Syntax Companion

for

Latent Variable Models

An Introduction to Factor, Path, and Structural Equation Analysis

Fifth Edition

John C. Loehlin
A. Alexander Beaujean

Routledge
Taylor & Francis Group
NEW YORK AND LONDON

Last updated on 2017/09/23

Contents

Preface to Syntax Companion iii

Chapter 1: Path Models in Factor, Path, and Structural Equation Analysis 1

Chapter 2: Fitting Path Models 3

Chapter 3: Fitting Path and Structural Models to Data from a Single Group on a Single Occasion 13

Chapter 4: Fitting Models Involving Repeated Measures, Multiple Groups, or Means 29

Chapter 5: Exploratory Factor Analysis—Basics 49

Chapter 6: Exploratory Factor Analysis—Elaborations 55

Chapter 7: Issues in the Application of Latent Variable Models 66

Appendices 67

G. Power of a Test of Poor Fit and Sample Sizes Needed for Power 68

Preface to Syntax Companion

This document contains sytax for all the in-text examples from *Latent variable models: An introduction to factor, path, and structural equation analysis* (5th edition). They are presented in the same order as they are in the book.

If you notice any errors, please bring them to our attention: Alex_Beaujean@baylor.edu or loehlin@utexas.edu.

Data

All the data used for the examples in this companion text are available on the website in a compressed file named *lvm5data.zip*. All the data files are stored as plain text. The majority of the examples used covariance matrices for data; the exception is Thurstone's box data, which is raw data. The covariance matrix files are full (i.e., same values above and below the main diagonal). The covariance data in the files can either be copy-and-pasted or read directly into a model-fitting program. The variables in the covariance matrices file are unnamed, but are named for raw data.

In what follows, we show to read a covariance matrix file in R and Mplus. We assume—as we do with all the syntax in this companion—the data file is placed in the model-fitting program's working directory.

R

To set the working directory in R, use the `setwd()` function, specifying the working directory's path in quotation marks as the single argument.

To read a covariance matrix file directly into R, use the `scan()` and `matrix()` functions, with the *ncol* argument equaling the number of variables. For example, the following R syntax will read the data for the Figure 2.5 example—a four-variable correlation matrix named *Figure2_5.dat*—and name the variables X1–X4.

```
#import file
fig2.5.cor <- matrix(scan(file="Figure2_5.dat"), ncol=4)

#name variables
rownames(fig2.5.cor) <- colnames(fig2.5.cor) <- c("X1", "X2", "X3", "X4")
```

Mplus

In Mplus, the working directly is the location where the syntax file is stored.

The following Mplus syntax will read the *Figure2_5.dat* data file. Since the covariance matrix is full, I need to use the *TYPE = FULLCORR* argument. Mplus also requires a sample size when importing a covariance matrix for dat; we arbitrarily set this to 100.

```
DATA:
  FILE = Figure2_5.dat;
  TYPE = FULLCORR;
  NOBSERVATIONS = 100;
VARIABLE:
  NAMES = X1 X2 X3 X4;
```

Citation

Loehlin, J. C., & Beaujean, A. A. (2016). *Syntax companion for* Latent variable models: An introduction to factor, path, and structural equation analysis (5th ed.). Waco, TX: Baylor Psychometric Laboratory.

Chapter 1: Path Models in Factor, Path, and Structural Equation Analysis

1.1. Extended Example

While we estimated the parmeter values for Chapter 1's extended chapter "by hand" in the text, any path analysis program can also compute the estimates. We used an arbitrary sample size of 100.

1.1.1. lavaan

All the lavaan functions and arguments are described in the text or elsewhere in the syntax companion except *fixed.x=FALSE* in the `summary()` function. We use that argument because we are label the correlation between two exogenous variables (W and X).

```
#laod lavaan
library(lavaan)

#import correlations from figure 1.21
ch1.ee.data <- '
1.00
0.50 1.00
0.64 0.40 1.00
0.60 0.36 0.59 1.00
'
ch1.ee.cor <- getCov(ch1.ee.data,names=c("m", "w", "x", "y"))

# specify the model in figure 1.21 with labels
ch1.ee.model <- '
# direct paths
m ~ c*w + d*x
y ~ b*w + f*m + e*x

# label correlation
w~~a*x

# label error variances
m~~g*m
y~~h*y

# indirect effects of x to y
## through m
df := d*f
```

```
## through w
ab := a*b
## through w and m
acf := a*c*f
## sum of all indirect effect of x to y
indirect.x := df + ab + acf

# total effect of x on y
total.x := e + indirect.x
'

ch1.ee.fit <- sem(ch1.ee.model, sample.cov=ch1.ee.cor, sample.n=1000, fixed.x=FALSE)
summary(ch1.ee.fit, standardized=TRUE, rsquare=TRUE)
```

Chapter 2: Fitting Path Models

2.1. Power to Detect an Added Path

2.1.1. R

Below is the syntax for the single-path situation with extra path value = .30.

```
library(lavaan)

alt.model.1 <- '
G =~ .7*A + .7*B + .7*C + .3*D
H =~ .7*D + .7*E + .7*F
G~~.6*H

A~~.51*A
B~~.51*B
C~~.51*C
D~~.168*D
E~~.51*E
F~~.51*F
'

# implied correlations under alternative model 1
alt1.values.fit <- cfa(alt.model.1, do.fit=FALSE)
alt1.values.cov <- fitted(alt1.values.fit)$cov

# original model
original.model <- '
G =~ A + B + C
H =~ D + E + F
G~~H

A~~A
B~~B
C~~C
D~~D
E~~E
F~~F
'

# fit alt 1 data to original model
alt1.fit <- cfa(original.model, sample.cov=alt1.values.cov, sample.nobs=500, likelihood =
    "wishart")

# chi-square and df
fitmeasures(alt1.fit,fit.measures = c("chisq","df", "fmin"))
```

```
# ncp is the chi-square from fitting the alternative data to the original model
ncp.1 <- fitmeasures(alt1.fit,fit.measures = "chisq")

# critical chi-square value for alpha=.05 and 1 df
c1 <- qchisq(p=.05,df=1,ncp=0,lower.tail = FALSE)

# power for 1 df
pchisq(q=c1,df=1,ncp=ncp.1,lower.tail = FALSE)

# critical chi-square value for alpha=.05 and 8 df
c8 <- qchisq(p=.05,df=8,ncp=0,lower.tail = FALSE)

# power for 8 df
pchisq(q=c8,df=8,ncp=ncp.1,lower.tail = FALSE)
```

Below is the syntax for the single-path situations with extra path value = .10.

```
# alternative model with extra path of .1
alt.model.2 <- '
G =~ .7*A + .7*B + .7*C + .1*D
H =~ .7*D + .7*E + .7*F
G~~.6*H


A~~.51*A
B~~.51*B
C~~.51*C
D~~.416*D
E~~.51*E
F~~.51*F
'

# implied correlations under alternative model 2
alt2.values.fit <- cfa(alt.model.2, do.fit=FALSE)
alt2.values.cov <- fitted(alt2.values.fit)$cov

# fit alt 2 data to original model
alt2.fit <- cfa(original.model, sample.cov=alt2.values.cov, sample.nobs=500, likelihood = "
    wishart")

# chi-square and df
fitmeasures(alt2.fit,fit.measures = c("chisq","df", "fmin"))

# ncp is the chi-square from fitting the alternative data to the original model
ncp.2 <- fitmeasures(alt2.fit,fit.measures = "chisq")

# critical chi-square value for alpha=.05 and 8 df
c8 <- qchisq(p=.05,df=8,ncp=0,lower.tail = FALSE)

# power
pchisq(q=c8,df=8,ncp=ncp.2,lower.tail = FALSE)
```

2.2. Overall Power to Reject a Model

See syntax for Appendix G.

2.3. Figures 2.5 and 2.6

2.3.1. EQS

(a)

```
/TITLE
INPUT FOR FIG 2.5 PROBLEM
/SPECIFICATIONS
VAR=4; CAS=100; MA=COR;
ANAL=COR;
/EQUATIONS
V1 = *F1 + E1;
V2 = *F1 + E2;
V3 = *F2 + E3;
V4 = *F2 + E4;
/VARIANCES
F1, F2 = 1; E1 TO E4 = *;
/COVARIANCES
F1,F2 = *;
/MATRIX
1.00
.50 1.00
.10 .10 1.00
.20 .30 .20 1.00
/END
```

(b)

```
/TITLE
INPUT FOR FIG 2.6 PROBLEM
/SPEC
VAR=4; CAS=100; MA=COR;
ANAL=COR; ME=LS;
/EQU
V1 = .5*F1 + E1;
V2 = .5*F1 + E2;
V3 = F2 + E3;
V4 = .5*F2 + E4;
F2 = .5*F1 + D2;
/VAR
F1=1; E1 TO E4 = .5*;
D2 = .5*;
/MAT
1.00
.50 1.00
.10 .10 1.00
.20 .30 .20 1.00
/END
```

2.3.2. lavaan

Note. To match the χ^2 values in the book (i.e., use $N - 1$ instead of N), use the *likelihood="wishart"* argument in the `cfa()` or `sem()` functions.

```
library(lavaan)
#model
fig2.5.data <- '
1
.5 1
.1 .1 1
.2 .3 .2 1
'

fig2.5.cor <- getCov(fig2.5.data,
names=c(c("X1", "X2", "X3", "X4")))

fig2.5.model <- '
# latent variables
F1 =~ NA*X1 + a*X1 + b*X2
F2 =~ NA*X3 + c*X3 + d*X4

# variances/covariances
# fix variance to 1
F1 ~~ 1*F1
# fix variance to 1
F2 ~~ 1*F2
F1 ~~ e*F2

# label residual variances
X1 ~~ w*X1
X2 ~~ x*X2
X3 ~~ y*X3
X4 ~~ z*X4
'

fig2.5.fit <- cfa(fig2.5.model,
sample.cov=fig2.5.cor, sample.nobs=100)
summary(fig2.5.fit)
```

```
library(lavaan)
#model
fig2.6.model<-'
# latent variables
F1 =~ NA*X1 + a*X1 + b*X2
 #the first indicator is set to one by default
F2 =~ X3 + d*X4

# variances
F1 ~~ 1*F1 # fix variance to 1

# regression
F2 ~ e*F1

#residual variances
X1 ~~ w*X1
X2 ~~ x*X2
X3 ~~ y*X3
X4 ~~ z*X4
F2 ~~ v*F2
'

fig2.6.fit <- sem(fig2.6.model,
sample.cov=fig2.5.cor, sample.nobs=100)
summary(fig2.6.fit)
```

2.3.3. Mplus

(a)

```
TITLE: FIG 2.5 PROBLEM
DATA:
FILE = Figure2_5.dat;
TYPE = CORRELATION;
NOBSERVATIONS = 100;
VARIABLE:
NAMES = X1 X2 X3 X4;
MODEL:
F1 BY X1* X2;
F2 BY X3* X4;
F1@1;
F2@1;
```

(b)

```
TITLE: FIG 2.6 PROBLEM
DATA:
FILE = Figure2_5.dat;
TYPE = CORRELATION;
NOBSERVATIONS = 100;
VARIABLE:
NAMES = X1 X2 X3 X4;
MODEL:
F1 BY X1*.5 X2*.5;
F2 BY X3 X4*.5;
F1@1;
F2 ON F1*.5;
ANALYSIS:
ESTIMATOR = GLS;
```

2.3.4. Mx

(a)

```
INPUT FOR FIG. 2.5 PROBLEM
DATA NG=1 NI=4 NO=100
CMATRIX
1.00
.50 1.00
.10 .10 1.00
.20 .30 .20 1.00
MATRICES
A FULL 6 6
S SYMM 6 6
F FULL 4 6
I IDENT 6 6
COVARIANCES F*(I-A)~*S*((I-A)~)'*F' /
SPECIFICATION A
0 0 0 0 0 0
0 0 0 0 0 0
1 0 0 0 0 0
2 0 0 0 0 0
0 3 0 0 0 0
0 4 0 0 0 0
SPECIFICATION S
0
5 0
0 0 6
0 0 0 7
0 0 0 0 8
0 0 0 0 0 9
VALUE 1 S 1 1 S 2 2
VALUE 1 F 1 3 F 2 4 F 3 5
VALUE 1 F 4 6
END
```

(b)

```
INPUT FOR FIG. 2.6 PROBLEM
DATA NG=1 NI=4 NO=100
CMATRIX
1.00
.50 1.00
.10 .10 1.00
.20 .30 .20 1.00
MATRICES
A FULL 6 6
S SYMM 6 6
F FULL 4 6
I IDENT 6 6
COVARIANCES F*(I-A)~*S*((I-A)~)'*F' /
SPECIFICATION A
0 0 0 0 0 0
5 0 0 0 0 0
1 0 0 0 0 0
2 0 0 0 0 0
0 0 0 0 0 0
0 4 0 0 0 0
SPECIFICATION S
0
0 3
0 0 6
0 0 0 7
0 0 0 0 8
0 0 0 0 0 9
VALUE 1 S 1 1 A 5 2
VALUE 1 F 1 3 F 2 4 F 3 5
VALUE 1 F 4 6
END
```

