Notes on three VM-based simulation studies for ordinal SEM

Supplementary material for "A Problem with Discretizing Vale-Maurelli in Simulation Studies"

January, 2019

Contents

1 Introduction

In this report we re-visit three influential and representative simulation studies in the field of ordinal SEM. There are many similar simulation studies, and our choice of these three studies was motivated by their visibility, i.e., their high citation counts. We could have added similar analyses for other articles, some of which are listed in ("A Problem with Discretizing Vale-Maurelli in Simulation Studies" 2019).

We emphasize that the three studies under scrutiny here are well-written and well-conducted, and contain many important findings that are useful to SEM practitioners. In the present review, we are solely concerned with claims of robustness to violation of non-normality that these papers put forth. Our aim is to demonstrate that these claims are not justified due to the way ordinal data were generated in these studies. This fact has just been recently discovered, and was unknown at the time of publication of these three studies.

Each of the three studies has as a central aim to determine how polychorics and SEM estimation are affected by violation of the underlying normality assumption of polychoric correlations. In these studies non-normality was generated with the Vale-Maurelli transform. We will look specifically in each study at the VM distribution that we think was discretised (the studies used mplus software for VM, and did not provide information on the specific Fleishman polynomials). In the majority of conditions the Fleishman polynomial is monotonous. In such conditions the discretised data are indistinguishable from data generated from a *multivariate normal vector* ("A Problem with Discretizing Vale-Maurelli in Simulation Studies" 2019), whose correlation matrix may depart from the target correlation matrix for the VM distribution. This correlation matrix is in fact the population polychoric correlation matrix. So, although the explicit intention in these studies is to discretize a truly non-normal vector, in reality they simulate from a multivariate normal vector whose correlation possibly departs from the target correlation. Our aim is to identify this correlation matrix,

i.e., the polychoric correlation matrix, and to fit the model to it, in order to obtain population values for the model parameters. We will see that that parameter estimates are only slightly off target, which explains why these studies conclude that parameter bias is negligible for methods based on polychorics (cat-DWLS and cat-LS). We conjecture that using a truly non-normal vector to discretize will shift the polychoric estimates far away from the target, and thus create severe parameter bias. This would contradict the findings in the three highly cited papers under review here.

The final section of the present report contains source code for the illustration section in ("A Problem with Discretizing Vale-Maurelli in Simulation Studies" 2019). All calculations here presented are based on the R software environment, and specifically the package lavaaan (Rosseel 2012). We first initialize R by loading packages and defining auxiliary functions. We assume that the reader is familiar with the model syntax used by lavaan.

```
knitr::opts_chunk$set(echo = TRUE)
knitr::opts_chunk$set(cache = TRUE)
knitr::opts_chunk$set(message   = FALSE)
knitr::opts_chunk$set(warning = FALSE)
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------------------- tidyverse 1.2.1 --
```

```
## ✓ ggplot2 3.1.0     ✓ purrr   0.2.5
## ✓ tibble  1.4.2     ✓ dplyr   0.7.8
## ✓ tidyr   0.8.1     ✓ stringr 1.3.1
## ✓ readr   1.1.1     ✓ forcats 0.3.0
```

```
## -- Conflicts ------------------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(MASS)
```

```
##
## Attaching package: 'MASS'
```

```
## The following object is masked from 'package:dplyr':
##
##     select
```

```
library(psych)
```

```
##
## Attaching package: 'psych'
```

```
## The following objects are masked from 'package:ggplot2':
##
##     %+%, alpha
```

```
library(lavaan)
```

```
## This is lavaan 0.6-3
```

```
## lavaan is BETA software! Please report any bugs.
```

```
##
## Attaching package: 'lavaan'
```

```
## The following object is masked from 'package:psych':
##
##     cor2cov
```

```r
#auxiliary functions
is.monotone <- function(f){
  b=f[2]; c=f[3]; d=f[4]
  return( 4*c^2-12*b*d <0 )
}

getVM<- function(skewness, kurtosis, sigma0){
    COR = cov2cor(sigma0)
    nvar <- ncol(COR)
    # check skewness
    if(is.null(skewness)) {
        SK <- rep(0, nvar)
    } else if(length(skewness) == nvar) {
        SK <- skewness
    } else if(length(skewness) == 1L) {
        SK <- rep(skewness, nvar)
    } else {
        stop("skewness has wrong length")
    }

    if(is.null(kurtosis)) {
        KU <- rep(0, nvar)
    } else if(length(kurtosis) == nvar) {
        KU <- kurtosis
    } else if(length(kurtosis) == 1L) {
        KU <- rep(kurtosis, nvar)
    } else {
        stop("kurtosis has wrong length")
    }
    fleishman1978_abcd <- function(skewness, kurtosis) {
        system.function <- function(x, skewness, kurtosis) {
            b.=x[1L]; c.=x[2L]; d.=x[3L]
            eq1 <- b.*b. + 6*b.*d. + 2*c.*c. + 15*d.*d. - 1
            eq2 <- 2*c.*(b.*b. + 24*b.*d. + 105*d.*d. + 2) - skewness
            eq3 <- 24*(b.*d. + c.*c.*(1 + b.*b. + 28*b.*d.) +
                           d.*d.*(12 + 48*b.*d. + 141*c.*c. + 225*d.*d.)) - kurtosis
            eq <- c(eq1,eq2,eq3)
            sum(eq*eq) ## SS
        }

        out <- nlminb(start=c(1,0,0), objective=system.function,
                      scale=10,
                      control=list(trace=0),
                      skewness=skewness, kurtosis=kurtosis)
        if(out$convergence != 0) warning("no convergence")
        b. <- out$par[1L]; c. <- out$par[2L]; d. <- out$par[3L]; a. <- -c.
        if(out$objective > 10^{-5}) {
            warning("!!!Not valid combination of skew/kurtosis")
        }

        return(c(a.,b.,c.,d.))
    }
```

```
    getICOV <- function(b1, c1, d1, b2, c2, d2, R) {
        objectiveFunction <- function(x, b1, c1, d1, b2, c2, d2, R) {
            rho=x[1L]
            eq <- rho*(b1*b2 + 3*b1*d2 + 3*d1*b2 + 9*d1*d2) +
                rho*rho*(2*c1*c2) + rho*rho*rho*(6*d1*d2) - R
            eq*eq
        }


        out <- nlminb(start=R, objective=objectiveFunction,
                      scale=10, control=list(trace=0),
                      b1=b1, c1=c1, d1=d1, b2=b2, c2=c2, d2=d2, R=R)
        if(out$convergence != 0) warning("no convergence")
        rho <- out$par[1L]
        rho
    }

    # create Fleishman table
    FTable <- matrix(0, nvar, 4L)
    for(i in 1:nvar) {
        FTable[i,] <- fleishman1978_abcd(skewness=SK[i], kurtosis=KU[i])
    }
    #
    ICOR <- diag(nvar)
    for(j in 1:(nvar-1L)) {
        for(i in (j+1):nvar) {
            if(COR[i,j] == 0) next
            ICOR[i,j] <- ICOR[j,i] <-
                getICOV(FTable[i,2], FTable[i,3], FTable[i,4],
                        FTable[j,2], FTable[j,3], FTable[j,4], R=COR[i,j])
        }
    }
    list(FTable*sqrt(diag(sigma0)), ICOR)
}

get.fleishman <- function(skewness, kurtosis){
  system.function <- function(x, skewness, kurtosis) {
    b = x[1L]
    c = x[2L]
    d = x[3L]
    eq1 <- b * b + 6 * b * d + 2 * c * c + 15 * d * d - 1
    eq2 <- 2 * c * (b * b + 24 * b * d + 105 * d * d + 2) -
      skewness
    eq3 <- 24 * (b * d + c * c * (1 + b * b + 28 * b * d) +
                   d * d * (12 + 48 * b * d + 141 * c * c + 225 * d *
                              d)) - kurtosis
    eq <- c(eq1, eq2, eq3)
    sum(eq * eq)
  }
  out <- nlminb(start = c(1, 0, 0), objective = system.function,
scale = 10, control = list(trace = ifelse(verbose,1,0),rel.tol = 1e-10),
                skewness = skewness, kurtosis = kurtosis)
  if (out$convergence != 0 || out$objective > 1e-05)
```

```
    warning("no convergence")
  b <- out$par[1L]
  c <- out$par[2L]
  d <- out$par[3L]
  a <- -c
  return(c(a,b,c,d))
}
```

