Chapter 5: adjusted MIC

R-code to estimate the adjusted MIC, predictive MIC and ROC-based MIC

This code simulates an example data set with 10 items that are measured at two time points.

```
# Acquire packages
library(mirt)     # for simulating data
library(lavaan)   # for estimating the transition ratings reliability
library(pROC)     # for estimating ROC-based MIC
```

Simulate dataset using 'mirt'

Gernerate a set of IRT item parameters

```
set.seed(12345)

b2  <- c(-0.8, -0.8, -0.4, -0.4, 0, 0, 0.4, 0.4, 0.8, 0.8)
bc <- b2/4
b1 <- b2 - 1 + sample(bc)
b3 <- b2 + 1 + sample(bc)
a1 <- sample( 1+b2/2 )

( cf.simb <- data.frame(a1,b1,b2,b3) )
```

```
##     a1   b1   b2           b3
## 1  1.4 -1.9 -0.8  4.000000e-01
## 2  0.8 -1.7 -0.8 -5.551115e-17
## 3  0.6 -1.6 -0.4  7.000000e-01
## 4  1.2 -1.4 -0.4  7.000000e-01
## 5  1.0 -0.8  0.0  1.000000e+00
## 6  1.2 -0.8  0.0  1.200000e+00
## 7  1.4 -0.5  0.4  1.300000e+00
## 8  0.8 -0.6  0.4  1.200000e+00
## 9  1.0 -0.3  0.8  1.700000e+00
## 10 0.6 -0.4  0.8  1.800000e+00
```

Transform b-parameters to d-parameters (mirt works with d-parameters)

difficulty (b) = easiness (d) / -a

```
cf.sim <- cf.simb

colnames(cf.sim) <- c("a1","d1","d2","d3")

cf.sim$d1 <- -cf.simb$b1*cf.sim$a1
cf.sim$d2 <- -cf.simb$b2*cf.sim$a1
cf.sim$d3 <- -cf.simb$b3*cf.sim$a1
round(cf.sim, 3)
```

```
##     a1   d1    d2    d3
## 1  1.4 2.66  1.12 -0.56
## 2  0.8 1.36  0.64  0.00
## 3  0.6 0.96  0.24 -0.42
## 4  1.2 1.68  0.48 -0.84
## 5  1.0 0.80  0.00 -1.00
## 6  1.2 0.96  0.00 -1.44
## 7  1.4 0.70 -0.56 -1.82
## 8  0.8 0.48 -0.32 -0.96
## 9  1.0 0.30 -0.80 -1.70
## 10 0.6 0.24 -0.48 -1.08
```

```
round(colMeans(cf.sim), 3)
```

```
##     a1     d1     d2     d3
##  1.000  1.014  0.032 -0.982
```

Create theta T1 and dataset 1 (baseline)

```
a1 <- as.matrix(cf.sim[ , 1])
d1 <- as.matrix(cf.sim[ , -1])

N <- 1000         # sample size
prop.imp <- 0.3   # proportion improved
mn.imic  <- 0.5   # mean individual MICs in terms of theta change

tet1s <- rnorm(N, 0, 1)   # simulate theta T1
tet1s <- as.matrix(tet1s)

set.seed( sample(10000:20000,1) )

dat1 <- simdata(a1, d1, N, itemtype="graded", Theta=tet1s)
dat1 <- as.data.frame(dat1)

xo1 <- rowSums(dat1)   # This is the baseline test (sum) score
```

Create theta change

```
rt1ch <- -0.5            # correlation between theta T1 and theta change
tetchs <- rt1ch*tet1s + sqrt(1-rt1ch^2)*rnorm(N, mean(tet1s), sd(tet1s))
cor(tet1s,tetchs)
```

```
##             [,1]
## [1,] -0.5012465
```

```
tetchs <- tetchs - mean(tetchs)          # makes mean = 0
tetchs <- (tetchs/sd(tetchs))*1          # makes SD = 1

( qtl <- quantile(tetchs, prob=(1-prop.imp)) )
```

```
##       70%
## 0.4724834
```

```
( mean.tetchs <- mn.imic - qtl )         # Estimate mean theta change to get
```

```
##        70%
## 0.02751659
```

```
# the desired proportion improved
tetchs <- tetchs + mean.tetchs           # transform mean of tetchs
```

Create theta T2 and dataset 2 (follow-up)

```
tet2s <- as.matrix(tet1s + tetchs)     # theta T2

cor(tet1s, tet2s)
```

```
##           [,1]
## [1,] 0.4855813
```

```
set.seed( sample(30000:40000,1) )

dat2 <- simdata(a1, d1, N, itemtype="graded", Theta=tet2s)
dat2 <- as.data.frame(dat2)
```

Insert error in transition ratings

```
rel.trat <- 0.5   # reliability of the perceived change/transition rating

( sd.ch.error <- sqrt(((1-rel.trat)/rel.trat)*sd(tetchs)^2) )
```

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

```
tetch.percv <- tetchs + rnorm(N, 0, sd.ch.error)  # perceived change
```

