor(data$ITEM)x4
##                             -2.9015793
##                    factor(data$ITEM)x5
##                             -2.3022572
##                    factor(data$ITEM)x6
##                             -2.8366645
##                    factor(data$ITEM)x7
##                             -2.8638504
##                    factor(data$ITEM)x8
##                             -2.4487127
##                    factor(data$ITEM)x9
##                             -2.3754795
##                             data$SCORE
##                              0.5832833
## factor(data$ITEM)x1:factor(data$sex)2
##                              0.0000000
## factor(data$ITEM)x2:factor(data$sex)2
##                              0.0000000
## factor(data$ITEM)x3:factor(data$sex)2
```

```
##                             -0.7074783
## factor(data$ITEM)x4:factor(data$sex)2
##                              0.1664686
## factor(data$ITEM)x5:factor(data$sex)2
##                              0.0000000
## factor(data$ITEM)x6:factor(data$sex)2
##                              0.0000000
## factor(data$ITEM)x7:factor(data$sex)2
##                              0.2025925
## factor(data$ITEM)x8:factor(data$sex)2
##                              0.0000000
## factor(data$ITEM)x9:factor(data$sex)2
##                              0.0000000
##
## $opt.lambda
## [1] 0.002986456
```

Mixture models with rasch models
http://www2.uaem.mx/r-mirror/web/packages/psychomix/vignettes/raschmix.pdf
http://epm.sagepub.com/content/early/2014/06/20/0013164414536183

```
library(psychomix)
```

```
## Loading required package: flexmix
## Loading required package: lattice
```

```
hs.mat <- as.matrix(HS.xs)
m1 <- raschmix(hs.mat,k=1); BIC(m1)
```

```
## 1 : * * *
```

```
## [1] 3607.675
```

```
m2 <- raschmix(hs.mat,k=2); BIC(m2) # best BIC
```

```
## 2 : * * *
```

```
## [1] 3558.929
```

```
m3 <- raschmix(hs.mat,k=3); BIC(m3)
```

```
## 3 : * * *
```

```
## [1] 3617.097
```

```
m4 <- raschmix(hs.mat,k=4); BIC(m4)
```

```
## 4 : * * *
```

```
## [1] 3104.084
```

```
# won't give class for people with all wrong or right
hs.mat2 <- data.frame(hs.mat)
hs.mat2$sum <- rowSums(hs.mat)
hs.mat3 <- HS[hs.mat2$sum != 0 & hs.mat2$sum != 9,]
```

How do our derived classes correspond to covariates

```
library(psych)
```

```
##
## Attaching package: 'psych'
##
## The following object is masked from 'package:OpenMx':
##
##     tr
```

```
cor(hs.mat3$ageyr,m2@cluster)
```

```
## [1] -0.1987088
```

```
tetrachoric(data.frame(hs.mat3$sex,m2@cluster))
```

```
## Loading required package: mvtnorm
```

```
## Call: tetrachoric(x = data.frame(hs.mat3$sex, m2@cluster))
## tetrachoric correlation
##             hs.3. m2.cl
## hs.mat3.sex  1.00
## m2.cluster  -0.01  1.00
##
##  with tau of
## hs.mat3.sex  m2.cluster
##      -0.037      -0.073
```

```
cor(hs.mat3$grade[1:273],m2@cluster[1:273])
```

```
## [1] 0.05036862
```

```
tetrachoric(data.frame(as.numeric(hs.mat3$school),m2@cluster)) # highest
```

```
## Call: tetrachoric(x = data.frame(as.numeric(hs.mat3$school), m2@cluster))
## tetrachoric correlation
##                              a...3 m2.cl
## as.numeric.hs.mat3.school.  1.00
## m2.cluster                  -0.35  1.00
##
##  with tau of
## as.numeric.hs.mat3.school.                m2.cluster
##                     -0.055                    -0.073
```

Do we get similar results using the original dataset (back to CFA model)?

Factor Mixture Models in OpenMx – only package to do it in R (I think?)

