Study 1: Optimization and Exploratory Factor Analysis

The goal of the first study was to create a short scale with good psychometric properties and to gain initial evidence for the scale's convergent validity.

Table 1

Abbreviations for the Scales and Subscales Used in Syntaxes (Study 1)

Abbreviation	Description
SWSRS	Abbreviation used in German for SSAW
SWSRS_prä	Item of the SSAW scale constructed with regard to the forethought phase
SWSRS_akt	Item of the SSAW scale constructed with regard to the performance phase
SWSRS_post	Item of the SSAW scale constructed with regard to the self-reflection phase
SWSRS_prä_aa/sm/etc.	Lower case letters following the indicator of a phase represent a major category comprised in the distinct phase
SWSRS_PRÄ_22	Forethought subscale of the SSAW scale
SWSRS_AKT_22	Performance subscale of the SSAW scale
SWSRS_POST_22	Self-reflection subscale of the SSAW scale
ASW	Abbreviation for the General Self-Efficacy Scale (GSE)
REG	Abbreviation for the Self-Regulation Scale

SPSS-Syntax for Exploratory Factor Analysis including MAP-Test and Parallel Analysis

Step 1: Factor analysis with the 70 items of the SSAW scale.

```
FACTOR
  /VARIABLES SWSRS_prä_aa_1 SWSRS_prä_aa_2 SWSRS_prä_aa_3 SWSRS_prä_aa_4
SWSRS_prä_aa_5 SWSRS_prä_aa_6 SWSRS_prä_sm_1 R_SWSRS_prä_sm_2
SWSRS_prä_sm_3 SWSRS_prä_sm_4 SWSRS_prä_sm_5 SWSRS_prä_sm_6
SWSRS_prä_sm_7 SWSRS_prä_sm_8 SWSRS_prä_sm_9 SWSRS_prä_sm_10
SWSRS_prä_sm_11 SWSRS_prä_sm_12 SWSRS_akt_sk_1 SWSRS_akt_sk_2
SWSRS_akt_sk_3 SWSRS_akt_sk_4 SWSRS_akt_sk_5 SWSRS_akt_sk_6
SWSRS_akt_sk_7 SWSRS_akt_sk_8 SWSRS_akt_sk_9 SWSRS_akt_sk_10
SWSRS_akt_sk_11 SWSRS_akt_sk_12 SWSRS_akt_sk_13 SWSRS_akt_sk_14
SWSRS_akt_sk_15 SWSRS_akt_sk_16 SWSRS_akt_sk_17 SWSRS_akt_sk_18
SWSRS_akt_sk_19 SWSRS_akt_sk_20 SWSRS_akt_sk_21 SWSRS_akt_sk_22
SWSRS_akt_sk_23 SWSRS_akt_sk_24 SWSRS_akt_sk_25 SWSRS_akt_sk_26
SWSRS_akt_sk_27 SWSRS_akt_sk_28 SWSRS_akt_sb_1 SWSRS_akt_sb_2
SWSRS_akt_sb_3 SWSRS_akt_sb_4 SWSRS_akt_sb_5 SWSRS_akt_sb_6
SWSRS_akt_sb_7 SWSRS_akt_sb_8 SWSRS_akt_sb_9 SWSRS_akt_sb_10
SWSRS_post_sb_1 SWSRS_post_sb_2 SWSRS_post_sb_3 SWSRS_post_sb_4
SWSRS_post_sb_5 SWSRS_post_sb_6 SWSRS_post_sb_7 SWSRS_post_sr_1
SWSRS_post_sr_2 SWSRS_post_sr_3 SWSRS_post_sr_4 SWSRS_post_sr_5
SWSRS_post_sr_6 SWSRS_post_sr_7
  /MISSING LISTWISE
  /ANALYSIS SWSRS_prä_aa_1 SWSRS_prä_aa_2 SWSRS_prä_aa_3 SWSRS_prä_aa_4
SWSRS_prä_aa_5 SWSRS_prä_aa_6 SWSRS_prä_sm_1 R_SWSRS_prä_sm_2
SWSRS_prä_sm_3 SWSRS_prä_sm_4 SWSRS_prä_sm_5 SWSRS_prä_sm_6
SWSRS_prä_sm_7 SWSRS_prä_sm_8 SWSRS_prä_sm_9 SWSRS_prä_sm_10