Create iMIC distribution

```
imic <- rnorm(N, mn.imic, 0.1*mn.imic)
```

create dichotomous transition ratings

```
trat <- numeric(N)
trat[tetch.percv > imic] <- 1
mean(trat)                              # proportion improved based on perceived change
```

```
## [1] 0.355
```

Create dataset

```
org <- data.frame(dat1, dat2, trat)
head(org)
```

```
##   Item_1 Item_2 Item_3 Item_4 Item_5 Item_6 Item_7 Item_8 Item_9 Item_10
## 1      1      0      1      0      0      0      0      0      0       2
## 2      3      2      0      2      2      0      1      1      3       0
## 3      3      0      0      3      0      3      0      0      2       0
## 4      2      0      0      0      2      1      0      0      0       3
## 5      3      2      2      1      0      3      1      1      0       1
## 6      3      3      2      0      0      0      0      3      0       0
##   Item_1.1 Item_2.1 Item_3.1 Item_4.1 Item_5.1 Item_6.1 Item_7.1 Item_8.1
## 1        1        3        1        2        0        0        0        2
## 2        2        0        2        2        2        2        2        1
## 3        3        3        3        3        3        3        3        2
## 4        0        3        1        2        2        1        2        3
## 5        3        1        3        3        3        0        2        3
## 6        3        0        3        0        0        1        3        0
##   Item_9.1 Item_10.1 trat
## 1        0         0    1
## 2        0         0    0
## 3        2         2    1
## 4        0         1    0
## 5        2         3    1
## 6        0         0    0
```

```
nitems = 10     # Provide the number of items in the scale

## Simplify/standardize the item names
names(org)[1:nitems] <- paste0('v1', '_', 1:nitems)
names(org)[(nitems+1):(2*nitems)] <- paste0('v2', '_', 1:nitems)
names(org)[2*nitems+1] <- "trat"
names(org)
```

```
##  [1] "v1_1"  "v1_2"  "v1_3"  "v1_4"  "v1_5"  "v1_6"  "v1_7"  "v1_8"  "v1_9"
## [10] "v1_10" "v2_1"  "v2_2"  "v2_3"  "v2_4"  "v2_5"  "v2_6"  "v2_7"  "v2_8"
## [19] "v2_9"  "v2_10" "trat"
```

```
## copy original data into a workfile (dat)
dat <- org
head(dat)
```

```
##   v1_1 v1_2 v1_3 v1_4 v1_5 v1_6 v1_7 v1_8 v1_9 v1_10 v2_1 v2_2 v2_3 v2_4 v2_5
## 1    1    0    1    0    0    0    0    0    0     2    1    3    1    2    0
## 2    3    2    0    2    2    0    1    1    3     0    2    0    2    2    2
## 3    3    0    0    3    0    3    0    0    2     0    3    3    3    3    3
## 4    2    0    0    0    2    1    0    0    0     3    0    3    1    2    2
## 5    3    2    2    1    0    3    1    1    0     1    3    1    3    3    3
## 6    3    3    2    0    0    0    0    3    0     0    3    0    3    0    0
##   v2_6 v2_7 v2_8 v2_9 v2_10 trat
## 1    0    0    2    0     0    1
## 2    2    2    1    0     0    0
## 3    3    3    2    2     2    1
## 4    1    2    3    0     1    0
## 5    0    2    3    2     3    1
## 6    1    3    0    0     0    0
```

Estimate TR reliability with lavaan

Specify the factor model

A factor for each time point, with correlated errors over time.

```
model <- '
   # Factors
   F1 =~ v1_1+v1_2+v1_3+v1_4+v1_5+v1_6+v1_7+v1_8+
         v1_9+v1_10+trat
   F2 =~ v2_1+v2_2+v2_3+v2_4+v2_5+v2_6+v2_7+v2_8+
         v2_9+v2_10+trat

   # Correlated errors over time
   v1_1 ~~ v2_1
   v1_2 ~~ v2_2
   v1_3 ~~ v2_3
   v1_4 ~~ v2_4
   v1_5 ~~ v2_5
   v1_6 ~~ v2_6
   v1_7 ~~ v2_7
   v1_8 ~~ v2_8
   v1_9 ~~ v2_9
   v1_10 ~~ v2_10
   '
```

Fit the confirmatory factor analysis

```
fit <- cfa(model, data=dat, ordered=T)

fitMeasures(fit, fit.measures = c("cfi.scaled","tli.scaled","rmsea.scaled",
```

```
                                  "rmsea.ci.lower.scaled","rmsea.ci.upper.scaled","rmsea.pvalue.scaled"
                                  "srmr") )
```

```
##            cfi.scaled            tli.scaled          rmsea.scaled
##                 1.000                 1.003                 0.000
## rmsea.ci.lower.scaled rmsea.ci.upper.scaled   rmsea.pvalue.scaled
##                 0.000                 0.011                 1.000
##                  srmr
##                 0.029
```

```
MI <- modificationIndices(fit)
MI[MI$sepc.lv>0.3,]
```

```
## [1] lhs      op       rhs      mi       epc      sepc.lv  sepc.all sepc.nox
## <0 rows> (or 0-length row.names)
```