```
resVars <-  mxPath(from=c("x1","x2","x3","x4","x5","x6","x7","x8","x9"),
                        arrows=2,
                        free=TRUE,
                        values=c(1.1,1.3,1.2,.4,.5,.35,1.1,1,.9),
                        labels=c("e1","e2","e3","e4","e5","e6","e7","e8","e9"))

latVars <- mxPath(from="F1",
                        arrows=2,
                        free=TRUE,
                        values=0.26,
                        labels ="varF1")

manMeans <- mxPath(from="one",
                        to=c("x1","x2","x3","x4","x5","x6","x7","x8","x9"),
                        arrows=1,
                        free=c(TRUE,TRUE,TRUE,TRUE,T,T,T,T,T),
                        values=c(4.93,6,2.2,3,4.3,2.1,4.1,5.5,5.3),
                        labels =c("meanx1","meanx2","meanx3","meanx4","meanx5",
                                  "meanx6","meanx7","meanx8","meanx9"))

loadings <- mxPath(from="F1",
                        to=c("x1","x2","x3","x4","x5","x6","x7","x8","x9"),
                        arrows=1,
                        free=c(FALSE,T,T,T,T,T,T,T,T),
                        values=c(1,0.5,0.5,1.9,2.1,1.8,0.4,0.4,0.6),
                        labels =c("l1","l2","l3","l4","l5","l6","l7","l8","l9"))

latMeans <- mxPath(from="one", to="F1", arrows=1,
                    free=TRUE, values=0, labels="meanF1")

funML <- mxFitFunctionML(vector=TRUE)


class1 <- mxModel("Class1", type="RAM",
                  manifestVars=c("x1","x2","x3","x4","x5","x6","x7","x8","x9"),
                  latentVars="F1",resVars,loadings,manMeans,latMeans,latVars,funML)

latVars2 <- mxPath(from="F1",
                        arrows=2,
                        free=TRUE,
                        values=2,
                        labels ="varF2")
# latent means
latMeans2 <- mxPath(from="one", to="F1", arrows=1,
                    free=TRUE, values=2, labels="meanF12")

class2 <- mxModel(class1, name="Class2", latVars2, latMeans2)

classP <- mxMatrix(type="Full", nrow=2, ncol=1,
                  free=c(TRUE, FALSE), values=1, lbound=0.001,
```

```r
                   labels = c("p1","p2"), name="Props")

classS <- mxAlgebra( Props %x% (1/sum(Props)), name="classProbs" )

algFit <- mxAlgebra(-2*sum(log(classProbs[1,1] %x% Class1.fitfunction
                     + classProbs[2,1] %x% Class2.fitfunction)),
                     name="mixtureObj")

fit <- mxFitFunctionAlgebra("mixtureObj")
data <- mxData(observed=HS,type="raw")
fmm <- mxModel("Factor Mixture Model",
                data, class1, class2, classP, classS, algFit, fit)
fmmFit <- mxRun(fmm, suppressWarnings=TRUE)
```

```
## Running Factor Mixture Model
```

```
## Warning in runHelper(model, frontendStart, intervals, silent,
## suppressWarnings, : Data[1] 'id' must be an ordered factor. Please use
## mxFactor()
```

```
## Warning in runHelper(model, frontendStart, intervals, silent,
## suppressWarnings, : Data[5] 'school' must be an ordered factor. Please use
## mxFactor()
```