2 Flora and Curran (2004)

Flora and Curran (2004) is a well-cited (currently more than 2000 citations according to Google Scholar) and extensive simulation study. The abstract states:

Confirmatory factor analysis (CFA) is widely used for examining hypothesized relations among ordinal variables (e.g., Likert-type items). A theoretically appropriate method fits the CFA model to polychoric correlations using either weighted least squares (WLS) or robust WLS. Importantly, this approach assumes that a continuous, normal latent process determines each observed variable. The extent to which violations of this assumption undermine CFA estimation is not well-known. In this article, the authors empirically study this issue using a computer simulation study. The results suggest that estimation of polychoric correlations is robust to modest violations of underlying normality. Further, WLS performed adequately only at the largest sample size but led to substantial estimation difficulties with smaller samples. Finally, robust WLS performed well across all conditions.

The authors compare ADF (WLS) with DWLS (which they refer to as robust WLS). In the abstract we see that the bias of the polychoric correlations are a main focus of this article. The abstract also reveals a main finding, that the bias is not too bad under modest violations of non-normality. In the literature review of this paper we read:

To our knowledge, Quiroga (1992) represents the only simulation study that has empirically evaluated the accuracy of polychoric correlations under violations of the latent normality assumption. Quiroga manipulated the skewness and kurtosis of two continuous variables to examine the effects of nonnormality on polychoric cor- relation estimates between two variables, each with four observed ordinal categories. The polychoric correlation values consistently overestimated the true correlation between the nonnormal latent response variables. However, the extent of the overestimation was small, with bias typically less than 2% of the true correlation.

Flora and Curran has the bias of the polychoric correlation as one of three main focuses, and they hypothesize, based on the PhD work of Quiroga, that *we predicted that the estimates of polychoric correlations would be relatively unbiased as a function of minor to modest violations of normality of the latent response distributions.*

2.1 Data generation in Flora and Curran (2004)

The authors used the Vale-Marelli technique to generate continous non-normal data prior to discretization. It is important to note that the Fleishman polynomial in three of these conditions was monotonous. This means that the multivariate non-normal vector has a normal copula, so that this vector has a distribution that closely resembles multivariate normality (Foldnes and Grønneberg 2015) which was clearly not the intention. In fact, it can be shown ("A Problem with Discretizing Vale-Maurelli in Simulation Studies" 2019) that in such cases the VM approach is ineffectual: The discretized data is numerically identical to data obtained by discretizing a multivariate normal vector. So the polychoric correlations will be identical to the correlations of this normal vector.

The non-normal condititions are as follows (See Table 1 in Flora and Curran 2004)

- Low skewness vs. low kurtosis. s=0.75, k=1.75. Fleishman polynomial is monotonous, and hence equivalent to multivariate normal data.
- Low skewness vs. moderate kurtosis. s=0.75, k=3.75. Monotonous: equivalent to multivariate normal data.
- Moderate skewness vs. low kurtosis. s=1.25, k=1.75. Non-Monotonous: Truly non-normal.
- Moderate skewness vs. moderate kurtosis. s=1.25, k=3.75. Monotonous: equivalent to multivariate normal data.

The Vale-Maurelli approach works by first estimating the Fleishman polynomial, and then working out the intermediate correlation matrix ICOR of the multivariate Z-vector that is used to generate the non-normal data. It can be shown ("A Problem with Discretizing Vale-Maurelli in Simulation Studies" 2019) that under monotonous Fleishman polynomials, the estimated polychoric correlations will converge toward a population matrix that equals the intermediate matrix ICOR. This matrix does not equal the target covariance matrix that is given as input for the VM transform.

In the next section we investigate the models employed by Flora and Curran.

2.2 Models and polychorics in Flora and Curran (2004)

```
loading <- 0.7; uniq <- 1-loading^2
m1 ='F1 =~ start(loading)*x1+start(loading)*x2+start(loading)*x3+start(loading)*x4+start(loading)*x5;
      x1~~start(uniq)*x1;x2~~start(uniq)*x2;x3~~start(uniq)*x3;x4~~start(uniq)*x4;x5~~start(uniq)*x5;'
m1 <- str_replace_all(m1, "loading", as.character(loading))
m1 <- str_replace_all(m1, "uniq", as.character(1-loading^2))
fit = sem(m1, data=NULL)
sigma0 = inspect(fit, "sigma.hat")## correlation matrix.
```

Model 1 has one factor with five indicators. Factor loadings are 0.7 and residual variances 0.51 so that the resulting target covariance matrix is a correlation matrix, with all correlations at 0.49:

```
sigma0
```

```
##    x1   x2   x3   x4   x5
## x1 1.00
## x2 0.49 1.00
## x3 0.49 0.49 1.00
## x4 0.49 0.49 0.49 1.00
## x5 0.49 0.49 0.49 0.49 1.00
```

Let us now calculate the intermediate correlation matrix ICOR used in the VM transform in the three conditions of low skewness vs. low kurtosis, low skewness vs. moderate kurtosis, and moderate skewness vs. moderate kurtosis, respectively:

```
s=0.75; k=1.75
is.monotone(getVM(s,k, sigma0)[[1]][1, ])
```

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

```
ICOR <- getVM(s,k, sigma0)[[2]]; colnames(ICOR) <- colnames(sigma0)
ICOR
```

```
##             x1        x2        x3        x4        x5
## [1,] 1.0000000 0.4978064 0.4978064 0.4978064 0.4978064
## [2,] 0.4978064 1.0000000 0.4978064 0.4978064 0.4978064
## [3,] 0.4978064 0.4978064 1.0000000 0.4978064 0.4978064
## [4,] 0.4978064 0.4978064 0.4978064 1.0000000 0.4978064
## [5,] 0.4978064 0.4978064 0.4978064 0.4978064 1.0000000
```

```
s=0.75; k=3.75
is.monotone(getVM(s,k, sigma0)[[1]][1, ])
```