Note: Check the model fit. If necessary improve model fit by allowing correlated errors cross-sectionally (e.g., v1_1 ~~ v1_2, v2_1 ~~ v2_2)

Reliability of TR

```
pe <- parameterEstimates(fit, standardized=T, rsquare=T)
( rel.trat <- pe$est[pe$lhs=="trat" & pe$op=="r2"] )   # Reliability TR
```

```
## [1] 0.4916676
```

Estimate MICs

ROC-based MIC

```
xo1 <- rowSums(dat[,1:nitems])                 # sumscore T1
xo2 <- rowSums(dat[,(nitems+1):(2*nitems)])    # sumscore T2
dat$xoc <- xo2 - xo1                           # change score

( q <- mean(dat$trat) )            # q = proportion improved (perceived)
```

```
## [1] 0.355
```

```
( p <- log(q/(1-q)) )              # p = logodds(pre)
```

```
## [1] -0.5971325
```

```
( cor.trat.xoc <- cor(dat$trat, dat$xoc) )  # correlation anchor-change
```

```
## [1] 0.4044711
```

```
## Do ROC analysis

rocobj <- roc(dat$trat, dat$xoc, quiet = TRUE)

( mic.roc <- coords(rocobj, x="best", input="threshold", ret="threshold",
                    best.method="youden", transpose = TRUE) )
```

```
## threshold
##      -0.5
```

predicted MIC

Apply a logistic regression for both the MIC(predicted) and MIC(adjusted)

```
mylogit <- glm(trat ~ xoc, data = dat, family = "binomial")

C <- coef(mylogit)[1]                 # intercept coefficient C
B <- coef(mylogit)[2]                 # regression coefficient B

( mic.pred <- (p-C)/B )               # MIC(predicted)
```

```
## (Intercept)
##   0.8484085
```

Adjusted MIC

```
rf <- (0.8/rel.trat - 0.5) * sd(dat$xoc) * cor.trat.xoc
( mic.adj <- mic.pred - rf * p )
```

```
## (Intercept)
##     2.80477
```

Bootstrap confidence intervals

Bootstrap code and application

```
nboot    <- 1000                # number of bootstrap samples
mic.roc  <- numeric(nboot)
mic.pred <- numeric(nboot)
mic.adj  <- numeric(nboot)
boot <- data.frame(mic.roc, mic.pred, mic.adj)

start.time <- Sys.time()

for(i in 1:nboot) {

  #print(i)
```

```
  dat <- org[sample(1:dim(dat)[1], dim(dat)[1], replace=TRUE),]

  ### Estimate TR reliability with lavaan

  # model is already defined in the previous section

  fit <- cfa(model, data=dat, ordered=T,
             test="mean.var.adjusted")

  pe <- parameterEstimates(fit, standardized=T, rsquare=T)
  ( rel.trat <- pe$est[pe$lhs=="trat" & pe$op=="r2"] )   # Reliability TR

  ### Estimate MICs

  xo1 <- rowSums(dat[,1:nitems])                  # sumscore T1
  xo2 <- rowSums(dat[,(nitems+1):(2*nitems)])     # sumscore T2
  dat$xoc <- xo2 - xo1

  ( q <- mean(dat$trat) )            # q = proportion improved (perceived)
  ( p <- log(q/(1-q)) )              # p = logodds(pre)

  ( cor.trat.xoc <- cor(dat$trat, dat$xoc) )  # correlation anchor-change

  ## Do ROC analysis

  rocobj <- roc(dat$trat, dat$xoc, quiet = TRUE)

  mic.roc <- coords(rocobj, x="best", input="threshold", ret="threshold",
                    best.method="youden", transpose = TRUE)

  ( boot$mic.roc[i] <- mic.roc[sample(length(mic.roc),1)] )


  ## apply logistic regression and calculate parameters and MIC(pred)

  mylogit <- glm(trat ~ xoc, data = dat, family = "binomial")

  C <- coef(mylogit)[1]                 # intercept coefficient C
  B <- coef(mylogit)[2]                 # regression coefficient B

  ( boot$mic.pred[i] <- (p-C)/B )               # MIC(predicted)

  ## Adjusted MIC
  rf <- (0.8/rel.trat - 0.5) * sd(dat$xoc) * cor.trat.xoc
  ( boot$mic.adj[i] <- boot$mic.pred[i] - rf * p )

}


end.time <- Sys.time()
time.taken <- end.time - start.time
time.taken
```

```
## Time difference of 16.69499 mins
```

Bootstrap results

```
round(mean(boot$mic.roc),1)
```

```
## [1] -0.4
```

```
round(mean(boot$mic.pred),1)
```

```
## [1] 0.8
```

```
round(mean(boot$mic.adj),1)
```

```
## [1] 2.8
```

```
round(quantile(boot$mic.roc, c(0.025, 0.975)),1)
```

```
##  2.5% 97.5%
##  -2.5   2.5
```

```
round(quantile(boot$mic.pred, c(0.025, 0.975)),1)
```

```
##  2.5% 97.5%
##   0.4   1.3
```

```
round(quantile(boot$mic.adj, c(0.025, 0.975)),1)
```

```
##  2.5% 97.5%
##   2.1   3.5
```