```r
summary(fmmFit)
```

```
## Summary of Factor Mixture Model
##
## The final iterate satisfies the optimality conditions to the accuracy requested, but the sequence of
##
## free parameters:
##       name   matrix row col    Estimate  Std.Error lbound ubound
## 1       p1    Props   1   1  0.06635980 0.08489885  0.001
## 2       l2 Class1.A  x2  F1  0.51386809 0.14935345
## 3       l3 Class1.A  x3  F1  0.50222219 0.14319373
## 4       l4 Class1.A  x4  F1  1.91210984 0.25552610
## 5       l5 Class1.A  x5  F1  2.07753025 0.28224282
## 6       l6 Class1.A  x6  F1  1.80529839 0.24094992
## 7       l7 Class1.A  x7  F1  0.37836922 0.13658750
## 8       l8 Class1.A  x8  F1  0.39799560 0.12760302
## 9       l9 Class1.A  x9  F1  0.60620162 0.13496820
## 10      e1 Class1.S  x1  x1  1.09404236 0.09249306
## 11      e2 Class1.S  x2  x2  1.31197708 0.10774642
## 12      e3 Class1.S  x3  x3  1.20818632 0.09934191
## 13      e4 Class1.S  x4  x4  0.38419233 0.05026125
## 14      e5 Class1.S  x5  x5  0.51885367 0.06197367
## 15      e6 Class1.S  x6  x6  0.33484908 0.04588226
## 16      e7 Class1.S  x7  x7  1.14529164 0.09380039
## 17      e8 Class1.S  x8  x8  0.98011113 0.08039114
## 18      e9 Class1.S  x9  x9  0.91786530 0.07601051
## 19   varF1 Class1.S  F1  F1  0.09555741 0.12443066
## 20  meanx1 Class1.M   1  x1  2.99393779 2.25750854
```

```
## 21  meanx2 Class1.M   1  x2  5.09019349 1.19372265
## 22  meanx3 Class1.M   1  x3  1.27517696 1.16632559
## 23  meanx4 Class1.M   1  x4 -0.65208866 4.34044734
## 24  meanx5 Class1.M   1  x5  0.30631548 4.71458720
## 25  meanx6 Class1.M   1  x6 -1.32001657 4.09450651
## 26  meanx7 Class1.M   1  x7  3.45117730 0.89265394
## 27  meanx8 Class1.M   1  x8  4.75423529 0.93424936
## 28  meanx9 Class1.M   1  x9  4.19698140 1.39238097
## 29  meanF1 Class1.M   1  F1  2.95915559 2.29404245
## 30   varF2 Class2.S  F1  F1  0.20230226 0.06537716
## 31 meanF12 Class2.M   1  F1  1.87432208 2.25728364
##
## observed statistics:  2709
## estimated parameters:  31
## degrees of freedom:  2678
## -2 log likelihood:  7698.128
## number of observations:  301
## Information Criteria:
##       |  df Penalty  |  Parameters Penalty  |  Sample-Size Adjusted
## AIC:       2342.128              7760.128                       NA
## BIC:      -7585.513              7875.048                 7776.734
## CFI: NA
## TLI: 1
## RMSEA:  0  [95% CI (NA, NA)]
## Some of your fit indices are missing.
##   To get them, fit saturated and independence models, and include them with
##   summary(yourModel, SaturatedLikelihood=..., IndependenceLikelihood=...).
## timestamp: 2015-05-05 16:27:07
## Wall clock time (HH:MM:SS.hh): 00:00:02.48
## optimizer:  NPSOL
## OpenMx version number: 2.0.1.4157
## Need help?  See help(mxSummary)
```

```
fmmFit$classProbs
```

```
## mxAlgebra 'classProbs'
## $formula:  Props %x% (1/sum(Props))
## $result:
##            [,1]
## [1,] 0.06223022
## [2,] 0.93776978
## dimnames: NULL
```

```
#str(fmmFit)

fmmFit$submodels$Class2$fitfunction$likelihoods
```

```
## NULL
```

```
comp <- fmmFit$output$algebras$Class1.fitfunction > fmmFit$output$algebras$Class2.fitfunction
sum(comp)/301
```