2.3.5. OpenMx

(a)

```
library(OpenMx)
# input data
figure2_5.cor <- matrix(c(1,.50,.10,.20,.50, 1, .10,.30,.10, .10, 1, .2,.20, .30, .20,1), nrow=4,
     ncol=4, dimnames=list(c("X1", "X2", "X3", "X4"), c("X1", "X2", "X3", "X4")))
figure2_5.cor  <- mxData(observed=figure2_5.cor, type="cov", numObs=100)

# FAS matrices
figure2_5.matrF <- mxMatrix(type="Full", nrow=4, ncol=6, free=FALSE, name="F",byrow=TRUE,
values = c( 1,0,0,0,0,0,
      0,1,0,0,0,0,
      0,0,1,0,0,0,
      0,0,0,1,0,0))

figure2_5.matrA <- mxMatrix(type="Full", nrow=6, ncol=6, name="A", byrow=TRUE,
free =  c(FALSE,FALSE,FALSE,FALSE,TRUE,FALSE,
      FALSE,FALSE,FALSE,FALSE,TRUE,FALSE,
      FALSE,FALSE,FALSE,FALSE,FALSE,TRUE,
      FALSE,FALSE,FALSE,FALSE,FALSE,TRUE,
      FALSE,FALSE,FALSE,FALSE,FALSE,FALSE,
      FALSE,FALSE,FALSE,FALSE,FALSE,FALSE),
values = c(0,0,0,0,1,0,
      0,0,0,0,1,0,
      0,0,0,0,0,1,
      0,0,0,0,0,1,
      0,0,0,0,0,0,
      0,0,0,0,0,0))

figure2_5.matrS <- mxMatrix(type="Symm", nrow=6, ncol=6, name="S",byrow=TRUE,
free =  c(TRUE,FALSE,FALSE,FALSE,FALSE,FALSE,
      FALSE,TRUE,FALSE,FALSE,FALSE,FALSE,
      FALSE,FALSE,TRUE,FALSE,FALSE,FALSE,
      FALSE,FALSE,FALSE,TRUE,FALSE,FALSE,
      FALSE,FALSE,FALSE,FALSE,FALSE,TRUE,
      FALSE,FALSE,FALSE,FALSE,TRUE,FALSE),
values = c(1,0,0,0,0,0,
      0,1,0,0,0,0,
      0,0,1,0,0,0,
      0,0,0,1,0,0,
      0,0,0,0,1,1,
      0,0,0,0,1,1))

figure2_5.exp <- mxExpectationRAM(F="F",A="A",S="S",  dimnames=c("X1","X2","X3","X4","F1","F2"))
figure2_5.funML <- mxFitFunctionML()

#fit model
figure2_5.model <- mxModel("Figure 2_5",figure2_5.cor, figure2_5.matrF, figure2_5.matrA,
    figure2_5.matrS, figure2_5.exp, figure2_5.funML)
figure2_5.fit <- mxRun(figure2_5.model)
summary(figure2_5.fit)
```

(b)

```
library(OpenMx)

# input data
figure2_5.cor <- matrix(c(1,.50,.10,.20,.50, 1, .10,.30,.10, .10, 1, .2,.20, .30, .20,1), nrow=4,
     ncol=4, dimnames=list(c("X1", "X2", "X3", "X4"), c("X1", "X2", "X3", "X4")))
figure2_5.cor  <- mxData(observed=figure2_5.cor, type="cov", numObs=100)

# FAS matrices
figure2_6.matrF <- mxMatrix(type="Full", nrow=4, ncol=6, free=FALSE, name="F",byrow=TRUE,
values = c(1,0,0,0,0,0,
      0,1,0,0,0,0,
      0,0,1,0,0,0,
      0,0,0,1,0,0))

figure2_6.matrA <- mxMatrix(type="Full", nrow=6, ncol=6, name="A",byrow=TRUE,
free =  c(FALSE,FALSE,FALSE,FALSE,TRUE,FALSE,
      FALSE,FALSE,FALSE,FALSE,TRUE,FALSE,
      FALSE,FALSE,FALSE,FALSE,FALSE,FALSE,
      FALSE,FALSE,FALSE,FALSE,FALSE,TRUE,
      FALSE,FALSE,FALSE,FALSE,FALSE,FALSE,
      FALSE,FALSE,FALSE,FALSE,TRUE,FALSE),
values = c(0,0,0,0,1,0,
      0,0,0,0,1,0,
      0,0,0,0,0,1,
      0,0,0,0,0,1,
      0,0,0,0,0,0,
      0,0,0,0,1,0))

figure2_6.matrS <- mxMatrix(type="Symm", nrow=6, ncol=6, name="S",byrow=TRUE,
free =  c(TRUE,FALSE,FALSE,FALSE,FALSE,FALSE,
      FALSE,TRUE,FALSE,FALSE,FALSE,FALSE,
      FALSE,FALSE,TRUE,FALSE,FALSE,FALSE,
      FALSE,FALSE,FALSE,TRUE,FALSE,FALSE,
      FALSE,FALSE,FALSE,FALSE,FALSE,FALSE,
      FALSE,FALSE,FALSE,FALSE,FALSE,TRUE),
values = c(1,0,0,0,0,0,
      0,1,0,0,0,0,
      0,0,1,0,0,0,
      0,0,0,1,0,0,
      0,0,0,0,1,0,
      0,0,0,0,0,1))

figure2_6.exp <- mxExpectationRAM(F="F",A="A",S="S",  dimnames=c("X1","X2","X3","X4","F1","F2"))
figure2_6.funML <- mxFitFunctionML()

#fit model
figure2_6.model <- mxModel("Figure 2_6",figure2_5.cor, figure2_6.matrF, figure2_6.matrA,
    figure2_6.matrS, figure2_6.exp, figure2_6.funML)
figure2_6.fit <- mxRun(figure2_6.model)
summary(figure2_6.fit)
```

2.3.6. SIMPLIS/LISREL

(a)

```
INPUT FOR FIG. 2.5 PROBLEM
OBSERVED VARIABLES X1 X2 X3 X4
CORRELATION MATRIX
  1.00
  .50   1.00
  .10   .10  1.00
  .20   .30   .20  1.00
LATENT VARIABLES F1 F2
SAMPLE SIZE 100
PATHS
  F1 -> X1 X2
  F2 -> X3 X4
END OF PROBLEM
```

(b)

```
INPUT FOR FIG. 2.6 PROBLEM
OBSERVED VARIABLES X1 X2 X3 X4
CORRELATION MATRIX
  1.00
  .50 1.00
  .10 .10 1.00
  .20 .30 .20 1.00
LATENT VARIABLES F1 F2
SAMPLE SIZE 100
PATHS
  F1 -> (.5)*X1 (.5)*X2 (.5)*F2
  F2 -> 1*X3 (.5)*X4
OPTIONS UL ND=3
END OF PROBLEM
```

2.4. Extended Example

2.4.1. lavaan

```
#import data
ch2.data <- '
1.00
.30 1.00
.20 .20 1.00
.10 .20 .30 1.00
'
# make symmetric matrix and name variables
ch2.cor <- getCov(ch2.data, names=c("A","B","C","D"))
#models
model.a <- '
G =~ a*A + b*B + c*C + d*D
'
model.b <- '
E =~ a*A + b*B
F =~ c*C + d*D
'

#fit models
fit.a <- cfa(model.a, sample.cov=ch2.cor, sample.nobs=120)
fit.b <- cfa(model.b, sample.cov=ch2.cor,sample.nobs=120)
```

Chapter 3: Fitting Path and Structural Models to Data from a Single Group on a Single Occasion

3.1. Desegregation—Maruyama & McGarvey (1980)

3.1.1. lavaan

```
mm.cor <- matrix(scan(file="MaruyamaMcGarvey.dat"), ncol=13)
rownames(mm.cor) <- colnames(mm.cor) <-c("SEI","EDH","RP","VACH","VGR","RAV","PEA","FEV","MEV","
    TEV","SP","PP","WP")

library(lavaan)

model.3.3 <-'
#latent variables
SES =~ NA*SEI + EDH + RP
ABL =~ NA*RAV + PEA
ASA =~ NA*FEV + MEV + TEV
ACH =~ VACH + VGR
APR =~ SP + PP + WP
# latent variances
SES~~1*SES
ABL~~1*ABL
ASA~~1*ASA
# latent covariances
ABL ~~ SES
SES ~~ 0*ASA
ABL ~~ 0*ASA
# regressions
APR ~ ASA + ACH
ACH ~ SES + ABL + APR
'

model.3.3.fit <- sem(model.3.3, sample.cov=mm.cor, sample.nobs=249)
summary(model.3.3.fit,standardized=TRUE)
```

3.1.2. Mplus

```
TITLE:
Figure 3.3 Example
DATA:
  FILE = MaruyamaMcGarvey.dat;
  TYPE - FULLCORR;
  NOBSERVATIONS = 100;
VARIABLE:
  NAMES = SEI EDH RP VACH VGR RAV PEA FEV MEV TEV SP PP WP;
MODEL:
! standardize latent variables
  SES@1;
  ABL@1;
  ASA@1;
! Set covariances to zero
  SES WITH ASA@0;
  ABL WITH ASA@0;
! Regression
  APR ON ASA ACH;
  ACH ON SES ABL APR;
OUTPUT:
standardized;
```

3.2. Attitudes Toward Police—McIver, Carmines, and Zeller (1980)

3.2.1. lavaan

Note. In lavaan, the *std.lv=TRUE* argument will standardize all latent variables and free all factor loadings. The `fitMeasures()` function provides the desired fit measures for a fit model (if the *fit.measure* argument is not specified, then all fit measures are returned). The `resid()` function with the *type="cor"* argument specified returns the residual correlations.

```
mcz.cor <- matrix(scan(file="McIver.dat"), ncol=9)
rownames(mcz.cor) <-colnames(mcz.cor) <-c("PS", "RE", "RT", "HO", "CO", "ET", "BU", "VA", "RO")
library(lavaan)

#original model
model.3.4<-'
# latent variables
F1 =~ a*PS + b*RE + c*RT
F2 =~ d*HO + e*CO + f*ET
F3 =~ g*BU + h*VA + i*RO
# label latent variances
F1 ~~ j*F2
F1 ~~ k*F3
F2 ~~ l*F3
```

```
# revised model
model.3.4.rev <-'
# latent variables
F1 =~ a*PS + b*RE + c*RT
F2 =~ d*HO + e*CO + f*ET + RT
F3 =~ g*BU + h*VA + i*RO
# label latent variances
F1 ~~ j*F2
F1 ~~ k*F3
F2 ~~ l*F3
```

```
# second model revision
model.3.4.rev2 <-'
# latent variables
F1 =~ a*PS + b*RE + c*RT
F2 =~ d*HO + e*CO + f*ET + RT
F3 =~ g*BU + h*VA + i*RO
# label latent variances
F1 ~~ j*F2
F1 ~~ k*F3
F2 ~~ l*F3
# free residual correlations
#4 & 9
HO ~~ RO
#2 & 7
RE ~~ BU
```

```
# fit original model
model.3.4.fit <- cfa(model.3.4, sample.cov=mcz.cor, sample.nobs=11000, std.lv=TRUE)

# model fit measures
fitMeasures(model.3.4.fit, fit.measure=c("chisq", "df", "pvalue", "rmsea", "rmsea.ci.lower", "
    rmsea.ci.upper"))

# residual correlations
resid(model.3.4.fit,type="cor")
```

3.3. Parallel and Congeneric Tests—Jöreskog (1978)/Lord (1956)

See extended example.

3.4. Estimating Reliability

See extended example.

