APPENDICES

Appendix 1. Sample application of the "sim" function.

A description of the "sim" function for applying the Monte Carlo method can be found in the R documentation. Using the "sim" function requires prior installation of the "simsem" package. The following R codes show an example of the application of the "sim" function with structural equations derived from the A-I-C model.

```
require(simsem)

# Population model
popModel <- 'f2 =~ 0.78*y1
            f2 =~ 0.86*y2
            f2 =~ -0.74*y3
            f1 =~ 0.74*y4
            f1 =~ 0.85*y5
            f1 =~ 0.64*y6
            f3 =~ 0.82*y7
            f3 =~ 0.86*y8
            f3 =~ 0.84*y9
            f4=~ 0.90*y10
            f4=~ 0.87*y11
            f4=~ 0.78*y12
            f5=~ 0.68*y13
            f5=~ 0.78*y14
            f5=~ 0.71*y15
            f6=~ 0.84*y16
            f6=~ -0.73*y17
```

```
                                f6=~ 0.87*y18
                                f7=~ 0.77*y19
                                f7=~ 0.75*y20
                                f7=~ -0.74*y21
                                f8=~ 0.87*y22
                                f8=~ -0.60*y23
                                f8=~ 0.89*y24

                                f1 ~ 0.24*f2 + (-0.10)*f6 + 0.49*f7 + 0.295*f8
                                f2 ~ 0.16*f3 + 0.14*f4 + 0.72*f5
                                f2 ~~ 0.16*f2
                                f1 ~~ 0.32*f1    '

# Analysis model

analysisModel           <-    'f2 =~ y1
                                f2 =~ y2
                                f2 =~ y3
                                f1 =~ y4
                                f1 =~ y5
                                f1 =~ y6
                                f3 =~ y7
                                f3 =~ y8
                                f3 =~ y9
                                f4=~ y10
                                f4=~ y11
                                f4=~ y12
                                f5=~ y13
                                f5=~ y14
```

```
f5=~ y15
f6=~ y16
f6=~ y17
f6=~ y18
f7=~ y19
f7=~ y20
f7=~ y21
f8=~ y22
f8=~ y23
f8=~ y24

f1 ~ f2 + f6 + f7 + f8
f2 ~ f3 + f4 + f5
f2 ~~ 0.16*f2
f1 ~~ 0.32*f1  '

Output <- sim(1000, model=analysisModel, n=150, generate=list(model = popModel, skewness = rep(3,24), kurtosis = rep(21,24)), std.lv=TRUE, lavaanfun = "sem",estimator = "ml")

summary(Output)

design4_3_2 <- data.frame(CHISQ = Output@fit$chisq, CFI = Output@fit$cfi, TLI = Output@fit$tli, RMSEA = Output@fit$rmsea, SRMR = Output@fit$srmr, GFI = Output@fit$gfi, AGFI = Output@fit$agfi)
```

Appendix 2. Results of the Kolmogorov-Smirnov normality test.

Indices	Statistics	df	p-value
CFI	0,311	60000	0,000
TLI	0,213	60000	0,000

RMSEA	0,355	60000	0,000
SRMR	0,114	60000	0,000
GFI	0,124	60000	0,000
AGFI	0,120	60000	0,000

Appendix 3. Example of applying Wilcox (2012) robust method on R.

R codes for AGFI:

```
Require(WRS2)

> t3way(formula = as.numeric(AGFI) ~ as.factor(EM) * as.factor(NN) * as.factor(SS), data = fullData2)

Call:

t3way(formula = as.numeric(AGFI) ~ as.factor(EM) * as.factor(NN) * as.factor(SS), data = fullData2)
```

	Value	p-value
as.factor(EM)	3610.802	1e-04
as.factor(NN)	108674.080	1e-04
as.factor(SS)	912898.042	1e-03
as.factor(EM):as.factor(NN)	128081.918	1e-03
as.factor(EM):as.factor(SS)	8297.563	1e-03
as.factor(NN):as.factor(SS)	10641.188	1e-03
as.factor(EM):as.factor(NN):as.factor(SS)	12641.994	1e-03