SWSRS_prä_sm_11 SWSRS_prä_sm_12 SWSRS_akt_sk_1 SWSRS_akt_sk_2
SWSRS_akt_sk_3 SWSRS_akt_sk_4 SWSRS_akt_sk_5 SWSRS_akt_sk_6
SWSRS_akt_sk_7 SWSRS_akt_sk_8 SWSRS_akt_sk_9 SWSRS_akt_sk_10
SWSRS_akt_sk_11 SWSRS_akt_sk_12 SWSRS_akt_sk_13 SWSRS_akt_sk_14
SWSRS_akt_sk_15 SWSRS_akt_sk_16 SWSRS_akt_sk_17 SWSRS_akt_sk_18
SWSRS_akt_sk_19 SWSRS_akt_sk_20 SWSRS_akt_sk_21 SWSRS_akt_sk_22
SWSRS_akt_sk_23 SWSRS_akt_sk_24 SWSRS_akt_sk_25 SWSRS_akt_sk_26
SWSRS_akt_sk_27 SWSRS_akt_sk_28 SWSRS_akt_sb_1 SWSRS_akt_sb_2
SWSRS_akt_sb_3 SWSRS_akt_sb_4 SWSRS_akt_sb_5 SWSRS_akt_sb_6
SWSRS_akt_sb_7 SWSRS_akt_sb_8 SWSRS_akt_sb_9 SWSRS_akt_sb_10
SWSRS_post_sb_1 SWSRS_post_sb_2 SWSRS_post_sb_3 SWSRS_post_sb_4
SWSRS_post_sb_5 SWSRS_post_sb_6 SWSRS_post_sb_7 SWSRS_post_sr_1
SWSRS_post_sr_2 SWSRS_post_sr_3 SWSRS_post_sr_4 SWSRS_post_sr_5
SWSRS_post_sr_6 SWSRS_post_sr_7
  /PRINT UNIVARIATE INITIAL CORRELATION KMO REPR AIC EXTRACTION
ROTATION
  /PLOT EIGEN
  /CRITERIA MINEIGEN(1) ITERATE(100)
  /EXTRACTION PAF
  /CRITERIA ITERATE(100)
  /ROTATION PROMAX(4)
  /METHOD=CORRELATION.
```

```
* Parallel Analysis program[1].

set mxloops=9000 printback=off width=80  seed = 1953125.
matrix.

* enter your specifications here.
compute ncases   = 121.
compute nvars    = 70.
compute ndatsets = 100.
compute percent  = 95.

* Specify the desired kind of parallel analysis, where:
  1 = principal components analysis
  2 = principal axis/common factor analysis.
compute kind = 2 .

****************** End of user specifications. ******************

* principal components analysis.
do if (kind = 1).
compute evals = make(nvars,ndatsets,-9999).
compute nm1 = 1 / (ncases-1).
loop #nds = 1 to ndatsets.
compute x = sqrt(2 * (ln(uniform(ncases,nvars)) * -1) ) &*
        cos(6.283185 * uniform(ncases,nvars) ).
compute vcv = nm1 * (sscp(x) - ((t(csum(x))*csum(x))/ncases)).
compute d = inv(mdiag(sqrt(diag(vcv)))).
compute evals(:,#nds) = eval(d * vcv * d).
end loop.
end if.

* principal axis / common factor analysis with SMCs on the diagonal.
do if (kind = 2).
compute evals = make(nvars,ndatsets,-9999).
compute nm1 = 1 / (ncases-1).
loop #nds = 1 to ndatsets.
compute x = sqrt(2 * (ln(uniform(ncases,nvars)) * -1) ) &*
        cos(6.283185 * uniform(ncases,nvars) ).
compute vcv = nm1 * (sscp(x) - ((t(csum(x))*csum(x))/ncases)).
compute d = inv(mdiag(sqrt(diag(vcv)))).
compute r = d * vcv * d.
compute smc = 1 - (1 &/ diag(inv(r)) ).
call setdiag(r,smc).
compute evals(:,#nds) = eval(r).
end loop.
end if.

* identifying the eigenvalues corresponding to the desired percentile.
compute num = rnd((percent*ndatsets)/100).
```