3.5. Multitrait-Multimethod Models—Bentler and Lee (1979)

3.5.1. lavaan

```
bl.cor <- matrix(scan(file="BentlerLee.dat"),ncol=12)
rownames(bl.cor) <-colnames(bl.cor) <-c("Ep", "Ap", "Ip", "Mp", "Et", "At", "It", "Mt", "Es", "As
    ", "Is", "Ms")

library(lavaan)
#constrain some residual variances to be > 1
model.3.7 <- '
# latent variables
E =~ Ep + Et + Es
A =~ Ap + At + As
I =~ Ip + It + Is
M =~ Mp + Mt + Ms
P =~ Ep + Ap + Ip + Mp
T =~ Et + At + It + Mt
S =~ Es + As + Is + Ms
# constrain trait and method covariances to 0
E ~~ 0*P
A ~~ 0*P
I ~~ 0*P
M ~~ 0*P
E ~~ 0*T
A ~~ 0*T
I ~~ 0*T
M ~~ 0*T
E ~~ 0*S
A ~~ 0*S
I ~~ 0*S
M ~~ 0*S
# constrain residual variances
Mp ~~ a*Mp
a > 0.00001
Is ~~ b*Is
b > 0.000001
Ep ~~c*Ep
c > 0.0001
'
# fit
model.3.7.fit <- cfa(model.3.7, sample.cov=bl.cor, sample.nobs=68, std.lv=TRUE)
```

```
# fix variance of MP and IS to .10
model.3.7.alt <-'
# latent variables
E =~ Ep + Et + Es
A =~ Ap + At + As
I =~ Ip + It + Is
M =~ Mp + Mt + Ms
P =~ Ep + Ap + Ip + Mp
T =~ Et + At + It + Mt
S =~ Es + As + Is + Ms
# constrain trait and method covariances to 0
E ~~ 0*P
A ~~ 0*P
I ~~ 0*P
M ~~ 0*P
E ~~ 0*T
A ~~ 0*T
I ~~ 0*T
M ~~ 0*T
E ~~ 0*S
A ~~ 0*S
I ~~ 0*S
M ~~ 0*S
# set Mp and IS variances
Mp ~~ a*Mp
a == .10
Is ~~ b*Is
b == .10
'
```

```
model.3.7.alt.fit <- cfa(model.3.7.alt, sample.cov=bl.cor, sample.nobs=68, std.lv=TRUE)
```

3.6. Partial Correlation

3.6.1. lavaan

Note. The `inspect()` function using the *what="cor.lv"* argument returns the correlations among the latent variables.

```
pc.cor <- matrix(scan(file="Figure3_8.dat"),ncol=3)
rownames(pc.cor) <-colnames(pc.cor) <-c("x", "y", "z")

library(lavaan)

# uncorrected partial correlation
model.pc <-'
# regressions
x ~ z
y ~ z
# partial correlation
x~~y
'
# partial correlation
model.pc.fit <- cfa(model.pc, sample.cov=pc.cor, sample.nobs=100,estimator="uls")
summary(model.pc.fit,standardized=TRUE)

# corrected partial correlation
model.3.8 <-'
# correct for unreliability in x, y, and z
a =~ 1*x
b =~ 1*y
c =~ 1*z
x~~.19*x # 1 - .9*.9 = .19
y~~.19*y # 1 - .9*.9 = .19
z~~.51*z # 1 - .7*.7 = .51
# corrected regressions
a ~ c
b ~ c
# corrected partial correlation
a~~b
'
#corrected partial correlation
model.3.8.fit <- cfa(model.3.8, sample.cov=pc.cor, sample.nobs=100,estimator="uls")
summary(model.3.8.fit,standardized=TRUE)

# latent correlations
inspect(model.3.8.fit, what="cor.lv")
```

3.7. Head Start—Bentler and Woodward (1978)

3.7.1. lavaan

Note. Instead of creating a revised model explicitly, we used the *constraint* argument to constrain the regression to zero. The `anova()` function tests the difference in χ^2 values for nested models.

```
bw.cor <- matrix(scan(file="BentlerWoodward.dat"),ncol=7)
rownames(bw.cor) <-colnames(bw.cor) <-c("MED", "FED", "FOC", "INC", "HS", "ITPA", "MRT")

# original model
model.3.9 <-'
# measurement model
ED =~ NA*MED + FED
SES =~ NA*FED + FOC + INC  + MED
OC =~  NA*FOC + INC
HSL =~ 1*HS
COG =~ 1*ITPA + 1*MRT
# latent exogenous variances
ED ~~ 1*ED
SES ~~ 1*SES
OC ~~ 1*OC
HSL ~~ 1*HSL
# residuals
INC ~~ 0*INC
HS ~~ 0*HS
# set the Cog indicator residuals equal to each other
ITPA ~~ r*ITPA
MRT ~~ r*MRT
# set covariances for ED and SES to zero
ED ~~ 0*SES
SES ~~ 0*OC
# regression
COG ~ SES + x*HSL
'

# fit original model
model.3.9.fit <- cfa(model.3.9, sample.cov=bw.cor, sample.nobs=303)
# fit revised model, constraining regression path x to 0
model.3.9.rev.fit <- cfa(model.3.9, sample.cov=bw.cor, sample.nobs=303, constraint="x==0")

# test difference in models' chi-square values
anova(model.3.9.rev.fit,model.3.9.fit)
```

3.8. Career Aspirations—Duncan, Haller, and Portes (1968)

3.8.1. lavaan

```
dhp.cor <- matrix(scan(file="Duncan.dat"),ncol=10)
rownames(dhp.cor) <-colnames(dhp.cor) <-c("RPA", "RIQ", "RSES", "REA", "ROA", "FPA", "FIQ", "FSES
    ", "FEA", "FOA")

# model 1 (unconstrained model)
model.3.11 <-'
# measurement
RAMB =~ 1*REA + a*ROA
FAMB =~ 1*FEA + b*FOA
RPAL =~ .837*RPA
RIQL =~ .894*RIQ
RSESL =~ .949*RSES
FSESL =~ .949*FSES
FIQL =~ .894*FIQ
FPAL =~ .837*FPA
# latent variances
RPAL ~~ 1*RPAL
RIQL ~~ 1*RIQL
RSESL ~~ 1*RSESL
FSESL ~~ 1*FSESL
FIQL ~~ 1*FIQL
FPAL ~~ 1*FPAL
# latent covariances
RPAL ~~ rpa_riq*RIQL
RPAL ~~ rpa_rses*RSESL
RSESL ~~ rses_riq*RIQL
FPAL ~~ fpa_fiq*FIQL
FPAL ~~ fpa_fses*FSESL
FSESL ~~ fses_fiq*FIQL
RPAL ~~ rpa_fiq*FIQL
RPAL ~~ rpa_fses*FSESL
RSESL ~~ rses_fiq*FIQL
FPAL ~~ fpa_riq*RIQL
FPAL ~~ fpa_rses*RSESL
FSESL ~~ fses_riq*RIQL
#regressions
RAMB ~  rpa*RPAL + riq*RIQL + rses*RSESL + rw*FSESL + rx*FAMB
FAMB ~  fpa*FPAL + fiq*FIQL + fses*FSESL + fw*RSESL + fx*RAMB
#residual variances
REA ~~ rear*REA
FEA ~~ fear*FEA
ROA ~~ roar*ROA
FOA ~~ foar*FOA
RAMB ~~ rambr*RAMB
FAMB ~~ fambr*FAMB
RPA~~rpar*RPA
```

```
RIQ ~~ riqr*RIQ
RSES ~~ rsesr*RSES
FSES ~~ fsesr*FSES
FIQ ~~ fiqr*FIQ
FPA ~~ frar*FPA
#residual covariances
FAMB ~~ y*RAMB
FOA ~~ z1*ROA
REA ~~ z2*FEA
'
# fit model 1
model.3.11.m1.fit <- cfa(model.3.11, sample.cov=dhp.cor, sample.nobs=329)

# equality constraints for model 2
m2.constraints <-'
# correlations among the source variables
rpa_riq == fpa_fiq
rpa_rses == fpa_fses
rses_riq == fses_fiq
# correlations across pairs
rpa_fiq == fpa_riq
rpa_fses == fpa_rses
rses_fiq == fses_riq
# paths from source variables to Ambition
rpa == fpa
riq==fiq
rses==fses
rw==fw
# reciprocal path
rx==fx
# loadings
a == b
#residuals
rear==fear
roar == foar
rambr == fambr
'

# fit model 2
model.3.11.m2.fit <- cfa(model.3.11, sample.cov=dhp.cor, sample.nobs=329, constraints=m2.
    constraints)

# fit model 3
model.3.11.m3.fit <- cfa(model.3.11, sample.cov=dhp.cor, sample.nobs=329, constraints=c(m2.
    constraints, "fw==0"))

# fit model 4
model.3.11.m4.fit <- cfa(model.3.11, sample.cov=dhp.cor, sample.nobs=329,  constraints=c(m2.
    constraints, "fx==0"))

# fit model 5
model.3.11.m5.fit <- cfa(model.3.11, sample.cov=dhp.cor, sample.nobs=329, constraints=c(m2.
    constraints, "y==0"))

# fit model 6
model.3.11.m6.fit <- cfa(model.3.11, sample.cov=dhp.cor, sample.nobs=329, constraints=c(m2.
    constraints, "y==0", "fx==0"))

# fit model 7
model.3.11.m7.fit <- cfa(model.3.11, sample.cov=dhp.cor, sample.nobs=329, constraints=c(m2.
    constraints, "z2==0", "z1==0"))
```

3.8.2. Mplus

Model 1

```
title: duncan-haller-portes model 1
data:
file=Duncan.dat;
type=FULLCORR;
nobservations=329;
variable:
names=rpa riq rses rea roa fpa fiq fses fea foa;
model:
rpal by rpa@.837;
riql by riq@.894;
rsesl by rses@.949;
fpal by fpa@.837;
fiql by fiq@.894;
fsesl by fses@.949;
ramb by rea@1.0;
famb by fea@1.0;
ramb by roa;
famb by foa ;
ramb on rpal;
famb on fpal;
ramb on riql;
famb on fiql;
ramb on rsesl ;
famb on fsesl ;
ramb on fsesl ;
famb on rsesl ;
ramb on famb ;
famb on ramb ;
rpal@1;
riql@1;
rsesl@1;
fpal@1;
fiql@1;
fsesl@1;
rpal with riql ;
fpal with fiql ;
rpal with rsesl ;
fpal with fsesl ;
riql with rsesl ;
fiql with fsesl ;
rpal with fiql ;
fpal with riql ;
rpal with fsesl ;
fpal with rsesl ;
riql with fsesl ;
fiql with rsesl ;
rea with fea ;
roa with foa ;
ramb with famb ;
rpal with fpal ;
riql with fiql ;
rsesl with fsesl ;
ramb ;
famb ;
rea ;
fea ;
```

```
roa ;
foa ;
rpa ;
riq ;
rses;
fpa ;
fiq ;
fses ;
```

Model 2

```
title: duncan-haller-portes model 2
data:
file=Duncan.dat;
type=FULLCORR;
nobservations=329;
variable:
names=rpa riq rses rea roa fpa fiq fses fea foa;
model:
rpal by rpa@.837;
riql by riq@.894;
rsesl by rses@.949;
fpal by fpa@.837;
fiql by fiq@.894;
fsesl by fses@.949;
ramb by rea@1.0;
famb by fea@1.0;
ramb by roa (1) ;
famb by foa (1);
ramb on rpal (2) ;
famb on fpal (2) ;
ramb on riql (3) ;
famb on fiql (3) ;
ramb on rsesl (4);
famb on fsesl (4);
ramb on fsesl (5);
famb on rsesl (5);
ramb on famb (6);
famb on ramb (6);
rpal@1;
riql@1;
rsesl@1;
fpal@1;
fiql@1;
fsesl@1;
rpal with riql (7);
fpal with fiql (7);
rpal with rsesl (8);
fpal with fsesl (8);
riql with rsesl (9);
fiql with fsesl (9);
rpal with fiql (10);
fpal with riql (10);
rpal with fsesl (11);
fpal with rsesl (11);
riql with fsesl (12);
fiql with rsesl (12);
rea with fea ;
roa with foa ;
ramb with famb ;
rpal with fpal ;
riql with fiql ;
```

```
rsesl with fsesl ;
ramb (13);
famb (13);
rea (14);
fea (14);
roa (15);
foa (15);
rpa ;
riq ;
rses;
fpa ;
fiq ;
fses ;
```

3.9. Nonlinear Effects Among Latent Variables—Kenny and Judd (1984)

3.9.1. lavaan

```
kj.cov <- matrix(scan(file="KennyJudd.dat"), ncol=6)
rownames(kj.cov) <-colnames(kj.cov) <-c("A", "B", "A2", "B2", "AB", "Y")

model.3.12 <- '
#latent variables
X =~ 1*A + b*B
X2 =~ 1*A2 + f*B2 + b*AB
XU =~ 2*A2 + b*AB
XV =~ 1*AB + i*B2
#regression
Y ~ c*X + d*X2
#set covariances to 0
X ~~ 0*X2
X ~~ 0*XU
X ~~ 0*XV
X2 ~~ 0*XU
X2 ~~ 0*XV
XU ~~ 0*XV
#label latent variances
X ~~   vX*X
X2 ~~ vX2*X2
XU ~~ vXU*XU
XV ~~ vXV*XV
#label residual variances
A~~vU*A
B~~vV*B
A2~~vU2*A2
B2~~vV2*B2
AB~~vUV*AB
Y~~vZ*Y
# constraints
vX2 == 2*(vX^2)
vXU == vX*vU
vXV == vX*vV
```

```
vU2 == 2*(vU^2)
vV2 == 2*(vV^2)
vUV == vU*vV
f==b^2
i==2*b
'

model.3.12.fit <- sem(model.3.12, sample.cov=kj.cov, sample.nobs=500)
```