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

```
ICOR <- getVM(s,k, sigma0)[[2]]; colnames(ICOR) <- colnames(sigma0)
ICOR
```

```
##             x1        x2        x3        x4        x5
## [1,] 1.0000000 0.5045185 0.5045185 0.5045185 0.5045185
## [2,] 0.5045185 1.0000000 0.5045185 0.5045185 0.5045185
## [3,] 0.5045185 0.5045185 1.0000000 0.5045185 0.5045185
## [4,] 0.5045185 0.5045185 0.5045185 1.0000000 0.5045185
## [5,] 0.5045185 0.5045185 0.5045185 0.5045185 1.0000000
```

```
s=1.25; k=3.75
is.monotone(getVM(s,k, sigma0)[[1]][1, ])
```

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

```
ICOR <- getVM(s,k, sigma0)[[2]]; colnames(ICOR) <- colnames(sigma0)
ICOR
```

```
##             x1        x2        x3        x4        x5
## [1,] 1.0000000 0.5083669 0.5083669 0.5083669 0.5083669
## [2,] 0.5083669 1.0000000 0.5083669 0.5083669 0.5083669
## [3,] 0.5083669 0.5083669 1.0000000 0.5083669 0.5083669
## [4,] 0.5083669 0.5083669 0.5083669 1.0000000 0.5083669
## [5,] 0.5083669 0.5083669 0.5083669 0.5083669 1.0000000
```

We see that the population values of the polychoric correlations are not far from the target correlation of 0.49. The values here reported are in accord with the estimated values reported in Table 2 of Flora and Curran (2004). Even in the condition with moderate skewness and kurtosis, the data that is discretized is very close to the data generated by the population model. Moreover, **the model fits perfectly to each of the polychoric correlation matrices above!** This means that Flora and Curran (2004) in reality simulated multivariate normal data that fits perfectly to the model. If we fit the last (moderate skew and kurtosis) polychoric correlation matrix to the proposed one-factor model (identified by setting the factor variance to one) with DWLS we get the following parameter estimates:

```
f <- sem(m1, sample.cov=ICOR, sample.nobs=500, std.lv=TRUE)
parameterestimates(f)[, 1:4]
```

```
##    lhs op rhs   est
## 1   F1 =~  x1 0.712
## 2   F1 =~  x2 0.712
## 3   F1 =~  x3 0.712
## 4   F1 =~  x4 0.712
## 5   F1 =~  x5 0.712
## 6   x1 ~~  x1 0.491
## 7   x2 ~~  x2 0.491
## 8   x3 ~~  x3 0.491
## 9   x4 ~~  x4 0.491
## 10  x5 ~~  x5 0.491
## 11  F1 ~~  F1 1.000
```

So the loadings are slightly off the data-generation values of 0.7, but the model fits perfectly.

Let us do the same for Model 3 in Flora and Curran (2004). In fact, all four models in their paper has the same property that the model fits perfectly to the population value of the polychoric correlation matrix, so

that the authors in reality have simulated from a multivarate normal vector for which the model is correctly specified. Model 3 has two factors, each with five indicators, and similar loadings and unique variances as for Model 1. In addition the two factors correlated with 0.3. The target covariance matrix is

```
loading <- 0.7; uniq <- 1-loading^2
m3 ='F1 =~ start(loading)*x1+start(loading)*x2+start(loading)*x3+start(loading)*x4+start(loading)*x5;
     F2 =~ start(loading)*x6+start(loading)*x7+start(loading)*x8+start(loading)*x9+start(loading)*x10;
x1~~start(uniq)*x1;x2~~start(uniq)*x2;x3~~start(uniq)*x3;x4~~start(uniq)*x4;x5~~start(uniq)*x5;
x6~~start(uniq)*x6;x7~~start(uniq)*x7;x8~~start(uniq)*x8;x9~~start(uniq)*x9;x10~~start(uniq)*x10;
F1 ~~start(0.3)*F2'
m3 <- str_replace_all(m3, "loading", as.character(loading))
m3 <- str_replace_all(m3, "uniq", as.character(1-loading^2))
fit = sem(m3, data=NULL, std.lv=T)
sigma0 = inspect(fit, "sigma.hat")## correlation matrix.
sigma0
```

```
##     x1    x2    x3    x4    x5    x6    x7    x8    x9    x10
## x1  1.000
## x2  0.490 1.000
## x3  0.490 0.490 1.000
## x4  0.490 0.490 0.490 1.000
## x5  0.490 0.490 0.490 0.490 1.000
## x6  0.147 0.147 0.147 0.147 0.147 1.000
## x7  0.147 0.147 0.147 0.147 0.147 0.490 1.000
## x8  0.147 0.147 0.147 0.147 0.147 0.490 0.490 1.000
## x9  0.147 0.147 0.147 0.147 0.147 0.490 0.490 0.490 1.000
## x10 0.147 0.147 0.147 0.147 0.147 0.490 0.490 0.490 0.490 1.000
```

Consider next the three conditions of low skewness vs. low kurtosis, low skewness vs. moderate kurtosis, and moderate skewness vs. moderate kurtosis. We list the intermediate correlations (the polychoric correlations that are estimated) and the parameter estimates when fitting the model to these matrices. First out is the low skew and low kurtosis condition:

```
s=0.75; k=1.75
is.monotone(getVM(s,k, sigma0)[[1]][1, ])
```

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

```
ICOR <- getVM(s,k, sigma0)[[2]]; colnames(ICOR) <- colnames(sigma0)
f<-sem(m3, sample.cov=ICOR, sample.nobs=500, std.lv=T)#perfect fit!
round(ICOR,3)
```

```
##          x1    x2    x3    x4    x5    x6    x7    x8    x9   x10
##  [1,] 1.000 0.498 0.498 0.498 0.498 0.151 0.151 0.151 0.151 0.151
##  [2,] 0.498 1.000 0.498 0.498 0.498 0.151 0.151 0.151 0.151 0.151
##  [3,] 0.498 0.498 1.000 0.498 0.498 0.151 0.151 0.151 0.151 0.151
##  [4,] 0.498 0.498 0.498 1.000 0.498 0.151 0.151 0.151 0.151 0.151
##  [5,] 0.498 0.498 0.498 0.498 1.000 0.151 0.151 0.151 0.151 0.151
##  [6,] 0.151 0.151 0.151 0.151 0.151 1.000 0.498 0.498 0.498 0.498
##  [7,] 0.151 0.151 0.151 0.151 0.151 0.498 1.000 0.498 0.498 0.498
##  [8,] 0.151 0.151 0.151 0.151 0.151 0.498 0.498 1.000 0.498 0.498
##  [9,] 0.151 0.151 0.151 0.151 0.151 0.498 0.498 0.498 1.000 0.498
## [10,] 0.151 0.151 0.151 0.151 0.151 0.498 0.498 0.498 0.498 1.000
```

```
parameterestimates(f)[, 1:4]
```

```
##    lhs op rhs   est
```

```
## 1   F1 =~  x1 0.705
## 2   F1 =~  x2 0.705
## 3   F1 =~  x3 0.705
## 4   F1 =~  x4 0.705
## 5   F1 =~  x5 0.705
## 6   F2 =~  x6 0.705
## 7   F2 =~  x7 0.705
## 8   F2 =~  x8 0.705
## 9   F2 =~  x9 0.705
## 10  F2 =~ x10 0.705
## 11  x1 ~~  x1 0.501
## 12  x2 ~~  x2 0.501
## 13  x3 ~~  x3 0.501
## 14  x4 ~~  x4 0.501
## 15  x5 ~~  x5 0.501
## 16  x6 ~~  x6 0.501
## 17  x7 ~~  x7 0.501
## 18  x8 ~~  x8 0.501
## 19  x9 ~~  x9 0.501
## 20 x10 ~~ x10 0.501
## 21  F1 ~~  F2 0.303
## 22  F1 ~~  F1 1.000
## 23  F2 ~~  F2 1.000
```