```
## [1] 0.1594684
```

```r
# http://openmx.psyc.virginia.edu/thread/717
indClassProbs <- function(model, classProbs, round=NA){
  # this function takes a mixture model in OpenMx
  # and returns the posterior class probabilities
    # using Bayes rule, individual person-class likelihoods
    # and the model class probability matrix, as described in
    # Ramaswamy, Desarbo, Reibstein, and Robinson, 1993
    cp <- mxEval(classProbs, model)
    cp2 <- as.vector(cp)
    cps <- diag(length(cp2))
    diag(cps) <- cp2
    subs <- model@submodels
    if(min(dim(cp))!=1)stop("Class probabilities matrix must be a row or column vector.")
    if(max(dim(cp))==1)stop("Class probabilities matrix must contain two or more classes.")
    of <- function(num){
        return(mxEval(objective, subs[[num]]))
        }
    rl <- sapply(1:length(names(subs)), of)
    raw <- (rl%*%cps)
    tot <- 1/apply(raw, 1, sum)
    div <- matrix(rep(tot, length(cp2)), ncol=length(cp2))
    icp <- raw * div
    if (is.numeric(round)){icp <- round(icp, round)}
    return(icp)
    }

#indClassProbs(fmmFit,fmmFit$classProbs)


prbs <- indClassProbs(fmmFit,fmmFit$classProbs)[,1]

HS$prbs <- prbs
lmm = lm(prbs ~ sex + ageyr + school + grade, data=HS)
summary(lmm)
```

```
##
## Call:
## lm(formula = prbs ~ sex + ageyr + school + grade, data = HS)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.14913 -0.07936 -0.04227 -0.00244  0.81070
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)   -0.15832    0.15328  -1.033  0.30251
## sex            0.02034    0.01876   1.084  0.27909
## ageyr         -0.02107    0.01077  -1.957  0.05124 .
## schoolPasteur -0.04529    0.01916  -2.364  0.01875 *
## grade          0.06516    0.02159   3.018  0.00277 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## Residual standard error: 0.1597 on 296 degrees of freedom
## Multiple R-squared:  0.06175,    Adjusted R-squared:  0.04907
## F-statistic: 4.871 on 4 and 296 DF,  p-value: 0.0008108
```

```
cor(prbs,HS$sex)
```

```
## [1] 0.07661222
```

```
cor(prbs,HS$ageyr)
```

```
## [1] -0.07827931
```

```
cor(prbs,as.numeric(HS$school))
```

```
## [1] -0.1605882
```

```
cor(prbs[1:300],HS$grade[1:300])
```

```
## [1] 0.1214811
```

```
# compare 2 derived classes from both mixture models
cor(m2@cluster,prbs[hs.mat2$sum != 0 & hs.mat2$sum != 9])
```

```
## [1] 0.3488794
```

Moderation Try

```
resVars <-  mxPath(from=c("x1","x2","x3","x4","x5","x6","x7","x8","x9"),
                        arrows=2,
                        free=TRUE,
                        values=c(1.1,1.3,1.2,.4,.5,.35,1.1,1,.9),
                        labels=c("e1","e2","e3","e4","e5","e6","e7","e8","e9"))

latVars <- mxPath(from="F1",
                        arrows=2,
                        free=TRUE,
                        values=0.26,
                        labels ="varF1")

manMeans <- mxPath(from="one",
                        to=c("x1","x2","x3","x4","x5","x6","x7","x8","x9"),
                        arrows=1,
                        free=c(TRUE,TRUE,TRUE,TRUE,T,T,T,T,T),
                        values=c(4.93,6,2.2,3,4.3,2.1,4.1,5.5,5.3),
                        labels =c("meanx1","meanx2","meanx3","meanx4","meanx5",
                                  "meanx6","meanx7","meanx8","meanx9"))

loadings <- mxPath(from="F1",
                        to=c("x1","x2","x3","x4","x5","x6","x7","x8","x9"),
```

```
                        arrows=1,
                        free=c(FALSE,T,T,T,T,T,T,T,T),
                        values=c(1,0.5,0.5,1.9,2.1,1.8,0.4,0.4,0.6),
                        labels =c("l1","l2","l3","l4","l5","l6","l7","l8","l9"))

latMeans <- mxPath(from="one", to="F1", arrows=1,
                    free=TRUE, values=0, labels="meanF1")

defValues    <- mxPath( from="one", to="DefDummy", arrows=1,
                        free=FALSE, values=1, labels="data.sex" )