3.10. Extended Example

3.10.1. lavaan

```
lord.cov <- matrix(scan(file="Lord.dat"),ncol=4)
rownames(lord.cov) <-colnames(lord.cov) <-c("A", "B", "C", "D")

# hypothesis 1
ch.3.6.h1.model <- '
# latent variable
    structure
V1 =~  a*A + b*B
V2 =~  c*C + d*D
# label covariance
V1 ~~ i*V2
# label variance
A ~~ e*A
B ~~ f*B
C ~~ g*C
D ~~ h*D
# constraints
a==b
c==d
e==f
g==h
i==1
# reliability
omega := (a + b + c +
    d)^2 / ((a + b + c
     + d)^2 + e + f +
    g+ h)
'

# hypothesis 2
ch.3.6.h2.model <- '
# latent variables
V1 =~  a*A + b*B
V2 =~  c*C + d*D
# label covariance
V1 ~~ i*V2
# label variance
A ~~ e*A
B ~~ f*B
C ~~ g*C
D ~~ h*D
# constraints
a==b
c==d
e==f
g==h
# reliability
omega.v1 := (a + b)^2
    / ((a + b)^2 + e +
     f)
omega.v2 := (c + d)^2
    / ((c + d)^2 + g +
     h)
'

# hypothesis 3
ch.3.6.h3.model <-'
# latent variable
    structure
V1 =~  a*A + b*B
V2 =~  c*C + d*D
# label covariance
V1 ~~ i*V2
# label variance
A ~~ e*A
B ~~ f*B
C ~~ g*C
D ~~ h*D
# constraints
i==1
# reliability
omega := (a + b + c +
    d)^2 / ((a + b + c
     + d)^2 + e + f +
    g+ h)
'

# hypothesis 4
ch.3.6.h4.model<-'
# latent variable
    structure
V1 =~  a*A + b*B
V2 =~  c*C + d*D
# label covariance
V1 ~~ i*V2
# label variance
A ~~ e*A
B ~~ f*B
C ~~ g*C
D ~~ h*D
# reliability
omega.v1 := (a + b)^2
     / ((a + b)^2 + e
     + f)
omega.v2 := (c + d)^2
     / ((c + d)^2 + g
     + h)
'

# fit models
ch.3.6.h1.fit <- cfa(ch.3.6.h1.model, sample.cov=lord.cov, sample.nobs=649,std.lv=TRUE)
ch.3.6.h2.fit <- cfa(ch.3.6.h2.model, sample.cov=lord.cov, sample.nobs=649,std.lv=TRUE)
ch.3.6.h3.fit <- cfa(ch.3.6.h3.model, sample.cov=lord.cov, sample.nobs=649, std.lv=TRUE)
ch.3.6.h4.fit <- cfa(ch.3.6.h4.model, sample.cov=lord.cov, sample.nobs=649, std.lv=TRUE)
```

3.10.2. Mplus

To create a new parameter in Mplus, use the `MODEL CONSTRAINT` section with the `NEW ()` function, palcing the new parameter's name in the parentheses—e.g., *NEW (omega)*.

```
DATA:
  file is Lord.dat;
  type is FULLCOV;
  nobservations are 649;
VARIABLE:
  names are A B C D;
MODEL:
! have to put indicators on separate lines since they are labeled
  V1 BY A* (a)
    B  (b);
  V2 BY C* (c)
      D  (d);
  V1@1;
  V2@1;
! covariance
  V1 WITH V2* (i);
!residual variances
  A* (e);
    B* (f);
    C* (g);
    D* (h);
OUTPUT:
  sampstat standardized;
```

```
! H1
MODEL CONSTRAINT:
a = b;
c = d;
e = f;
g = h;
i = 1;

NEW ( omega );
omega = (a+b+c+d)**2/((
    a+b+c+d)**2+e+f+g+h
    );
```

```
! H2
MODEL CONSTRAINT:
a = b;
c =d;
e = f;
g = h;

NEW ( omega1 );
   omega1 = (a+b)**2/((
    a+b)**2+e+f);
NEW ( omega2 );
   omega2 = (c+d)**2/((
    c+d)**2+g+h);
```

```
! H3
MODEL CONSTRAINT:
   i = 1;

NEW ( omega );
omega = (a+b+c+d)**2/((
    a+b+c+d)**2+e+f+g+h
    );
```

```
! H4
MODEL CONSTRAINT:
NEW ( omega1 );
omega1 = (a+b)**2/((a+b
    )**2+e+f);
NEW ( omega2 );
omega2 = (c+d)**2/((c+d
    )**2+g+h);
```

Chapter 4: Fitting Models Involving Repeated Measures, Multiple Groups, or Means

4.1. A Minitheory of Love—Tesser and Paulhus (1976)

4.1.1. lavaan

```
tp.cor <- matrix(scan(file="TesserPaulhus.dat"),ncol=8)
rownames(tp.cor) <- colnames(tp.cor) <-c("T1", "L1", "C1", "D1", "T2", "L2", "C2", "D2")

library(lavaan)

# original model
model.4.1 <-'
#Latent Variables
A1 =~ NA*T1 + a*T1 + b*L1 + c*C1 + d*D1
A2 =~ NA*T2 + a*T2 + b*L2 + c*C2 + d*D2
# standardized A1
A1 ~~ 1*A1
#regression
A2 ~ m*A1
#residuals
A2 ~~ n2*A2
T1 ~~ e2*T1
T2 ~~ e2*T2
L1 ~~ f2*L1
L2 ~~ f2*L2
C1 ~~ g2*C1
C2 ~~ g2*C2
D1 ~~ h2*D1
D2 ~~ h2*D2
#residual covariances
T1 ~~ i*T2
L1 ~~ j*L2
C1 ~~ k*C2
D1 ~~ l*D2
'

# revised model
model.4.1.rev <-'
#Latent Variables
A1 =~ NA*T1 + a*T1 + b*L1 + c*C1 + d*D1
A2 =~ NA*T2 + a*T2 + b*L2 + c*C2 + d*D2
# standardized A1
A1 ~~ 1*A1
#regression
A2 ~ m*A1
#residuals
A2 ~~ n2*A2
T1 ~~ e2*T1
T2 ~~ e2*T2
L1 ~~ f2*L1
L2 ~~ f2*L2
C1 ~~ g2*C1
C2 ~~ g2*C2
D1 ~~ h2*D1
D2 ~~ h2*D2
#residual covariances
T1 ~~ i*T2
L1 ~~ j*L2
C1 ~~ k*C2
D1 ~~ l*D2
T1~~C2
C1~~L2
'

# results using correlations
tp.cor.fit <- sem(model.4.1, sample.cov=tp.cor, sample.nobs=202)
summary(tp.cor.fit)

# residual correlations
resid(tp.cor.fit, type="cor")

# ============================
# = Results Using covariances =
# ============================
# SDs
tp.sd <- c(3.59,19.49,1.80, 2.87, 3.75, 20.67, 1.72,3.16)
# convert correlations to covariances
tp.cov <- cor2cov(tp.cor, tp.sd)

tp.cov.fit <- sem(model.4.1, sample.cov=tp.cov, sample.nobs=202)
summary(tp.cov.fit)
```

4.2. Simplex—Bracht and Hopkins (1972)

4.2.1. lavaan

```
bh.cor <- matrix(scan(file="BrachtHopkinsCor.dat"),ncol=7)
bh.sd <- c(.51, .69, .89, 1.01, 1.2, 1.26, 1.38)
rownames(bh.cor) <-colnames(bh.cor) <-c("T1", "T2", "T3", "T4", "T5", "T6", "T7")

library(lavaan)
bh.cov <- cor2cov(bh.cor, bh.sd)

# original model
model.4.2 <-'
#latent variables
A1 =~1*T1
A2 =~1*T2
A3 =~1*T3
A4 =~1*T4
A5 =~1*T5
A6 =~1*T6
A7 =~1*T7
# regressions
A2 ~ w2*A1
A3 ~ w3*A2
A4 ~ w4*A3
A5 ~ w5*A4
A6 ~ w6*A5
A7 ~ w7*A6
#residuals
T1 ~~ u2*T1
T2 ~~ u2*T2
T3 ~~ u3*T3
T4 ~~ u4*T4
T5 ~~ u5*T5
T6 ~~ u6*T6
T7 ~~ u6*T7
A2 ~~z2*A2
A3 ~~z3*A3
A4 ~~z4*A4
A5 ~~z5*A5
A6 ~~z6*A6
A7 ~~z7*A7
# define A variances
A1 ~~ a1*A1
a2 := w2 * a1 * w2 + z2
a3 := w3 * a2 * w3 + z3
a4 := w4 * a3 * w4 + z4
a5 := w5 * a4 * w5 + z5
a6 := w6 * a5 * w6 + z6
a7 := w7 * a6 * w7 + z7
'
```

```
# revised model
model.4.2.rev <-'
#latent variables
A1 =~1*T1
A2 =~1*T2
A3 =~1*T3
A4 =~1*T4
A5 =~1*T5
A6 =~1*T6
A7 =~1*T7
# regression
A2 ~ w*A1
A3 ~ w*A2
A4 ~ w*A3
A5 ~ w*A4
A6 ~ w*A5
A7 ~ w*A6
#residuals
T1 ~~ u*T1
T2 ~~ u*T2
T3 ~~ u*T3
T4 ~~ u*T4
T5 ~~ u*T5
T6 ~~ u*T6
T7 ~~ u*T7
A2 ~~z*A2
A3 ~~z*A3
A4 ~~z*A4
A5 ~~z*A5
A6 ~~z*A6
A7 ~~z*A7
'
```

```
bh.fit <- sem(model.4.2, sample.cov=bh.cov, sample.nobs=795)
bh.rev.fit <- sem(model.4.2.rev, sample.cov=bh.cov, sample.nobs=795)

# compare models
anova(bh.rev.fit,bh.fit)
```

4.2.2. Mplus

```
TITLE: FIG 4.2 PROBLEM

DATA:
FILE = BrachtHopkinsCov.dat;
TYPE = FULLCOV;
NOBSERVATIONS = 795;
VARIABLE:
NAMES = T1 T2 T3 T4 T5 T6 T7;

MODEL:
! latent variables
A1 BY T1@1;
A2 BY T2@1;
A3 BY T3@1;
A4 BY T4@1;
A5 BY T5@1;
A6 BY T6@1;
A7 BY T7@1;

! regression
A2 ON A1 (w2);
A3 ON A2 (w3);
A4 ON A3 (w4);
A5 ON A4 (w5);
A6 ON A5 (w6);
A7 ON A6 (w7);

! residuals
T1 (u2);
T2 (u2);
T3 (u3);
T4 (u4);
T5 (u5);
T6 (u6);
T7 (u6);

A2 WITH A2 (z2);
A3 WITH A3 (z3);
A4 WITH A4 (z4);
A5 WITH A5 (z5);
A6 WITH A6 (z6);
A7 WITH A7 (z7);

OUTPUT:
stand;
```

4.3. Liberal-Conservative Attitudes—Judd and Milburn (1980)

4.3.1. lavaan

```
# college sample
jm.col.cor <- matrix(scan(file="JuddMilburn_College.dat"),ncol=9)
rownames(jm.col.cor) <-colnames(jm.col.cor) <-c("B72", "C72","J72","B74", "C74", "J74", "B76", "
    C76", "J76")
jm.col.sd <- c(2.03, 1.84, 1.67, 1.76, 1.68,  1.48,  1.74,  1.83, 1.54)

library(lavaan)
jm.col.cov <- cor2cov(jm.col.cor, jm.col.sd)

#original model
model.4.3 <-'
# latent variables
L72 =~ B72 + C72 + J72
L74 =~ B74 + C74 + J74
L76 =~ B76 + C76 + J76
# regression
L74 ~ L72
L76 ~ L74
#residual covariances
B72 ~~ B74 + B76
B74 ~~ B76
C72 ~~ C74 + C76
C74 ~~ C76
J72 ~~ J74 + J76
J74 ~~ J76
'

#revised model
model.4.3.rev <-'
# latent variables
L72 =~ B72 + C72 + J72
L74 =~ B74 + C74 + J74
L76 =~ B76 + C76 + J76
# regression
L74 ~ L72
L76 ~ L74 + L72
#residual covariances
B72 ~~ B74 + B76
B74 ~~ B76
C72 ~~ C74 + C76
C74 ~~ C76
J72 ~~ J74 + J76
J74 ~~ J76
'

jm.col.fit <- cfa(model.4.3, sample.cov=jm.col.cov, sample.nobs=143)
jm.rev.col.fit <- cfa(model.4.3.rev, sample.cov=jm.col.cov, sample.nobs=143)

# difference in models
anova(jm.rev.col.fit,jm.col.fit)
```

4.4. Attitudes in Elite and Non-Elite Groups—Judd and Milburn (1980)

4.4.1. lavaan

```
# no college sample
jm.nocol.cor <- matrix(scan(file="JuddMilburn_NoCollege.dat"),ncol=9)
rownames(jm.nocol.cor) <-colnames(jm.nocol.cor) <-c("B72","C72","J72","B74","C74","J74","B76",
    "C76", "J76")
jm.nocol.sd <- c(1.25,  2.11,  1.90,  1.31,  1.97,  1.82,  1.34,  2.00,  1.79)

library(lavaan)
jm.nocol.cov <- cor2cov(jm.nocol.cor, jm.nocol.sd)

#combine college and no college samples into a single lists
jm.all.cov <- list(College=jm.col.cov, noCollege=jm.nocol.cov)
jm.all.n <- list(College=143,noCollege=203)

# model fit to no college sample
jm.nocol.fit <- cfa(model.4.3, sample.cov=jm.nocol.cov, sample.nobs=203)

# no constraints
jm.combined.fit <- cfa(model.4.3, sample.cov=jm.all.cov, sample.nobs=jm.all.n)
summary(jm.combined.fit, standardized=TRUE)

# all constraints
jm.combined.fit2 <- sem(model.4.3, sample.cov=jm.all.cov, sample.nobs=jm.all.n, group.equal =
c("loadings","regressions","lv.variances","residual.covariances","residuals"))

# structural constraints
jm.combined.fit3 <- sem(model.4.3, sample.cov=jm.all.cov, sample.nobs=jm.all.n, group.equal =
c("regressions"))
```