Then consider the low skew, moderate kurtosis condition:

```
s=0.75; k=3.75
is.monotone(getVM(s,k, sigma0)[[1]][1, ])
```

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

```
ICOR <- getVM(s,k, sigma0)[[2]]; colnames(ICOR) <- colnames(sigma0)
f<-sem(m3, sample.cov=ICOR, sample.nobs=500, std.lv=T)#perfect fit!
round(ICOR,3)
```

```
##          x1    x2    x3    x4    x5    x6    x7    x8    x9   x10
##  [1,] 1.000 0.505 0.505 0.505 0.505 0.153 0.153 0.153 0.153 0.153
##  [2,] 0.505 1.000 0.505 0.505 0.505 0.153 0.153 0.153 0.153 0.153
##  [3,] 0.505 0.505 1.000 0.505 0.505 0.153 0.153 0.153 0.153 0.153
##  [4,] 0.505 0.505 0.505 1.000 0.505 0.153 0.153 0.153 0.153 0.153
##  [5,] 0.505 0.505 0.505 0.505 1.000 0.153 0.153 0.153 0.153 0.153
##  [6,] 0.153 0.153 0.153 0.153 0.153 1.000 0.505 0.505 0.505 0.505
##  [7,] 0.153 0.153 0.153 0.153 0.153 0.505 1.000 0.505 0.505 0.505
##  [8,] 0.153 0.153 0.153 0.153 0.153 0.505 0.505 1.000 0.505 0.505
##  [9,] 0.153 0.153 0.153 0.153 0.153 0.505 0.505 0.505 1.000 0.505
## [10,] 0.153 0.153 0.153 0.153 0.153 0.505 0.505 0.505 0.505 1.000
```

```
parameterestimates(f)[, 1:4]
```

```
##    lhs op rhs   est
## 1   F1 =~  x1 0.710
## 2   F1 =~  x2 0.710
## 3   F1 =~  x3 0.710
## 4   F1 =~  x4 0.710
## 5   F1 =~  x5 0.710
## 6   F2 =~  x6 0.710
## 7   F2 =~  x7 0.710
```

```
## 8   F2 =~  x8 0.710
## 9   F2 =~  x9 0.710
## 10  F2 =~ x10 0.710
## 11  x1 ~~  x1 0.494
## 12  x2 ~~  x2 0.494
## 13  x3 ~~  x3 0.494
## 14  x4 ~~  x4 0.494
## 15  x5 ~~  x5 0.494
## 16  x6 ~~  x6 0.494
## 17  x7 ~~  x7 0.494
## 18  x8 ~~  x8 0.494
## 19  x9 ~~  x9 0.494
## 20 x10 ~~ x10 0.494
## 21  F1 ~~  F2 0.304
## 22  F1 ~~  F1 1.000
## 23  F2 ~~  F2 1.000
```

And finally the moderate skew and moderate kurtosis condition

```
s=1.25; k=3.75
is.monotone(getVM(s,k, sigma0)[[1]][1, ])
```

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

```
ICOR <- getVM(s,k, sigma0)[[2]]; colnames(ICOR) <- colnames(sigma0)
f<-sem(m3, sample.cov=ICOR, sample.nobs=500, std.lv=T)#perfect fit!
round(ICOR,3)
```

```
##          x1    x2    x3    x4    x5    x6    x7    x8    x9   x10
##  [1,] 1.000 0.508 0.508 0.508 0.508 0.156 0.156 0.156 0.156 0.156
##  [2,] 0.508 1.000 0.508 0.508 0.508 0.156 0.156 0.156 0.156 0.156
##  [3,] 0.508 0.508 1.000 0.508 0.508 0.156 0.156 0.156 0.156 0.156
##  [4,] 0.508 0.508 0.508 1.000 0.508 0.156 0.156 0.156 0.156 0.156
##  [5,] 0.508 0.508 0.508 0.508 1.000 0.156 0.156 0.156 0.156 0.156
##  [6,] 0.156 0.156 0.156 0.156 0.156 1.000 0.508 0.508 0.508 0.508
##  [7,] 0.156 0.156 0.156 0.156 0.156 0.508 1.000 0.508 0.508 0.508
##  [8,] 0.156 0.156 0.156 0.156 0.156 0.508 0.508 1.000 0.508 0.508
##  [9,] 0.156 0.156 0.156 0.156 0.156 0.508 0.508 0.508 1.000 0.508
## [10,] 0.156 0.156 0.156 0.156 0.156 0.508 0.508 0.508 0.508 1.000
```

```
parameterestimates(f)[, 1:4]
```

```
##    lhs op rhs   est
## 1   F1 =~  x1 0.712
## 2   F1 =~  x2 0.712
## 3   F1 =~  x3 0.712
## 4   F1 =~  x4 0.712
## 5   F1 =~  x5 0.712
## 6   F2 =~  x6 0.712
## 7   F2 =~  x7 0.712
## 8   F2 =~  x8 0.712
## 9   F2 =~  x9 0.712
## 10  F2 =~ x10 0.712
## 11  x1 ~~  x1 0.491
## 12  x2 ~~  x2 0.491
## 13  x3 ~~  x3 0.491
## 14  x4 ~~  x4 0.491
```

```
## 15  x5 ~~  x5 0.491
## 16  x6 ~~  x6 0.491
## 17  x7 ~~  x7 0.491
## 18  x8 ~~  x8 0.491
## 19  x9 ~~  x9 0.491
## 20 x10 ~~ x10 0.491
## 21  F1 ~~  F2 0.307
## 22  F1 ~~  F1 1.000
## 23  F2 ~~  F2 1.000
```

The population parameter values in these tables are higher than the values specified in the data-generation process, for both factor loadings and the intrafactor correlations. This agrees with the result section in Flora and Curran (2004), whose parameter estimates are summarised as follows: *On average, the parameters are overestimated across all conditions, and this positive bias increases with increasing model size. … In general, the bias in parameter estimates obtained with robust WLS is consistently trivial*

The reason is that the ICOR is very close to target covariance matrix, so that the parameters shift only to a small degree. Several studies have also reported that the interfactor correlation is overestimated, which has been reported by many studies, see e.g. Li (2016) Here is the explanation, it is caused simply by the simulation method, and is not inherent to the estimators of polychorics or other deeper reasons.