# beta weights
betaWeights  <- mxPath( from="DefDummy", to=c("F1"), arrows=1,
                        free=TRUE, values=1, labels=c("beta_1") )

#LoadsDef           <- mxAlgebra(expression= l2 + beta*ageyr, name="l22" )


funML <- mxFitFunctionML()

HS.def <- HS[,c(2,7:15)]
HS.def$sex <- mxFactor(HS.def$sex,levels=c(1,2))

data <- mxData(observed=HS.def,type="raw")

Onefac <- mxModel("One fac mod",type="RAM",
                manifestVars=c("x1","x2","x3","x4","x5","x6","x7","x8","x9"),data,
                  latentVars=c("F1","DefDummy"),resVars,loadings,manMeans,latMeans,latVars,funML,
                betaWeights,defValues)
MxOut <- mxTryHard(Onefac)
```

```
## Running One fac mod
## Running One fac mod
```

```
## [1] "0.559080717875726,0.383995662617096,1.84923001372587,2.5643636267348,1.54800576912716,0.4245372
```

```
summary(MxOut)
```

```
## Summary of One fac mod
##
## free parameters:
##      name matrix row      col     Estimate    Std.Error
## 1      l2      A   x2       F1   0.50567220   0.15104496
## 2      l3      A   x3       F1   0.48979941   0.14476069
## 3      l4      A   x4       F1   1.93826925   0.26280317
## 4      l5      A   x5       F1   2.12886901   0.29452725
## 5      l6      A   x6       F1   1.79830966   0.24498422
## 6      l7      A   x7       F1   0.38749251   0.13851723
## 7      l8      A   x8       F1   0.39806993   0.12884667
## 8      l9      A   x9       F1   0.60688014   0.13870985
## 9  beta_1      A   F1 DefDummy   0.06169925   0.06281313
## 10     e1      S   x1       x1   1.09915509   0.09323576
```

```
## 11     e2      S  x2       x2   1.31550147   0.10800864
## 12     e3      S  x3       x3   1.21267905   0.10102482
## 13     e4      S  x4       x4   0.37682497   0.04769170
## 14     e5      S  x5       x5   0.48500398   0.05906114
## 15     e6      S  x6       x6   0.35807976   0.04308754
## 16     e7      S  x7       x7   1.14421813   0.09385792
## 17     e8      S  x8       x8   0.98090796   0.08062202
## 18     e9      S  x9       x9   0.91953318   0.07612169
## 19  varF1      S  F1       F1   0.25826362   0.06991137
## 20 meanx1      M   1       x1  17.85520538 255.36753976
## 21 meanx2      M   1       x2  12.62103944 128.96383482
## 22 meanx3      M   1       x3   8.57834721 124.87524576
## 23 meanx4      M   1       x4  28.10225247 494.42447762
## 24 meanx5      M   1       x5  31.84431765 542.94251559
## 25 meanx6      M   1       x6  25.41871746 458.70431318
## 26 meanx7      M   1       x7   9.19208635  99.10780408
## 27 meanx8      M   1       x8  10.66991493 101.62481516
## 28 meanx9      M   1       x9  13.21467209 155.30469242
## 29 meanF1      M   1       F1 -13.01290671 255.36778656
##
## observed statistics:  2709
## estimated parameters:  29
## degrees of freedom:  2680
## -2 log likelihood:  7701.475
## number of observations:  301
## Information Criteria:
##       |  df Penalty  |  Parameters Penalty  |  Sample-Size Adjusted
## AIC:       2341.475               7759.475                         NA
## BIC:      -7593.580               7866.982                    7775.01
## Some of your fit indices are missing.
##   To get them, fit saturated and independence models, and include them with
##   summary(yourModel, SaturatedLikelihood=..., IndependenceLikelihood=...).
## timestamp: 2015-05-05 16:27:09
## Wall clock time (HH:MM:SS.hh): 00:00:00.97
## optimizer:  NPSOL
## OpenMx version number: 2.0.1.4157
## Need help?  See help(mxSummary)
```