4.5. Genetics of Numerical Ability—Loehlin and Vandenberg (1968)

4.5.1. lavaan

```
mzm.cor <-matrix(scan(file="VandenbergMZM.dat"),ncol=6)
mzf.cor <-matrix(scan(file="VandenbergMZF.dat"),ncol=6)
dzm.cor <-matrix(scan(file="VandenbergDZM.dat"),ncol=6)
dzf.cor <-matrix(scan(file="VandenbergDZF.dat"),ncol=6)

rownames(mzm.cor) <- colnames(mzm.cor) <- rownames(mzf.cor) <-colnames(mzf.cor) <- rownames(dzm.
    cor) <-colnames(dzm.cor) <- rownames(dzf.cor) <- colnames(dzf.cor) <- c("AD1","MU1","TH1","
    AD2","MU2","TH2")

# SD
mzm.sd <- c(7.37, 13.81, 16.93,  8.17, 13.33, 17.56)
mzf.sd <- c(8.00, 12.37, 15.19,  6.85, 11.78, 14.76)
dzm.sd <- c(9.12, 16.51, 17.20,  7.70, 14.52, 14.74)
dzf.sd <- c(8.99, 15.44, 16.98,  7.65, 14.59, 18.56)

#convert correlations to covariances
library(lavaan)
mzm.cov <- cor2cov(mzm.cor, mzm.sd)
mzf.cov <- cor2cov(mzf.cor, mzf.sd)
dzm.cov <- cor2cov(dzm.cor, dzm.sd)
dzf.cov <- cor2cov(dzf.cor, dzf.sd)


#combine the covariance and sample sizes into a single list objects
lv.cov <- list(MZM=mzm.cov, MZF=mzf.cov, DZM=dzm.cov, DZF=dzf.cov)
lv.n <- list(MZM=63, MZF=59, DZM=29, DZF=46)

# original model
model.4.4 <-'
#latent variables
N1 =~ NA*AD1 + c(a,a,a,a)*AD1 + c(b,b,b,b)*MU1 + c(c,c,c,c)*TH1
N2 =~ NA*AD2 + c(a,a,a,a)*AD2 + c(b,b,b,b)*MU2 + c(c,c,c,c)*TH2
N1 ~~ c(1,1,.5,.5)*N2
N1 ~~ 1*N1
N2 ~~ 1*N2
# residual variances
AD1 ~~ c(r,r,r,r)*AD1
AD2 ~~ c(r,r,r,r)*AD2
MU1 ~~ c(r1,r1,r1,r1)*MU1
MU2 ~~ c(r1,r1,r1,r1)*MU2
TH1 ~~ c(r2,r2,r2,r2)*TH1
TH2 ~~ c(r2,r2,r2,r2)*TH2
# same-trait cross-twin residual covariances
AD1 ~~ c(s,s,s,s)*AD2
MU1 ~~ c(s1,s1,s1,s1)*MU2
TH1 ~~ c(s2,s2,s2,s2)*TH2
```

```
# within-twin residual covariances across trait
AD1 ~~ c(d,d,d,d)*MU1
AD2 ~~ c(d,d,d,d)*MU2
AD1 ~~ c(d1,d1,d1,d1)*TH1
AD2 ~~ c(d1,d1,d1,d1)*TH2
MU1 ~~ c(d2,d2,d2,d2)*TH1
MU2 ~~ c(d2,d2,d2,d2)*TH2
# across-twin residual covariances across trait
AD1 ~~ c(e,e,e,e)*TH2
TH1 ~~ c(e,e,e,e)*AD2
AD1 ~~ c(e1,e1,e1,e1)*MU2
MU1 ~~ c(e1,e1,e1,e1)*AD2
MU1 ~~ c(e2,e2,e2,e2)*TH2
TH1 ~~ c(e2,e2,e2,e2)*MU2
#calculate parameters in table
# manifest variances
vAD := a*a + r
vMU := b*b + r1
vTH := c*c + r2
# heritability/common genetic variance
AD.h2 := a^2
MU.h2 := b^2
TH.h2 := c^2
# Other shared variance--cross-twin within-trait residual correlation
AD.ct := s/(sqrt(vAD)*sqrt(vAD))
MU.ct := s1/(sqrt(vMU)*sqrt(vMU))
TH.ct := s2/(sqrt(vTH)*sqrt(vTH))
# common genetic covariance--genetic contribution to within-person correlation
AD.MU.wpr := a*b
AD.TH.wpr := a*c
MU.TH.wpr := b*c
# other shared correlation---cross-person cross-trait residual covariance
AD.MU.cpct := e1/(sqrt(vAD)*sqrt(vMU))
AD.TH.cpct := e/(sqrt(vAD)*sqrt(vTH))
TH.MU.cpct := e2/(sqrt(vTH)*sqrt(vMU))
# environmental contributions to correlations--within-person cross-trait residual correlation
AD.MU.wpct := d/(sqrt(vAD)*sqrt(vMU))
AD.TH.wpct := d1/(sqrt(vAD)*sqrt(vTH))
TH.MU.wpct := d2/(sqrt(vTH)*sqrt(vMU))
# residual correlations
rAD.MU := AD.MU.wpct - AD.MU.cpct
rAD.TH := AD.TH.wpct - AD.TH.cpct
rTH.MU := TH.MU.wpct - TH.MU.cpct
# "average" manifest correlations
corAD.MU := (a*b + d)/(sqrt(vAD)*sqrt(vMU))
corAD.TH := (a*c + d1)/(sqrt(vAD)*sqrt(vTH))
corTH.MU := (c*b + d2)/(sqrt(vTH)*sqrt(vMU))
'

lv.fit <- cfa(model.4.4, sample.cov=lv.cov, sample.nobs=lv.n)
summary(lv.fit,standardized=TRUE)

# revised model
model.4.4.rev <-'
#latent variables
N1 =~ NA*AD1 + c(aM,aF,aM,aF)*AD1 + c(bM,bF,bM,bF)*MU1 + c(cM,cF,cM,cF)*TH1
N2 =~ NA*AD2 + c(aM,aF,aM,aF)*AD2 + c(bM,bF,bM,bF)*MU2 + c(cM,cF,cM,cF)*TH2
N1 ~~ c(1,1,.5,.5)*N2
N1 ~~ 1*N1
N2 ~~ 1*N2
#within-twin residual (co)variances
```

```
AD1 ~~ c(rM,rF,rM,rF)*AD1
AD2 ~~ c(rM,rF,rM,rF)*AD2
MU1 ~~ c(r1M,r1F,r1M,r1F)*MU1
MU2 ~~ c(r1M,r1F,r1M,r1F)*MU2
TH1 ~~ c(r2M,r2F,r2M,r2F)*TH1
TH2 ~~ c(r2M,r2F,r2M,r2F)*TH2
AD1 ~~ c(dM,dF,dM,dF)*MU1
AD1 ~~ c(d1M,d1F,d1M,d1F)*TH1
MU1 ~~ c(d2M,d2F,d2M,d2F)*TH1
AD2 ~~ c(dM,dF,dM,dF)*MU2
AD2 ~~ c(d1M,d1F,d1M,d1F)*TH2
MU2 ~~ c(d2M,d2F,d2M,d2F)*TH2
#between-twin residual (co)variances
AD1 ~~ c(eM,eF,eM,eF)*MU2
MU1 ~~ c(eM,eF,eM,eF)*AD2
AD1 ~~ c(e1M,e1F,e1M,e1F)*TH2
TH1 ~~ c(e1M,e1F,e1M,e1F)*AD2
MU1 ~~ c(e2M,e2F,e2M,e2F)*TH2
TH1 ~~ c(e2M,e2F,e2M,e2F)*MU2
AD1 ~~ c(sM,sF,sM,sF)*AD2
MU1 ~~ c(s1M,s1F,s1M,s1F)*MU2
TH1 ~~ c(s2M,s2F,s2M,s2F)*TH2
'

lv.rev.fit <- cfa(model.4.4.rev, sample.cov=lv.cov, sample.nobs=lv.n)
summary(lv.rev.fit,standardized=TRUE)

% compare models
anova(lv.rev.fit,lv.fit)
```

4.5.2. LISREL

Note. We input the correlations and standard deviations directly.

```
numerical ability-group 1 MZ males
da ng=4 ni=6 no=63
km
1.0
.670 1.0
.489 .555 1.0
.598 .499 .526 1.0
.627 .697 .560 .784 1.0
.456 .567 .725 .576 .540 1.0
sd
7.37, 13.81, 16.93,  8.17, 13.33, 17.56
mo ny=6 ne=2 ps=sy,fi te= sy,fi
va 1.0 ps 1 1 ps 2 2 ps 2 1
fr ly 1 1 ly 2 1 ly 3 1
eq ly 1 1 ly 4 2
eq ly 2 1 ly 5 2
eq ly 3 1 ly 6 2
fr te 2 1 te 3 1 te 3 2
eq te 2 1 te 5 4
eq te 3 1 te 6 4
eq te 3 2 te 6 5
fr te 1 1 te 2 2 te 3 3
eq te 1 1 te 4 4
eq te 2 2 te 5 5
eq te 3 3 te 6 6
fr te 4 1 te 5 2 te 6 3
fr te 4 2 te 4 3 te 5 3
eq te 4 2 te 5 1
eq te 4 3 te 6 1
eq te 5 3 te 6 2
st .5 all
ou nd=3 ns

group 2 DZ males
da no=29
km
1.0
.664 1.0
.673 .766 1.0
.073 .313 .239 1.0
.194 .380 .347 .739 1.0
.379 .361 .545 .645 .751 1.0
sd
9.12, 16.51, 17.20,  7.70, 14.52, 14.74
mo ly=in te=in ps=ps
va .5 ps 2 1
ou

group 3 MZ females
da no=59
km
1.0
.611 1.0
.754 .676 1.0
.673 .464 .521 1.0
.622 .786 .635 .599 1.0
```

```
.614 .636 .650 .574 .634 1.0
sd
8.00, 12.37, 15.19,  6.85, 11.78, 14.76
mo
va 1.0 ps 2 1
ou

group 4 DZ females
da no=46
km
1.0
.779 1.0
.674 .679 1.0
.462 .412 .500 1.0
.562 .537 .636 .620 1.0
.392 .359 .565 .745 .603 1.0
sd
8.99, 15.44, 16.98,  7.65, 14.59, 18.56
mo
va .5 ps 2 1
ou
```

4.5.3. Mplus

The *VandenbergCombined.dat* file has the standard deviations and correlations for each of the four groups in a single file.

```
TITLE: FIG 4.11 PROBLEM
DATA:
FILE = VandenbergCombined.dat;
TYPE = STDEVIATIONS FULLCORR;
NOBSERVATIONS = 63 59 29 46;
NGROUPS = 4;
VARIABLE:
NAMES = AD1 MU1 TH1 AD2 MU2 TH2;
MODEL:
!MZM
!latent variables
Num1 BY   AD1* (a)
    MU1* (b)
    TH1* (c);
Num2 BY   AD2* (a)
    MU2* (b)
    TH2* (c);
! constrain latent variance and covariance
Num1@1;
Num2@1;

Num1 WITH Num2@1;

!residual variances
AD1* (r); AD2* (r);
MU1* (r1); MU2* (r1);
TH1* (r2); TH2* (r2);

!same-trait cross-twin residual covariances
AD1 WITH AD2* (s);
MU1 WITH MU2* (s1);
TH1 WITH TH2* (s2);

!within-twin residual covariances across trait
AD1 WITH MU1* (d);
AD2 WITH MU2* (d);

AD1 WITH TH1* (d1);
AD2 WITH TH2* (d1);

MU1 WITH TH1* (d2);
MU2 WITH TH2* (d2);

!across-twin residual covariances across trait
AD1 WITH TH2* (e);
TH1 WITH AD2* (e);

AD1 WITH MU2* (e1);
MU1 WITH AD2* (e1);

MU1 WITH TH2* (e2);
TH1 WITH MU2* (e2);

! MZF
MODEL g2:
```

```
! constrain latent covariance
Num1 WITH Num2@1.0;

! DZM
MODEL g3:
! constrain latent covariance
Num1 WITH Num2@0.5;

! DZF
MODEL g4:
! constrain latent covariance
Num1 WITH Num2@0.5;

OUTPUT:
SAMPSTAT STANDARDIZED TECH1;
```

4.6. Heredity, environment, and sociability

The average reliability-corrected correlations for each relationship pair are

MZ:	.529	DZ:	.046
Parent-adopted child:	.023	Parent-natural child:	.201
Adopted children:	−.149		

To estimate the model parameters, we used general purpose equation solving programs that allowed for parameter constraints.

4.6.1. R

In R, the *limSolve* package provides the `lsei()` function that will solve equations with parameter constraints.