3 Rhemtulla, Brosseau-Liard, and Savalei (2012)

This paper currently has more than 760 citations in Google Scholar, and the abstract goes:

A simulation study compared the performance of robust normal theory maximum likelihood (ML) and robust categorical least squares (cat-LS) methodology for estimating confirmatory factor analysis models with ordinal variables. Data were generated from 2 models with 2–7 categories, 4 sample sizes, 2 latent distributions, and 5 patterns of category thresholds. Results revealed that factor loadings and robust standard errors were generally most accurately estimated using cat-LS, especially with fewer than 5 categories; however, factor correlations and model fit were assessed equally well with ML. Cat-LS was found to be more sensitive to sample size and to violations of the assumption of normality of the underlying continuous variables. Normal theory ML was found to be more sensitive to asymmetric category thresholds and was especially biased when estimating large factor loadings. Accordingly, we recommend cat-LS for data sets containing variables with fewer than 5 categories and ML when there are 5 or more categories, sample size is small, and category thresholds are approximately symmetric. With 6–7 categories, results were similar across methods for many conditions; in these cases, either method is acceptable.

3.1 Data generation in Rhemtulla, Brosseau-Liard, and Savalei (2012)

This paper has one normal condition and one non-normal condition, with skewness 2 and kurtosis 7. The authors state that this is more severe non-normality than used in Flora and Curran (2004), however, again the Fleishman polynomial is monotonous, so this is equivalent to discretizing a multivariate normal vector.

3.2 Models and polychorics in Rhemtulla, Brosseau-Liard, and Savalei (2012)

Model 1 was a two-factor CFA model with five indicators per factor, factor loadings for each factor were 0.3, 0.4, 0.5, 0.6, 0.7, so that the target covariance matrix is

```
m1 = 'F1 =~ start(0.3)*x1+start(0.4)*x2+start(0.5)*x3+start(0.6)*x4+start(0.7)*x5
      F2 =~start(0.3)*x6+start(0.4)*x7+start(0.5)*x8+start(0.6)*x9+start(0.7)*x10
      F1 ~~ start(1)*F1; F2 ~~ start(1)*F2; F1 ~~ start(.3)*F2
      x1~~start(1-.3^2)*x1; x2~~start(1-.4^2)*x2;x3~~start(1-.5^2)*x3;
```

```
      x4~~start(1-.6^2)*x4; x5~~start(1-.7^2)*x5
      x6~~start(1-.3^2)*x6; x7~~start(1-.4^2)*x7;x8~~start(1-.5^2)*x8;
      x9~~start(1-.6^2)*x9; x10~~start(1-.7^2)*x10
'
fit = sem(m1, data=NULL)
sigma0 = inspect(fit, "sigma.hat")## correlation matrix.
sigma0
```

```
##     x1    x2    x3    x4    x5    x6    x7    x8    x9    x10
## x1  1.000
## x2  0.120 1.000
## x3  0.150 0.200 1.000
## x4  0.180 0.240 0.300 1.000
## x5  0.210 0.280 0.350 0.420 1.000
## x6  0.027 0.036 0.045 0.054 0.063 1.000
## x7  0.036 0.048 0.060 0.072 0.084 0.120 1.000
## x8  0.045 0.060 0.075 0.090 0.105 0.150 0.200 1.000
## x9  0.054 0.072 0.090 0.108 0.126 0.180 0.240 0.300 1.000
## x10 0.063 0.084 0.105 0.126 0.147 0.210 0.280 0.350 0.420 1.000
```

We can compare this with the polychoric correlation matrix, namely the intermediate correlation matrix ICOR:

```
library(xtable)
s=2; k=7
is.monotone(getVM(s,k, sigma0)[[1]][1, ])
```

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

```
ICOR <- getVM(s,k, sigma0)[[2]]; colnames(ICOR) <- colnames(sigma0)
round(ICOR, 3)
```

```
##          x1    x2    x3    x4    x5    x6    x7    x8    x9   x10
##  [1,] 1.000 0.138 0.172 0.205 0.238 0.032 0.042 0.053 0.063 0.073
##  [2,] 0.138 1.000 0.227 0.271 0.314 0.042 0.056 0.070 0.084 0.098
##  [3,] 0.172 0.227 1.000 0.335 0.388 0.053 0.070 0.087 0.104 0.121
##  [4,] 0.205 0.271 0.335 1.000 0.460 0.063 0.084 0.104 0.125 0.145
##  [5,] 0.238 0.314 0.388 0.460 1.000 0.073 0.098 0.121 0.145 0.169
##  [6,] 0.032 0.042 0.053 0.063 0.073 1.000 0.138 0.172 0.205 0.238
##  [7,] 0.042 0.056 0.070 0.084 0.098 0.138 1.000 0.227 0.271 0.314
##  [8,] 0.053 0.070 0.087 0.104 0.121 0.172 0.227 1.000 0.335 0.388
##  [9,] 0.063 0.084 0.104 0.125 0.145 0.205 0.271 0.335 1.000 0.460
## [10,] 0.073 0.098 0.121 0.145 0.169 0.238 0.314 0.388 0.460 1.000
```

We see that all the polychorics are larger than their corresponding target correlations.

In contrast to Flora and Curran (2004), the model no longer fits the polychoric correlations exactly. When we fit the model with the estimator used by Rhemtulla, Brosseau-Liard, and Savalei (2012), namely CAT-LS (ULS) we get the following estimates

```
par = parameterestimates(sem(m1, sample.cov=ICOR, estimator="ULS",sample.nobs=500, std.lv=T))[1:4]
par
```

```
##    lhs op rhs   est
## 1   F1 =~  x1 0.325
## 2   F1 =~  x2 0.429
## 3   F1 =~  x3 0.530
```

```
## 4   F1 =~  x4 0.631
## 5   F1 =~  x5 0.731
## 6   F2 =~  x6 0.325
## 7   F2 =~  x7 0.429
## 8   F2 =~  x8 0.530
## 9   F2 =~  x9 0.631
## 10  F2 =~ x10 0.731
## 11  F1 ~~  F1 1.000
## 12  F2 ~~  F2 1.000
## 13  F1 ~~  F2 0.312
## 14  x1 ~~  x1 0.895
## 15  x2 ~~  x2 0.816
## 16  x3 ~~  x3 0.719
## 17  x4 ~~  x4 0.602
## 18  x5 ~~  x5 0.465
## 19  x6 ~~  x6 0.895
## 20  x7 ~~  x7 0.816
## 21  x8 ~~  x8 0.719
## 22  x9 ~~  x9 0.602
## 23 x10 ~~ x10 0.465
```

We see that all factor loading estimates and also the factor correlation estimate will be positively biased with respect to the values used to generate the target covariance matrix. For instance, we will expect, regardless of number of categories and values of thresholds, a value of 0.3246 for the factor loadings that were set to 0.3 in the design. This is what Figure 4 in Rhemtulla, Brosseau-Liard, and Savalei (2012) is showing! That figure seems to aggregate results for both Model 1 and Model 2, so let us see what happens with Model 2, which has ten indicators for each factor. We use the same factor loadings as for Model 1, that is, F1 loads with $0.3, 0.4, 0.5, 0.6, 0.7, 0.3, 0.4, 0.5, 0.6, 0.7$. The model-implied target covariance is:

```
m1 = "
F1 =~ start(0.3)*x1+start(0.4)*x2+start(0.5)*x3+start(0.6)*x4+start(0.7)*x5+
      start(0.3)*x6+start(0.4)*x7+start(0.5)*x8+start(0.6)*x9+start(0.7)*x10;
F2 =~ start(0.3)*x11+start(0.4)*x12+start(0.5)*x13+start(0.6)*x14+start(0.7)*x15+
      start(0.3)*x16+start(0.4)*x17+start(0.5)*x18+start(0.6)*x19+start(0.7)*x20;
F1 ~~ start(1)*F1; F2 ~~ start(1)*F2; F1 ~~ start(.3)*F2;

x1~~start(1-.3^2)*x1; x2~~start(1-.4^2)*x2; x3~~start(1-.5^2)*x3;
x4~~start(1-.6^2)*x4; x5~~start(1-.7^2)*x5; x6~~start(1-.3^2)*x6;
x7~~start(1-.4^2)*x7;x8~~start(1-.5^2)*x8; x9~~start(1-.6^2)*x9;
x10~~start(1-.7^2)*x10; x11~~start(1-.3^2)*x11; x12~~start(1-.4^2)*x12;
x13~~start(1-.5^2)*x13; x14~~start(1-.6^2)*x14; x15~~start(1-.7^2)*x15;
x16~~start(1-.3^2)*x16; x17~~start(1-.4^2)*x17; x18~~start(1-.5^2)*x18;
x19~~start(1-.6^2)*x19; x20~~start(1-.7^2)*x20
"
fit = sem(m1, data=NULL)
sigma0 = inspect(fit, "sigma.hat")## correlation matrix.
sigma0
```

```
##     x1    x2    x3    x4    x5    x6    x7    x8    x9    x10   x11
## x1  1.000
## x2  0.120 1.000
## x3  0.150 0.200 1.000
## x4  0.180 0.240 0.300 1.000
## x5  0.210 0.280 0.350 0.420 1.000
## x6  0.090 0.120 0.150 0.180 0.210 1.000
```

```
## x7  0.120 0.160 0.200 0.240 0.280 0.120 1.000
## x8  0.150 0.200 0.250 0.300 0.350 0.150 0.200 1.000
## x9  0.180 0.240 0.300 0.360 0.420 0.180 0.240 0.300 1.000
## x10 0.210 0.280 0.350 0.420 0.490 0.210 0.280 0.350 0.420 1.000
## x11 0.027 0.036 0.045 0.054 0.063 0.027 0.036 0.045 0.054 0.063 1.000
## x12 0.036 0.048 0.060 0.072 0.084 0.036 0.048 0.060 0.072 0.084 0.120
## x13 0.045 0.060 0.075 0.090 0.105 0.045 0.060 0.075 0.090 0.105 0.150
## x14 0.054 0.072 0.090 0.108 0.126 0.054 0.072 0.090 0.108 0.126 0.180
## x15 0.063 0.084 0.105 0.126 0.147 0.063 0.084 0.105 0.126 0.147 0.210
## x16 0.027 0.036 0.045 0.054 0.063 0.027 0.036 0.045 0.054 0.063 0.090
## x17 0.036 0.048 0.060 0.072 0.084 0.036 0.048 0.060 0.072 0.084 0.120
## x18 0.045 0.060 0.075 0.090 0.105 0.045 0.060 0.075 0.090 0.105 0.150
## x19 0.054 0.072 0.090 0.108 0.126 0.054 0.072 0.090 0.108 0.126 0.180
## x20 0.063 0.084 0.105 0.126 0.147 0.063 0.084 0.105 0.126 0.147 0.210
##     x12   x13   x14   x15   x16   x17   x18   x19   x20
## x1
## x2
## x3
## x4
## x5
## x6
## x7
## x8
## x9
## x10
## x11
## x12 1.000
## x13 0.200 1.000
## x14 0.240 0.300 1.000
## x15 0.280 0.350 0.420 1.000
## x16 0.120 0.150 0.180 0.210 1.000
## x17 0.160 0.200 0.240 0.280 0.120 1.000
## x18 0.200 0.250 0.300 0.350 0.150 0.200 1.000
## x19 0.240 0.300 0.360 0.420 0.180 0.240 0.300 1.000
## x20 0.280 0.350 0.420 0.490 0.210 0.280 0.350 0.420 1.000
```

While the polychoric correlation matrix and the parameter estimates are

```
library(xtable)
s=2; k=7
ICOR <- getVM(s,k, sigma0)[[2]]; colnames(ICOR) <- colnames(sigma0)
round(ICOR, 3)
```

```
##          x1    x2    x3    x4    x5    x6    x7    x8    x9   x10   x11
##  [1,] 1.000 0.138 0.172 0.205 0.238 0.104 0.138 0.172 0.205 0.238 0.032
##  [2,] 0.138 1.000 0.227 0.271 0.314 0.138 0.183 0.227 0.271 0.314 0.042
##  [3,] 0.172 0.227 1.000 0.335 0.388 0.172 0.227 0.282 0.335 0.388 0.053
##  [4,] 0.205 0.271 0.335 1.000 0.460 0.205 0.271 0.335 0.398 0.460 0.063
##  [5,] 0.238 0.314 0.388 0.460 1.000 0.238 0.314 0.388 0.460 0.530 0.073
##  [6,] 0.104 0.138 0.172 0.205 0.238 1.000 0.138 0.172 0.205 0.238 0.032
##  [7,] 0.138 0.183 0.227 0.271 0.314 0.138 1.000 0.227 0.271 0.314 0.042
##  [8,] 0.172 0.227 0.282 0.335 0.388 0.172 0.227 1.000 0.335 0.388 0.053
##  [9,] 0.205 0.271 0.335 0.398 0.460 0.205 0.271 0.335 1.000 0.460 0.063
## [10,] 0.238 0.314 0.388 0.460 0.530 0.238 0.314 0.388 0.460 1.000 0.073
## [11,] 0.032 0.042 0.053 0.063 0.073 0.032 0.042 0.053 0.063 0.073 1.000
```

```
## [12,] 0.042 0.056 0.070 0.084 0.098 0.042 0.056 0.070 0.084 0.098 0.138
## [13,] 0.053 0.070 0.087 0.104 0.121 0.053 0.070 0.087 0.104 0.121 0.172
## [14,] 0.063 0.084 0.104 0.125 0.145 0.063 0.084 0.104 0.125 0.145 0.205
## [15,] 0.073 0.098 0.121 0.145 0.169 0.073 0.098 0.121 0.145 0.169 0.238
## [16,] 0.032 0.042 0.053 0.063 0.073 0.032 0.042 0.053 0.063 0.073 0.104
## [17,] 0.042 0.056 0.070 0.084 0.098 0.042 0.056 0.070 0.084 0.098 0.138
## [18,] 0.053 0.070 0.087 0.104 0.121 0.053 0.070 0.087 0.104 0.121 0.172
## [19,] 0.063 0.084 0.104 0.125 0.145 0.063 0.084 0.104 0.125 0.145 0.205
## [20,] 0.073 0.098 0.121 0.145 0.169 0.073 0.098 0.121 0.145 0.169 0.238
##         x12   x13   x14   x15   x16   x17   x18   x19   x20
##  [1,] 0.042 0.053 0.063 0.073 0.032 0.042 0.053 0.063 0.073
##  [2,] 0.056 0.070 0.084 0.098 0.042 0.056 0.070 0.084 0.098
##  [3,] 0.070 0.087 0.104 0.121 0.053 0.070 0.087 0.104 0.121
##  [4,] 0.084 0.104 0.125 0.145 0.063 0.084 0.104 0.125 0.145
##  [5,] 0.098 0.121 0.145 0.169 0.073 0.098 0.121 0.145 0.169
##  [6,] 0.042 0.053 0.063 0.073 0.032 0.042 0.053 0.063 0.073
##  [7,] 0.056 0.070 0.084 0.098 0.042 0.056 0.070 0.084 0.098
##  [8,] 0.070 0.087 0.104 0.121 0.053 0.070 0.087 0.104 0.121
##  [9,] 0.084 0.104 0.125 0.145 0.063 0.084 0.104 0.125 0.145
## [10,] 0.098 0.121 0.145 0.169 0.073 0.098 0.121 0.145 0.169
## [11,] 0.138 0.172 0.205 0.238 0.104 0.138 0.172 0.205 0.238
## [12,] 1.000 0.227 0.271 0.314 0.138 0.183 0.227 0.271 0.314
## [13,] 0.227 1.000 0.335 0.388 0.172 0.227 0.282 0.335 0.388
## [14,] 0.271 0.335 1.000 0.460 0.205 0.271 0.335 0.398 0.460
## [15,] 0.314 0.388 0.460 1.000 0.238 0.314 0.388 0.460 0.530
## [16,] 0.138 0.172 0.205 0.238 1.000 0.138 0.172 0.205 0.238
## [17,] 0.183 0.227 0.271 0.314 0.138 1.000 0.227 0.271 0.314
## [18,] 0.227 0.282 0.335 0.388 0.172 0.227 1.000 0.335 0.388
## [19,] 0.271 0.335 0.398 0.460 0.205 0.271 0.335 1.000 0.460
## [20,] 0.314 0.388 0.460 0.530 0.238 0.314 0.388 0.460 1.000
```