Note. Fisher's z transformation is the inverse hyperbolic tangent function. In R, this is the `atanh()` function. Also, the matrix multiplication operation is `%*%`.

```
library(limSolve)
#combined sample sizes
n <- c(102+45, 34+119,257+271,56+54,48+80)
#average corrected correlations
B <- c(.529,.046,.023,.201,-.149)

#model 1 coefficients
A1 <- matrix(nrow=5, ncol=1, data=c(1,1,1,1,1))
# solve for parameters
lsei(A = A1, B = B, fulloutput = TRUE, verbose = FALSE)
# approximate chi-square
sum((atanh(B)-atanh(A1%*%lsei(A = A1, B = B, fulloutput = TRUE, verbose = FALSE)$X))^2*(n-3))

# model 2
A2 <- matrix(nrow=5, ncol=1, data=c(1,.5,0,.5,0))
# constraints to make parameter values >= 0
G2 <- matrix(nrow = 2, ncol = 1, byrow = TRUE, data = c(1, 0))
H2 <- c(0,0)
lsei(A = A2, B = B, G = G2, H = H2, fulloutput = TRUE,type=2)
sum((atanh(B)-atanh(A2%*%lsei(A = A2, B = B, G = G2, H = H2, fulloutput = TRUE, verbose = FALSE)
    $X))^2*(n-3))

# model 3
A3 <- matrix(nrow=5, ncol=2, data=c(1,.5,0,.5,0,1,1,1,1,1))
G3 <- matrix(nrow = 2, ncol = 2, byrow = TRUE, data = c(1, 0, 0, 1))
H3 <- c(0,0)
lsei(A = A3, B = B, G = G3, H = H3, fulloutput = TRUE, ,type=2)
sum((atanh(B)-atanh(A3%*%lsei(A = A3, B = B, G = G3, H = H3, fulloutput = TRUE, verbose = FALSE)
    $X))^2*(n-3))

# model 4
A4 <- matrix(nrow=5, ncol=2, data=c(1,.5,0,.5,0,1,.25,0,0,0))
G4 <- matrix(nrow = 2, ncol = 2, byrow = TRUE, data = c(1, 0, 0, 1))
```

```
H4 <- c(0,0)
lsei(A = A4, B = B, G = G4, H = H4, fulloutput = TRUE, verbose = FALSE)
sum((atanh(B)-atanh(A4%*%lsei(A = A4, B = B, G = G4, H = H4, fulloutput = TRUE, verbose = FALSE)
    $X))^2*(n-3))

# model 5
A5 <- matrix(nrow=5, ncol=3, data=c(1,.5,0,.5,0,1,1,0,0,1,0,0,1,1,0))
G5 <- matrix(nrow = 3, ncol = 3, byrow = TRUE, data = c(1, 0, 0, 0, 1,0,0,0,1))
H5 <- c(0,0,0)
lsei(A = A5, B = B, G = G5, H = H5, fulloutput = TRUE, verbose = FALSE)
sum((atanh(B)-atanh(A5%*%lsei(A = A5, B = B, G = G5, H = H5, fulloutput = TRUE, verbose = FALSE)
    $X))^2*(n-3))
```

4.7. Stress, Resources, and Depression—Holahan and Moos (1991)

See extended example

4.8. Latent Curve Models—Willett and Sayer (1994)

4.8.1. lavaan

Note. We use the `growth()` function in lavaan to fit this model.

```
ws.cov <-matrix(scan(file="WillettSayer.dat"),ncol=5)
ws.mean<-c(.2008,.2263,.3255,.4168,-.0788)
rownames(ws.cov) <-colnames(ws.cov) <- c("T11", "T12", "T13", "T14", "EX11")

model.4.8 <-'
# single-indicator E
E =~ 1*EX11
EX11 ~~ .0139*EX11

#latent variables
int =~ 1*T11 + 1*T12 + 1*T13 + 1*T14
slo =~ 0*T11 + 1*T12 + 2*T13 + 3*T14

#latent covariance
int ~~0*slo

#means
int ~ j*1
slo ~ k*1
E ~ i*1

#regression
int ~ a*E
slo ~ b*E

#error variance
T11~~e*T11
T12~~f*T12
T13~~g*T13
T14~~h*T14
slo ~~ d*slo
int ~~ c*int
'
ws.fit1 <- growth(model.4.8, sample.cov=ws.cov, sample.mean=ws.mean, sample.nobs=168)

#fit model with alternative means
ws.fit2 <- growth(model.4.8, sample.cov=ws.cov, sample.mean=c(.1200,.2263,.3255,.4168,-.0788),
    sample.nobs=168)
```

4.8.2. Mplus

Note. We created a new data file (*Figure4_8.dat*) that contains the means on the first row and covariance matrix (found in *WillettSayer.dat*) on the subsequent rows.

```
TITLE: FIG 4.8 PROBLEM
DATA:
FILE = Figure4_8.dat;
TYPE = MEANS FULLCOV;
NOBSERVATIONS = 168;
VARIABLE:
NAMES = T11 T12 T13 T14 E11;
MODEL:
E BY E11@1;
E11@.0139;

i s | T11@0 T12@1 T13@2 T14@3;
[i s];
i s ON E;
i WITH s@0;

OUTPUT: stand;
```

4.9. Extended Example

```
library(lavaan)
hm.high.cor <-matrix(scan(file="HolahanHighStress.dat"),ncol=5)
hm.low.cor <-matrix(scan(file="HolahanLowStress.dat"),ncol=5)
hm.high.sd <- c(5.97,7.98,3.97,2.27,4.91)
hm.low.sd <- c(4.84,6.33,3.84,2.14,4.43)
hm.high.cov <- cor2cov(hm.high.cor,hm.high.sd)
hm.low.cov <- cor2cov(hm.low.cor,hm.low.sd)
hm.high.mean <-c(8.82,13.87,15.24,7.92,19.03)
hm.low.mean <-c(6.15,9.96,15.14,8.80,20.43)

# combine data
hm.cov <- list(low=hm.low.cov, high=hm.high.cov)
hm.n <- list(low=126, high=128)
hm.mean <- list(low=hm.low.mean, high=hm.high.mean)
```

```
#original model
model.4.7 <-'
# latent variables
D =~  NA*DM + a*DM + b*DF
R =~  NA*SC + c*SC + d*EG + e*FS
#error variances
DM~~m*DM
DF~~n*DF
SC~~o*SC
EG~~p*EG
FS~~q*FS
#intercepts
DM ~ h*1
DF ~ i*1
SC ~ j*1
EG ~ k*1
FS ~ l*1
# latent variances
D ~~ c(1,NA)*D
R ~~ c(1, NA)*R
'
```

```
# revised model
model.4.7.rev <-'
# latent variables
D =~  NA*DM + a*DM + b*DF
R =~  NA*SC + c*SC + d*EG + e*FS
#error variances
DM~~m*DM
DF~~n*DF
SC~~o*SC
EG~~p*EG
FS~~q*FS
#intercepts
DM ~ h*1
DF ~ i*1
SC ~ j*1
EG ~ k*1
FS ~ l*1
# latent variances
D ~~ c(1,NA)*D
R ~~ c(1, NA)*R
# latent means
D ~ c(fl,fh)*1
R ~ c(gl,gh)*1
fl==0
gl==0
fh==-1*gh
'
```

```
# fit original model
hm.fit1 <- cfa(model.4.7, sample.cov=hm.cov, sample.nobs=hm.n, sample.mean=hm.mean, group.equal=c
    ("loadings", "residuals", "intercepts"))
summary(hm.fit1, standardized=TRUE)

# fit revised model
hm.fit2 <- cfa(model.4.7.rev, sample.cov=hm.cov, sample.nobs=hm.n, sample.mean=hm.mean, group.
    equal=c("loadings", "residuals", "intercepts"))
summary(hm.fit2, standardized=TRUE)

#compare models
anova(hm.fit1,hm.fit2)
```

Chapter 5: Exploratory Factor Analysis—Basics

5.1. Determining the Number of Factors

5.1.1. R

```
# import data and name variables
hs.cor <- matrix(scan(file="HolzingerSwineford.dat"),ncol=14)

colnames(hs.cor) <- rownames(hs.cor) <- c("T1", "T2", "T3.4", "T6","T28", "T29","T32","T34",
    "T35","T36a","T13","T18","T25b","T77")


# eigenvalues
eigen(hs.cor)$values

# scree plot
plot(eigen(hs.cor)$values, type="b", ylab="Eigenvalue Size", xlab="Eigenvalue Number")
# add horizontal line at y = 1
abline(h=1)

# parallel analysis
require(psych)
fa.parallel(hs.cor, n.obs=355)

# minimum average partial correlations
require(psych)
nfactors(hs.cor,n.obs=355)$map
```

5.2. Rotation

5.2.1. R

Note. For the rotations, we used the GPArotation package's `GPForth()` (orthogonal) and `GPFoblq()` (oblique) functions. We use the *psych* package's rotation options in this chapter's extended example.

```
# import correlations from table 5-6 and name variables
table5.6.cor <- matrix(scan(file="Table5_6.dat"),ncol=6)
colnames(table5.6.cor) <- rownames(table5.6.cor) <- c("C","D","E","F","G","H")

# initial, unrotated principal factors, using exact communalities
library(psych)
table5.6.pf <- fa(table5.6.cor,nfactors=2,rotate="none",fm="pa",SMC=c(.64,.36,.37,.61,.49,.36))
    $loadings

# quartimax loadings and rotation matrix
library(GPArotation)
GPForth(table5.6.pf, method="quartimax")

# varimax loadings and rotation matrix
GPForth(table5.6.pf, method="varimax")

# oblimin loadings and rotation matrix
GPFoblq(table5.6.pf, method="oblimin")

# structure coefficients for oblimin rotation (using matrix multiplication)
GPFoblq(table5.6.pf, method="oblimin")$loadings%*%GPFoblq(table5.6.pf, method="oblimin")$Phi
```

5.3. Thurstone's Box Problem

See the Extended Example.

5.4. Factor Analysis Using Packaged Programs

5.4.1. SPSS

```
TITLE  'CHAPTER 5 EXAMPLE'.
MATRIX DATA VARIABLES=ROWTYPE_ D E F G H.
BEGIN DATA
N 100 100 100 100 100
CORR 1.00
CORR  .20  1.00
CORR  .24   .58  1.00
CORR  .00   .56   .41  1.00
CORR  .00   .21   .21   .51  1.00
END DATA.
FACTOR
  /MATRIX=IN (COR=*)
  /DIAGONAL=.16 .74 .55 .91 .36
  /CRITERIA=FACTORS (3)
  /EXTRACTION=PAF
  /ROTATION=VARIMAX
  /METHOD=CORRELATION.
```

5.4.2. SAS

```
DATA EXAMP (TYPE=CORR);
_TYPE_ ='CORR';
INPUT _NAME_ $ D E F G H;
CARDS;
  D 1.0 . . . .
  E  .20  1.0 . . .
  F  .24   .58  1.0 . .
  G  .00   .56   .41  1.0 .
  H  .00   .21   .21   .51  1.0
  ;
PROC FACTOR  METHOD=PRIN  NFACT=3 ROTATE=VARIMAX;
  PRIORS .16 .74 .55 .91 .36;
  TITLE 'CHAPTER 5 EXAMPLE';
RUN;
```

5.4.3. R—`fa()` function

```
#load lavaan package for data entry
library(lavaan)
# input correlation matrix and name variables
ch5.data <- '
```

```
1.00
.20  1.00
.24   .58  1.00
.00   .56   .41  1.00
.00   .21   .21   .51  1.00
'
ch5.cor <- getCov(ch5.data, names=c("D","E","F","G","H"))

# load psych package
library(psych)

# conduct exploratory factor analysis
fa(ch5.cor,nfactors=3,fm="pa", rotate="varimax",SMC=c(.16,.74,.55,.91,.36))
```

5.4.4. Mplus

```
TITLE: CHAPTER 5 EXAMPLE
DATA:
FILE = Chapter5.dat;
TYPE = FULLCORR;
NOBSERVATIONS = 100;
VARIABLE:
NAMES ARE D E F G H;
ANALYSIS:
TYPE = EFA 1 3;
ESTIMATOR = ULS;
ROTATION=VARIMAX;
```

Specifying communalities in Mplus requires using the exploratory SEM (ESEM) procedure (see Chapter 7 of the text). ESEM in Mplus can only be done with raw data, so we simulated 1000 (standardized) observations from the correlation matrix.

```
# simulate data based on correlation matrix for Mplus ESEM
require(MASS)
ch5.data <- mvrnorm(n = 1000, rep(0, 5), ch5.cor,empirical=TRUE)
write.table(ch5.data, file="Chapter5Raw.dat",row.names = FALSE, col.names = FALSE)
```

To use Varimax rotation in ESEM requires using the Crawford-Ferguson (CF) version of the criterion. The *ROWSTANDARDIZATION = KAISER* option applies Kaiser normalization. The communalities are set indirectly by specifying the uniqueness for each variable in the `MODEL CONSTRAINT` section.

```
TITLE: CHAPTER 5 EXAMPLE USING ESEM
DATA:
FILE = Chapter5Raw.dat;
TYPE = INDIVIDUAL;
VARIABLE:
NAMES ARE D E F G H;
ANALYSIS:
ESTIMATOR = ML;
ROTATION=CF-VARIMAX (ORTHOGONAL);
ROWSTANDARDIZATION = KAISER;
MODEL:
! (*1) is needed for the exploratory analysis
F1-F3 BY D-H (*1);
```

```
! label uniquenesses
D-H (uD-uH);
! specify communality values
MODEL CONSTRAINT:
uD = .84;
uE = .26;
uF = .45;
uG = .09;
uH = .64;
```