And the parameter population values under cat-LS are

```
par = parameterestimates(sem(m1, estimator="ULS",sample.cov=ICOR, sample.nobs=500, std.lv=T))[1:4]
par
```

```
##    lhs op rhs   est
## 1   F1 =~  x1 0.325
## 2   F1 =~  x2 0.429
## 3   F1 =~  x3 0.531
## 4   F1 =~  x4 0.631
## 5   F1 =~  x5 0.730
## 6   F1 =~  x6 0.325
## 7   F1 =~  x7 0.429
## 8   F1 =~  x8 0.531
## 9   F1 =~  x9 0.631
## 10  F1 =~ x10 0.730
## 11  F2 =~ x11 0.325
## 12  F2 =~ x12 0.429
## 13  F2 =~ x13 0.531
## 14  F2 =~ x14 0.631
## 15  F2 =~ x15 0.730
## 16  F2 =~ x16 0.325
## 17  F2 =~ x17 0.429
## 18  F2 =~ x18 0.531
```

```
## 19  F2 =~ x19 0.631
## 20  F2 =~ x20 0.730
## 21  F1 ~~  F1 1.000
## 22  F2 ~~  F2 1.000
## 23  F1 ~~  F2 0.312
## 24  x1 ~~  x1 0.895
## 25  x2 ~~  x2 0.816
## 26  x3 ~~  x3 0.718
## 27  x4 ~~  x4 0.602
## 28  x5 ~~  x5 0.467
## 29  x6 ~~  x6 0.895
## 30  x7 ~~  x7 0.816
## 31  x8 ~~  x8 0.718
## 32  x9 ~~  x9 0.602
## 33 x10 ~~ x10 0.467
## 34 x11 ~~ x11 0.895
## 35 x12 ~~ x12 0.816
## 36 x13 ~~ x13 0.718
## 37 x14 ~~ x14 0.602
## 38 x15 ~~ x15 0.467
## 39 x16 ~~ x16 0.895
## 40 x17 ~~ x17 0.816
## 41 x18 ~~ x18 0.718
## 42 x19 ~~ x19 0.602
## 43 x20 ~~ x20 0.467
```

So the factor loadings for M1 and M2 are identical, and it makes sense to combine the information from these two models, as Rhemtulla, Brosseau-Liard, and Savalei (2012) do.

To sum up, what Rhemtulla, Brosseau-Liard, and Savalei (2012) did to simulate data is numerically equivalent to simulating and discretizing normal data whose covariance matrix is slightly misspecified for the model at hand. The mild specification error means that the factor loadings and intrafactor correlation in the population are only slightly different from the values used in the data-generating set-up. Hence it seems that parameter bias under cat-LS is negligible.

4 Li (2016)

This paper currently is already cited more than 200 times according to Google Scholar. Similar to Rhemtulla, Brosseau-Liard, and Savalei (2012) it focuses on comparing MLR with a categorical-ordinal method, namely cat-DWLS (WLSMV in MPlus terms). The abstract goes

… Robust ML (MLR) has been introduced into CFA models when this normality assumption is slightly or moderately violated. Diagonally weighted least squares (WLSMV), on the other hand, is specifically designed for ordinal data. … A Monte Carlo simulation was carried out to compare the effects of different configurations of latent response distributions, numbers of categories, and sample sizes on model parameter estimates, standard errors, and chi-square test statistics in a correlated two-factor model. The results showed that WLSMV was less biased and more accurate than MLR in estimating the factor loadings across nearly every condition. However, WLSMV yielded moderate overestimation of the interfactor correlations when the sample size was small or/and when the latent distributions were moderately nonnormal. …

4.1 Data generation in Li (2016)

Two non-normal conditions, using VM method, with

- s=0.5 and k=1.5
- s=1.5 and k=3

Only the first condition has a monotonous Fleishman polynomial, and we consider only this condition in the following.

4.2 Model and polychorics in Li (2016)

Only one model, a two-factor model with five indicators per factor, was employed. Factor loadings were set to 0.7 and unique variances to 0.51. It hence turns out that this is exact the same model used in Flora and Curran (2004), therein named Model 3. The target covariance matrix, and the polychoric correlations for the s=0.5, k=1.5 condition are:

```
loading <- 0.7; uniq <- 1-loading^2
m3 ='F1 =~ start(loading)*x1+start(loading)*x2+start(loading)*x3+start(loading)*x4+start(loading)*x5;
     F2 =~ start(loading)*x6+start(loading)*x7+start(loading)*x8+start(loading)*x9+start(loading)*x10;
x1~~start(uniq)*x1;x2~~start(uniq)*x2;x3~~start(uniq)*x3;x4~~start(uniq)*x4;x5~~start(uniq)*x5;
x6~~start(uniq)*x6;x7~~start(uniq)*x7;x8~~start(uniq)*x8;x9~~start(uniq)*x9;x10~~start(uniq)*x10;
F1 ~~start(0.3)*F2'
m3 <- str_replace_all(m3, "loading", as.character(loading))
m3 <- str_replace_all(m3, "uniq", as.character(1-loading^2))
fit = sem(m3, data=NULL, std.lv=T)
sigma0 = inspect(fit, "sigma.hat")## correlation matrix.¨
s=0.5; k=1.5
is.monotone(getVM(s,k, sigma0)[[1]][1, ])
```