5.5. Extended Example

5.5.1. R

```
# import data
thurstone.data <- read.table(file="ThurstoneBox.dat",header=TRUE)

# correlations of variables
thurstone.cor <-cor(thurstone.data)

#eigenvalues
eigen(thurstone.cor)

#scree plot
library(psych)
scree(thurstone.data, hline=TRUE, factors=FALSE)

#parallel analysis
fa.parallel(thurstone.data)

# minimum average partial
nfactors(thurstone.data)$map

#EFA with no rotation
fa(thurstone.cor, nfactors=3, rotate="none", fm="pa")
# varimax rotation
fa(thurstone.cor, nfactors=3, rotate="varimax", fm="pa")
# oblimin rotation
fa(thurstone.cor, nfactors=3, rotate="oblimin", fm="pa")
```

Chapter 6: Exploratory Factor Analysis—Elaborations

6.1. Rescalings—Alpha and Canonical Factors

6.1.1. R

The syntax below imports the correlation matrix, creates the reduced correlation matrix, and rescales the reduced matrix to produce the alpha and canonical matrixes.

```
table5.1.R <- matrix(scan(file="Table5_1.dat"),ncol=5)
colnames(table5.1.R) <- rownames(table5.1.R) <- c("D","E","F","G","H")

# reduced correlation matrix
table5.1.Rr <- table5.1.R
diag(table5.1.Rr) <- c(.16,.74,.55,.91,.36)

# alpha rescaling
H <- diag(5)
diag(H) <- sqrt(c(.16,.74,.55,.91,.36))
Hinv <- solve(H)
alpha.cor <- Hinv%*%table5.1.Rr%*%Hinv

# canonical rescaling
U <- diag(5)
diag(U) <- sqrt(1-c(.16,.74,.55,.91,.36))
Uinv <- solve(U)
canonical.cov <- Uinv%*%table5.1.Rr%*%Uinv
```

The syntax below extracts the first three factors using eigenvalues and eigenvectors, and then rescales the results.

```
# principal factors
V <- eigen(table5.1.Rr)$vectors[,1:3]
L <- diag(3)
diag(L) <- sqrt(eigen(table5.1.Rr)$values[1:3])
P <- V%*%L

# alpha factors
V.alpha <- eigen(alpha.cov)$vectors[,1:3]
L.alpha <- diag(3)
diag(L.alpha) <- sqrt(eigen(alpha.cov)$values[1:3])
# switch the loadings when creating P to match book
P.alpha <- round(V.alpha%*%L.alpha,3)*-1
```

```
# rescale
H%*%P.alpha

# canonical factors
V.canonical <- eigen(canonical.cov)$vectors[,1:3]
L.canonical <- diag(3)
diag(L.canonical) <- sqrt(eigen(canonical.cov)$values[1:3])
P.canonical <- round(V.canonical%*%L.canonical,3)
# rescale
U%*%P.canonical
```

Alternatively, the alpha and canonical rescalings can be thought of as producing covariance matrices, then a "typical" EFA can be run specifying that the input is a covariance matrix. The resulting loadings can then be rescaled to the original metric. In the *psych* package's `fa()` function, the *covar=TRUE* argument tells the function to factor a covariance matrix.

```
table5.1.R <- matrix(scan(file="Table5_1.dat"),ncol=5)
colnames(table5.1.R) <- rownames(table5.1.R) <- c("D","E","F","G","H")

# reduced correlation matrix
table5.1.Rr <- table5.1.R
diag(table5.1.Rr) <- c(.16,.74,.55,.91,.36)

require(psych)
# principal factors
principal.fa <- fa(R, nfactors=3, fm="pa", rotate="none", SMC=c(.16,.74,.55,.91,.36))
principal.fa$loadings

# alpha
# alpha factor
H <- diag(5)
diag(H) <- sqrt(c(.16,.74,.55,.91,.36))
Hinv <- solve(H)
alpha.cov <- Hinv%*%table5.1.Rr%*%Hinv

alpha.fa <- fa(alpha.cov, nfactors=3, fm="pa", rotate="none", covar=TRUE, SMC=c(1,1,1,1,1))
# laodings
alpha.fa$loadings
# rescale
H%*%alpha.fa$loadings


# canonical
# canonical factor
U <- diag(5)
diag(U) <- sqrt(1-c(.16,.74,.55,.91,.36))
Uinv <- solve(U)
canonical.cov <- Uinv%*%table5.1.Rr%*%Uinv

canonical.fa <-fa(canonical.cov, nfactors=3, fm="pa", rotate="none", covar=TRUE)
# laodings
canonical.fa$loadings
# rescale
U%*%canonical.fa$loadings
```

6.2. Alternative Rotation Methods

6.2.1. R

```
table5.6.cor <- matrix(scan(file="Table5_6.dat"),ncol=6)
colnames(table5.6.cor) <- rownames(table5.6.cor) <- c("C","D","E","F","G","H")

# promax solution---promax() is past of base R library
promax(unrotated.pa$loadings)

# geomin
library(GPArotation)
GPFoblq(unrotated.pa$loadings,  normalize=TRUE, method="geomin")

# quartimax loadings and rotation matrix
library(GPArotation)
GPForth(table5.6.pf, method="quartimax")
```

6.2.2. Mplus

```
TITLE: CHAPTER 6 PROMAX EXAMPLE
DATA:
FILE = Table5_6.dat;
TYPE = FULLCORR;
NOBSERVATIONS = 100;
VARIABLE:
NAMES ARE C D E F G H;
ANALYSIS:
TYPE = EFA 2 2;
ESTIMATOR = ULS;
ROTATION=PROMAX;
ROWSTANDARDIZATION = KAISER;
```

```
TITLE: CHAPTER 6 GEOMIN EXAMPLE
DATA:
FILE = Table5_6.dat;
TYPE = FULLCORR;
NOBSERVATIONS = 100;
VARIABLE:
NAMES ARE C D E F G H;
ANALYSIS:
TYPE = EFA 2 2;
ESTIMATOR = ULS;
ROTATION=GEOMIN;
ROWSTANDARDIZATION = KAISER;
```

6.3. Factor Scores

6.3.1. R

Computing factor scores through matrix operations:

```
# factor loadings from figure 5.6
P <- matrix(c(.8,.6,.4,.4,0,0,0,0,.3,.5,.7,.6),ncol=2)
# uniqueness from 1 - h2 in figure 5.6
U2 <- diag(6)
diag(U2) <- c(.36,.64,.63,.39,.51,.64)
# factor correlations
F <- matrix(c(1,.5,.5,1),ncol=2)
# correlation matrix
R <- P%*%F%*%t(P) + U2
# inverse correlation matrix
invR <- solve(R)
# structure coefficients
S <- P%*%F
# beta weights
B <- invR%*%S
# scaling matrix
BS <- t(B)%*%S
D <- diag(2)
diag(D) <- diag(BS)
D.12 <- D
diag(D.12) <- 1/sqrt(diag(D.12))
# factor score weights
W <- B%*%D.12
# new data
Z <- matrix(c(1.2,-1,-.7,.6,-1.6,1.2,1.5,-.1,.9,.8,0,-1,.1,.8,-1.3,1.1,-1.4,.7),nrow=3)
#new factor scores
Zf <- Z%*%W
```

Computing factor scores through the *psych* package's `fa()` and *predict()* functions for an Oblimin rotation.

```
# factor loadings from figure 5.6
P <- matrix(c(.8,.6,.4,.4,0,0,0,0,.3,.5,.7,.6),ncol=2)
# uniqueness from 1 - h2 in figure 5.6
U2 <- diag(6)
diag(U2) <- c(.36,.64,.63,.39,.51,.64)
# factor correlations
F <- matrix(c(1,.5,.5,1),ncol=2)
# correlation matrix
R <- P%*%F%*%t(P) + U2

library(psych)
scores.fa <- fa(R,nfactors=2,rotate="oblimin",fm="pa")
# structure coefficients
scores.fa$Structure
# beta weights
scores.fa$weights
# new data
Z <- matrix(c(1.2,-1,-.7,.6,-1.6,1.2,1.5,-.1,.9,.8,0,-1,.1,.8,-1.3,1.1,-1.4,.7),nrow=3)
# new factor scores
predict(scores.fa,Z)
```

6.4. Higher-Order Factors

6.4.1. R

What follows is the Schmid–Leiman transformation of a second-order EFA using matrix multiplication (`%*%`).

```
table6.9.cor <- matrix(scan(file="Table6_9.dat"),ncol=4)
colnames(table6.9.cor) <- rownames(table6.9.cor) <- c("D","E","F","G")

# first order EFA
sl.o1.fa <- fa(table6.9.cor, nfactors=2, fm="pa", rotate="oblimin", n.obs=100, SMC=c
    (.36,.64,.25,.25))
P01 <- sl.o1.fa$loadings
F <- sl.o1.fa$Phi

# second order EFA
sl.o2.fa <- fa(F, nfactors=1, fm="pa", rotate="none", n.obs=100, SMC=c(.64,.36))
P12 <- sl.o2.fa$loadings
U1 <- diag(sqrt(1-sl.o2.fa$communality))

# Schmid-Leiman transformation
P02 <- P01%*%P12
P01r <- P01 %*% U1
cbind(P02,P01r)

# h2
sl.o1.fa$communality
# u2
sl.o1.fa$uniqueness
```

The *psych* package has the `schmid()` function, which will do a higher-order EFA and apply the Schmid–Leiman transformation.

```
table6.9.cor <- matrix(scan(file="Table6_9.dat"),ncol=4)
colnames(table6.9.cor) <- rownames(table6.9.cor) <- c("D","E","F","G")

schmid(table6.9.cor, nfactors=2, fm="pa", rotate="oblimin", n.obs=100, SMC=c(.36,.64,.25,.25))
```

The `schmid()` function assumes there is a single second-order factor and multiple first order factors. Sometimes data require more orders or more top-order factors. We wrote the `sl()` function as an extension of the `schmid()` function to handle three orders and multiple factors at the highest order.

```
# extension of the schmid() function to perform the Schmid-Leiman transformation with up to three
     factor orders and have multipel factors at the highest order
# R argument: original correlation matrix.
# o1.factors argument: number of first-order factors. Has to be specified
# o2.factors argument: number of second-order factors. Has to be specified
# o3.factors argument: number of third-order factors, default is 0
```

```
# fm argument: factor extraction method, default is principal axis. Other options are those
    listed for the fm argument of the psych function
# rotate argument: factor rotation, default is Promax. Other options are those listed for the
    rotate argument of the psych function, but needs to be oblique
# n.obs argument: number of observations
# o1.SMC argument: initial communality estimates for first-order factors. There are three options.
     1. TRUE, uses squared multiple correlations (default option). 2. FALSE, uses 1 as initial
    communality estimate. 3. For principal axis extraction (fm="pa"), if o1.SMC is a vector of
    length the number of first-order variables, then these values are used as starting values.
# o2.SMC argument: same as o1.SMC, but for second-order factors
# o3.SMC argument: same as o1.SMC, but for third-order factors
# max.iter argument: maximum number of iterations for convergence. Applies to all orders. Default
     value is 1000
# h2.print argument: print communality and uniqueness estimates. Default is FALSE, so these
    values are not printed. Set it to TRUE to print them
# min.err argument: change-in-communalities criterion to stop the iteration process, default is
    .001

sl <- function(R=NULL,o1.factors=NULL,o2.factors=NULL,o3.factors=NULL,fm="pa",rotate="promax",n.
    obs=NA,o1.SMC=TRUE,o2.SMC=TRUE,o3.SMC=TRUE,max.iter=1000,h2.print=FALSE,min.err=.001)
{
require(psych)
nvar <- dim(R)[2]
orth.load <- fa(R,nfactors=o1.factors,n.obs=n.obs,rotate="varimax",fm=fm,SMC=o1.SMC,max.iter=max.
    iter,min.err=min.err)$loadings
h2 <- diag(orth.load %*% t(orth.load))
u2 <- diag(R) - h2
if (is.null(o3.factors)|o3.factors==0) {
o1.fa <- fa(R, nfactors=o1.factors, n.obs = n.obs, rotate = rotate, fm = fm,SMC=o1.SMC,max.iter=
    max.iter,min.err=min.err)
o1.load <- o1.fa$loadings
o1.uniq <- diag(1-o1.fa$communality)
o1.cor <- o1.fa$Phi
o2.fa <- fa(o1.cor, nfactors=o2.factors, fm=fm, rotate=rotate, n.obs=n.obs,SMC=o2.SMC,max.iter=
    max.iter,min.err=min.err)
o2.load <- o2.fa$loadings
o2.uniq <- diag(1-o2.fa$communality)
o2.cor <- o2.fa$Phi
o2.load.sl <- o1.load %*% o2.load
# order 1 loadings
o1.load.sl <- o1.load %*% sqrt(o2.uniq)
if (h2.print==TRUE) {
  # sl matrix
  sl <- cbind(o2.load.sl,o1.load.sl,h2,u2)
  colnames(sl) <- c(paste("o2.f",1:o2.factors,sep=""),paste("o1.f",1:o1.factors,sep=""),"h2","u2
    ")
}
else{
# sl matrix
sl <- cbind(o2.load.sl,o1.load.sl)
  colnames(sl) <- c(paste("o2.f",1:o2.factors,sep=""),paste("o1.f",1:o1.factors,sep=""))
}
}
else{
o1.fa <- fa(R, nfactors=o1.factors, n.obs = n.obs, rotate = rotate, fm = fm,SMC=o1.SMC,max.iter=
    max.iter,min.err=min.err)
o1.load <- o1.fa$loadings
o1.uniq <- diag(1-o1.fa$communality)
o1.cor <- o1.fa$Phi
o2.fa <- fa(o1.cor, nfactors=o2.factors, fm=fm, rotate=rotate, n.obs=n.obs,SMC=o2.SMC,max.iter=
    max.iter,min.err=min.err)
```

```
o2.load <- o2.fa$loadings
o2.uniq <- diag(1-o2.fa$communality)
o2.cor <- o2.fa$Phi
o3.fa <- fa(o2.cor, nfactors=o3.factors, fm=fm, rotate=rotate, n.obs=n.obs,SMC=o3.SMC,max.iter=
    max.iter,min.err=min.err)
o3.load <- o3.fa$loadings
o3.uniq <- diag(1-o3.fa$communality)
o3.load.sl <- o1.load %*% o2.load %*% o3.load
# order 2 loadings
o2.load.sl <- o1.load %*% o2.load %*% sqrt(o3.uniq)
# order 1 loadings
o1.load.sl <- o1.load %*% sqrt(o2.uniq)
if (h2.print==TRUE) {
sl <- cbind(o3.load.sl,o2.load.sl,o1.load.sl,h2,u2)
colnames(sl) <- c(paste("o3.f",1:o3.factors,sep=""),paste("o2.f",1:o2.factors,sep=""),paste("o1.f
    ",1:o1.factors,sep=""),"h2","u2")
}
  else{
sl <- cbind(o3.load.sl,o2.load.sl,o1.load.sl)
colnames(sl) <- c(paste("o3.f",1:o3.factors,sep=""),paste("o2.f",1:o2.factors,sep=""),paste("o1.f
    ",1:o1.factors,sep=""))
  }
}
sl.matrix <- round(sl,3)
return(sl.matrix)
}
```