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

```
ICOR <- getVM(s,k, sigma0)[[2]]; colnames(ICOR) <- colnames(sigma0)
round(sigma0,3)
```

```
##     x1    x2    x3    x4    x5    x6    x7    x8    x9    x10
## x1  1.000
## x2  0.490 1.000
## x3  0.490 0.490 1.000
## x4  0.490 0.490 0.490 1.000
## x5  0.490 0.490 0.490 0.490 1.000
## x6  0.147 0.147 0.147 0.147 0.147 1.000
## x7  0.147 0.147 0.147 0.147 0.147 0.490 1.000
## x8  0.147 0.147 0.147 0.147 0.147 0.490 0.490 1.000
## x9  0.147 0.147 0.147 0.147 0.147 0.490 0.490 0.490 1.000
## x10 0.147 0.147 0.147 0.147 0.147 0.490 0.490 0.490 0.490 1.000
```

```
round(ICOR, 3)
```

```
##          x1    x2    x3    x4    x5    x6    x7    x8    x9   x10
##  [1,] 1.000 0.495 0.495 0.495 0.495 0.149 0.149 0.149 0.149 0.149
##  [2,] 0.495 1.000 0.495 0.495 0.495 0.149 0.149 0.149 0.149 0.149
##  [3,] 0.495 0.495 1.000 0.495 0.495 0.149 0.149 0.149 0.149 0.149
##  [4,] 0.495 0.495 0.495 1.000 0.495 0.149 0.149 0.149 0.149 0.149
##  [5,] 0.495 0.495 0.495 0.495 1.000 0.149 0.149 0.149 0.149 0.149
##  [6,] 0.149 0.149 0.149 0.149 0.149 1.000 0.495 0.495 0.495 0.495
##  [7,] 0.149 0.149 0.149 0.149 0.149 0.495 1.000 0.495 0.495 0.495
##  [8,] 0.149 0.149 0.149 0.149 0.149 0.495 0.495 1.000 0.495 0.495
##  [9,] 0.149 0.149 0.149 0.149 0.149 0.495 0.495 0.495 1.000 0.495
## [10,] 0.149 0.149 0.149 0.149 0.149 0.495 0.495 0.495 0.495 1.000
```

The model fits perfectly to the polychoric correlation matrix, with estimates

```
parameterestimates(sem(m3, sample.cov=ICOR, sample.nobs=500, std.lv=T))[, 1:4]
```

```
##    lhs op rhs   est
## 1   F1 =~  x1 0.703
## 2   F1 =~  x2 0.703
## 3   F1 =~  x3 0.703
## 4   F1 =~  x4 0.703
## 5   F1 =~  x5 0.703
## 6   F2 =~  x6 0.703
## 7   F2 =~  x7 0.703
## 8   F2 =~  x8 0.703
## 9   F2 =~  x9 0.703
## 10  F2 =~ x10 0.703
## 11  x1 ~~  x1 0.504
## 12  x2 ~~  x2 0.504
## 13  x3 ~~  x3 0.504
## 14  x4 ~~  x4 0.504
## 15  x5 ~~  x5 0.504
## 16  x6 ~~  x6 0.504
## 17  x7 ~~  x7 0.504
## 18  x8 ~~  x8 0.504
## 19  x9 ~~  x9 0.504
## 20 x10 ~~ x10 0.504
## 21  F1 ~~  F2 0.301
## 22  F1 ~~  F1 1.000
## 23  F2 ~~  F2 1.000
```

We can now look at Table 1 in Li (2016), which gives the average relative bias for the factor loadings. In the population this equals 0.4271143 which agrees with the values in their Table 1. For the factor correlation average bias should be (0.301497-0.3)*100/0.3=0.499 as n goes to infinity. This agrees quite well with Table 2 in Li (2016).

5 R code for the Illustration section

Here we provide R code and output for the illustration section in ("A Problem with Discretizing Vale-Maurelli in Simulation Studies" 2019).

```
C1=c(1.25, 3.75) #Condition 1
C2=c(3,21)       #Condition 2

## Target covariance
R=0.49
sigma0 <- matrix(c(1, R, R, 1), ncol=2)
## thresholds: muthen kaplan 1985 used by flora, see page 176 in muthen.
thr=c(-1.645, -0.643, 0.643, 1.645)

## Choose condition C1 for illustration.
cond = C1
s=cond[1]; k = cond[2]
vm <- getVM(s,k, sigma0)

#Fleishman coefficients,  Table 1
```

```
a <- vm[[1]][1,1]; b <- vm[[1]][1,2]; c <- vm[[1]][1,3]; d <- vm[[1]][1,4]
c(a,b,c,d)
```

```
## [1] -0.16064256  0.81888156  0.16064256  0.04916517
```

```
#thresholds for Fleishman polynomial, Table 2
new.thr <-  a + b*thr+c*thr**2+d*thr**3
new.thr
```

```
## [1] -1.2918543 -0.6338363  0.4453862  1.8399748
```

```
#Intermediate correlation, Table 3
icor <- vm[[2]][1,2]
icor
```

```
## [1] 0.5083669
```

```
## simulating a large sample
n=10^7
set.seed(1)
z = mvrnorm(n, mu=c(0,0), Sigma=matrix(c(1, icor, icor, 1), ncol=2))
vm.underlying =  a + b*z+c*z**2+d*z**3

##check skew and kurtosis
skew(vm.underlying); kurtosi(vm.underlying); cor(vm.underlying)[1,2]
```

```
## [1] 1.253952 1.251419
```

```
## [1] 3.778470 3.765357
```

```
## [1] 0.4901335
```

```
##discretize the VM vector
ordinal.data = cbind(cut(vm.underlying[,1], c(-Inf,new.thr, Inf)), cut(vm.underlying[,2], c(-Inf,new.thr

## lavaan internal estimator of the polychoric estimation
lavaan:::pc_cor_TS(ordinal.data[, 1],ordinal.data[, 2] )
```

```
## [1] 0.5086546
```

```
##thresholds
fit.y1 <- lavaan:::lavProbit(y=ordinal.data[, 1], X=NULL)
fit.y1$theta
```

```
## [1] -1.6453736 -0.6433106  0.6423761  1.6440310
```

References

"A Problem with Discretizing Vale-Maurelli in Simulation Studies." 2019. *Psychometrika*, Forthcoming.

Flora, David B, and Patrick J Curran. 2004. "An Empirical Evaluation of Alternative Methods of Estimation for Confirmatory Factor Analysis with Ordinal Data." *Psychological Methods* 9 (4). American Psychological Association:466.

Foldnes, Njål, and Steffen Grønneberg. 2015. "How General Is the Vale–Maurelli Simulation Approach?" *Psychometrika* 80 (4). Springer:1066–83.

Li, Cheng-Hsien. 2016. "Confirmatory Factor Analysis with Ordinal Data: Comparing Robust Maximum Likelihood and Diagonally Weighted Least Squares." *Behavior Research Methods* 48 (3). Springer:936–49.

Rhemtulla, Mijke, Patricia E Brosseau-Liard, and Victoria Savalei. 2012. "When can categorical variables be treated as continuous? A comparison of robust continuous and categorical SEM estimation methods under suboptimal conditions." *Psychological Methods* 17 (3):354–73.

Rosseel, Y. 2012. "Lavaan: An R Package for Structural Equation Modeling." *Journal of Statistical Software* 48 (2):1–36.