Here is an example of using the `sl()` function for the data in Table 6.9. Note that we set the initial communality values for both the first- and second-order factors.

```
sl(table6.9.cor, o1.factors=2,o2.factors=1,o3.factors=0,fm="pa",rotate="oblimin",n.obs=100,o1.SMC
    =c(.36,.64,.25,.25),o2.SMC=c(.64,.36))
```

6.5. Bi-Factor Rotation

Jennrich and Bentler (2011,2012) provide two bi-factor rotations, bi-quartimin and bi-geomin. They noted that either rotation should be done multiple times (10 or more) using random starting matrices and then selecting the solution that best optimizes the bi-factor rotation's criterion.

6.5.1. R

Bernaards and Jennrich provide a random start function in their *GPArotation* package: `Random.Start()`.

Here is an example of its use.

```
# unrotated factor solution
load.pa <- fa(table6.9.cor, nfactors=3,rotate="none",fm="pa",n.obs=100, SMC=c(.36,.64,.25,.25))
    $loadings

# bi-quartimin solution using one random starting matrix
library(GPArotation)
GPForth(load.pa, method="bifactor", Tmat = Random.Start(3))
```

One could apply this function multiple times and choose the solution that optimizes the criteria, but James Steiger wrote a helper function, `FindBifactor()`, that will do this a specified number of times for the bi-quartimin rotation.

```
# helper function to run many bi-factor solutions
# reps argument specifies the number of replications
FindBifactor <- function(A,reps){
require(GPArotation)
results <- rep( list(list()), reps )
criterion <- rep(NA,reps)
m <- dim(A)[2]
for(i in 1:reps) {
  x <- GPForth(A,method="bifactor",Tmat = Random.Start(m))
  criterion[i] <- min(x$Table[,2])
  results[[i]] <- x
}
j <- order(criterion)[1]
return(results[[j]])
}
```

Here is an example of its use with 10 random start matrices:

```
FindBifactor(load.pa,10)
```

Mansolf and Reise (2016) recommended using 1,000 random starting values and examining at least the best 10 solutions. Steiger's function does not extract more that one solution or allow for the use of the bi-geomin rotation. We modified his function to allow for extrating multiple unique

solutions and selecting either bifactor rotation criterion. The new function is named `FindBifactor.orth()`.

```
# helper function to run many orthogonal bi-factor solutions using bi-quartimin or bi-geomin and
#      extract an arbitrary number of solutions
# reps argument: specifies the number of random starts to use, default is 10. If the number
#      specified is less than the number of unique sets of results available,
#      then there will be `NULL' results
# rotation argument: specifies is the bifactor rotation to use. Options are
#      "bifactor" for bi-quartimin (default) or "geomin" for bi-geomin
# solutions argument: specifies the number of solutions to return, default is 1
# round argument specifies the precision of the criterion to use for determining the unique
#      values, default is 8 digits
# maxit argument: specifies the maximum number of iterations for each rotation, default is 1000
# seed argument: specifies a random seed for the random starts, default is a random integer
#   between 1 and 10^12

FindBifactor.orth <- function(A,reps=10,rotation="bifactor",solutions=1,round=8,maxit=1000,
    seed=NA){
require(GPArotation)
m <- dim(A)[2]
seed <- ifelse(!is.na(seed),round(seed),ceiling(runif(1, 0, 10^9)))
set.seed(seed)
ran.mat <- replicate(reps,Random.Start(m),simplify=FALSE)
y <- lapply(ran.mat, function(z) GPForth(A,method=rotation,Tmat =z,maxit=maxit))
y <- y[lapply(y, function(z) z$convergence) ==TRUE]
criterion <- sapply(y, function(z) min(z$Table[,2]))
results <- lapply(y, function(z) z$loadings)
criterion <- round(criterion,round)
results <- lapply(results, function(x) round(x,3))
index.val <- 1:length(y)
criterion.index <- data.frame(criterion,index.val)
criterion.index <- criterion.index[order(criterion),]
criterion.index.u <- criterion.index[match(unique(criterion.index$criterion), criterion.
    index$criterion),]
index.keep <- criterion.index.u$index.val[1:solutions]
output <- list(criterion=criterion[index.keep], loadings=results[index.keep])
return(output)
}
```

Here is an example of its use using 1000 random start matrices, bi-geomin rotation, and returning the 10 best solutions.

```
FindBifactor.orth(load.pa,reps=1000,rotation="geomin",solutions=10)
```

If the correlation matrix or number of random starts is large, the `FindBifactor.orth()` function can take a while to complete. One way to shorten this time is use parallel processing if a computer has multiple cores. The details will differ depending on the operating system and parallel processing package used, but will largely just involve replacing the `lapply()` function with its parallel processing counterpart.

6.5.2. Mplus

Mplus will not return results for an EFA of the data in Table 6.9 because "Too many factors were requested for EFA." Nonetheless, the syntax can be generalized for use with other datasets.

Here is an example of using 1000 random start matrices, bi-geomin rotation, and returning the 10 best solutions in Mplus.[1]

```
TITLE: EXPLORATORY BIFACTOR ANALYSIS USING BI-GEOMIN
DATA:
FILE = Table6_9.dat;
TYPE = CORR;
NOBSERVATIONS = 100;
VARIABLE:
NAMES ARE D E F G;
ANALYSIS:
TYPE = EFA 3 3;
ESTIMATOR = ULS;
! number of random starts followed by number of solutions
RSTARTS = 1000 10;
ROTATION=BI-GEOMIN (ORTHOGONAL);
```

To use the bi-quartimin rotation, replace last line in the syntax with

```
ROTATION=BI-CF-QUARTIMAX (ORTHOGONAL);
```

[1] Thank you to Max Mansolf for his help with the Mplus syntax for requesting the random start matrices and returning the best solutions.

6.6. Extended Example

```
hs.cor <- matrix(scan(file="HolzingerSwineford.dat"),ncol=14)
colnames(hs.cor) <- rownames(hs.cor) <- c("T1", "T2", "T3.4", "T6","T28", "T29","T32","T34","T35
    ","T36a","T13","T18","T25b","T77")

#Schmid-Leiman through matrices
library(psych)
hs.fit <- fa(r=hs.cor,nfactors=4,fm="pa",rotate="oblimin")
P01 <- hs.fit$loadings
F <- hs.fit$Phi
hs.so.fit <- fa(r=F,nfactors=1,fm="pa")
P12 <- hs.so.fit$loadings
# uniqueness values
u <-  hs.so.fit$uniqueness
# diagonal matrix of square root of uniqueness
U1 <-diag(sqrt(u))
# g "loadings"
P02 <- P01 %*% P12
# group loadings
P01r <- P01%*%U1

#Schmid-Leiman through schmid() function
library(psych)
schmid(model=as.matrix(hs.cor),nfactors=4,fm="pa",rotate="oblimin")
```

Bi-factor rotation of same data using 50 random starting values

```
hs.cor <- matrix(scan(file="HolzingerSwineford.dat"),ncol=14)
colnames(hs.cor) <- rownames(hs.cor) <- c("T1", "T2", "T3.4", "T6","T28", "T29","T32","T34","T35
    ","T36a","T13","T18","T25b","T77")

library(psych)
bi5.load <- fa(r=as.matrix(hs.cor),nfactors=5,fm="pa", rotate="none")$loadings
FindBifactor(bi5.load,50)
```

Chapter 7: Issues in the Application of Latent Variable Models

There are no computer application examples in this chapter.

Appendix G:
Power of a Test of Poor Fit and Sample Sizes Needed for Powers of .80 and .90

The two R functions in this section were adapted from MacCallum, Browne, and Sugawara's SAS syntax (1996, pp. 148–149).

Copy-and-paste the following function into R to calculate power for a given sample size and df. By default, the null RMSEA is .01, alternative RMSEA is .05, and statistical significance level is .05.

```
#RMSEA power
rmsea.power <- function(df,n,alpha=.05,rmsea0=.1,rmseaa=.05,)
{
  #  alpha is significance level
  #  rmsea0 is RMSEA under null
  #  rmseaa is RMSEA under alternative
  #  ncp0 is ncp under null
  #  ncpa is ncp under alternative
  #
  #Compute power
  if(rmsea0<rmseaa) {
  #critical value
  cval <- qchisq(p=alpha,df=df,ncp=ncp0,lower.tail=FALSE)
  # power
  power <- pchisq(q=cval,df=df,ncp=ncpa,lower.tail=FALSE)
  }
  if(rmsea0>rmseaa) {
  #critical value
  cval <- qchisq(p=alpha,df=df,ncp=ncp0,lower.tail=TRUE)
  # power
    power <- pchisq(q=cval,df=df,ncp=ncpa,lower.tail=TRUE)
  }
return(power)
}
```

```
#example use
rmsea.power(alpha=.05,n=300,rmsea0=.1,rmseaa=.05,df=5)
```

Copy-and-paste the following function into R to calculate sample size for a given power and df. By default, the null RMSEA is .01, alternative RMSEA is .05, statistical significance level is .05, and desired power is .80.

```
rmsea.n <- function(alpha=.05,power=.80,rmsea0=.1,rmseaa=.05,df)
{

  # rmsea0 is RMSEA under null
  # rmseaa is RMSEA under alternative
  # alpha is significance level
  # powd is desired power

powd <- power

#initialize values
powa <- 0.0
n <- 0
#begin loop for finding initial level of n
while (powa<powd) {
  n <- n+100
  ncp0 <- (n-1)*df*rmsea0^2
  ncpa <- (n-1)*df*rmseaa^2
  #compute power
  if(rmsea0<rmseaa) {
      cval <- qchisq(alpha,df,ncp=ncp0,lower.tail=FALSE)
      powa <- pchisq(cval,df,ncp=ncpa,lower.tail=FALSE)
  }
  else {
      cval <- qchisq(alpha,df,ncp=ncp0,lower.tail=TRUE)
      powa <- pchisq(cval,df,ncp=ncpa,lower.tail=TRUE)
  }
}

#begin loop for interval halving
dir <- -1
newn <- n
intv <- 200
powdiff <- powa - powd
while (powdiff>.001) {
  intv <- intv*.5
  newn <- newn + dir*intv*.5
  ncp0 <- (newn-1)*df*rmsea0^2
  ncpa <- (newn-1)*df*rmseaa^2
  #compute power
  if(rmsea0<rmseaa) {
      cval <- qchisq(alpha,df,ncp=ncp0,lower.tail=FALSE)
      powa <- pchisq(cval,df,ncp=ncpa,lower.tail=FALSE)
  }
  else {
      cval <- qchisq(alpha,df,ncp=ncp0,lower.tail=TRUE)
      powa <- pchisq(cval,df,ncp=ncpa,lower.tail=TRUE)
  }
  powdiff <- abs(powa-powd)
  if (powa<powd) {
    dir <- 1
  }
  if (powa>powd) {
    dir <- -1
  }
```

```
}

minn <- newn
return(round(minn))

}
```

```
#example use
rmsea.n(alpha=.05,power=.8,rmsea0=.1,rmseaa=.05,df=60)
```