[1] O'Connor, B. P. (2000). SPSS and SAS programs for determining the number of components using parallel analysis and Velicer's MAP test. *Behavior Research Methods, Instrumentation, and Computers, 32*, 396-402.

```
compute results = { t(1:nvars), t(1:nvars), t(1:nvars) }.
loop #root = 1 to nvars.
compute ranks = rnkorder(evals(#root,:)).
loop #col = 1 to ndatsets.
do if (ranks(1,#col) = num).
compute results(#root,3) = evals(#root,#col).
break.
end if.
end loop.
end loop.
compute results(:,2) = rsum(evals) / ndatsets.

print /title="PARALLEL ANALYSIS:".
do if   (kind = 1).
print /title="Principal Components".
else if (kind = 2).
print /title="Principal Axis / Common Factor Analysis".
end if.
compute specifs = {ncases; nvars; ndatsets; percent}.
print specifs /title="Specifications for this Run:"
 /rlabels="Ncases" "Nvars" "Ndatsets" "Percent".
print results /title="Random Data Eigenvalues"
 /clabels="Root" "Means" "Prcntyle"  /format "f12.6".

do if   (kind = 2).
print / space = 1.
print /title="Compare the random data eigenvalues to the".
print /title="real-data eigenvalues that are obtained from a".
print /title="Common Factor Analysis in which the # of factors".
print /title="extracted equals the # of variables/items, and the".
print /title="number of iterations is fixed at zero;".
print /title="To obtain these real-data values using SPSS, see the".
print /title="sample commands at the end of the parallel.sps program,".
print /title="or use the rawpar.sps program.".
print / space = 1.
print /title="Warning: Parallel analyses of adjusted correlation matrices".
print /title="eg, with SMCs on the diagonal, tend to indicate more factors".
print /title="than warranted (Buja, A., & Eyuboglu, N., 1992, Remarks on parallel".
print /title="analysis. Multivariate Behavioral Research, 27, 509-540.).".
print /title="The eigenvalues for trivial, negligible factors in the real".
print /title="data commonly surpass corresponding random data eigenvalues".
print /title="for the same roots. The eigenvalues from parallel analyses".
print /title="can be used to determine the real data eigenvalues that are".
print /title="beyond chance, but additional procedures should then be used".
print /title="to trim trivial factors.".
print / space = 1.
print /title="Principal components eigenvalues are often used to determine".
print /title="the number of common factors. This is the default in most".
print /title="statistical software packages, and it is the primary practice".
print /title="in the literature. It is also the method used by many factor".
print /title="analysis experts, including Cattell, who often examined".
```

```
print /title="principal components eigenvalues in his scree plots to determine".
print /title="the number of common factors. But others believe this common".
print /title="practice is wrong. Principal components eigenvalues are based".
print /title="on all of the variance in correlation matrices, including both".
print /title="the variance that is shared among variables and the variances".
print /title="that are unique to the variables. In contrast, principal".
print /title="axis eigenvalues are based solely on the shared variance".
print /title="among the variables. The two procedures are qualitatively".
print /title="different. Some therefore claim that the eigenvalues from one".
print /title="extraction method should not be used to determine".
print /title="the number of factors for the other extraction method.".
print /title="The issue remains neglected and unsettled.".

end if.

end matrix.


* Commands for obtaining the necessary real-data eigenvalues for
  principal axis / common factor analysis using SPSS;
  make sure to insert valid filenames/locations, and
  remove the '*' from the first columns.

corr SWSRS_prä_aa_1 to SWSRS_post_sr_7 / matrix out
('C:\Users\Cici\Desktop\SWSRS_EJPA Hauptachsen\Datensatz_SWSRS') / missing =
listwise.
matrix.
MGET /type= corr /file='C:\Users\Cici\Desktop\SWSRS_EJPA
Hauptachsen\Datensatz_SWSRS' .
compute smc = 1 - (1 &/ diag(inv(cr)) ).
call setdiag(cr,smc).
compute evals = eval(cr).
print { t(1:nrow(cr)) , evals }
 /title="Raw Data Eigenvalues"
 /clabels="Root" "Eigen."  /format "f12.6".
end matrix.

* MAP-Test program[2].
* Encoding: UTF-8.

*Mário Basto, José Manuel Pereira, IPCA
*Required: SPSS 21 and R Integration Plugin
*R Packages required: psych, polycor, GPArotation, nFactors, corpcor, ICS, R.utils.
set printback off.
BEGIN PROGRAM R.
# Correlations and descriptives
spsspkg.StartProcedure ("Correlation")
library (polycor)
```

[2] We used the SPSS R-Menu for conducting the MAP-Test. The procedure is described in: Courtney, M. G. R. (2013). Determining the Number of Factors to Retain in EFA: Using the SPSS R-Menu v2.0 to Make More Judicious Estimations. Practical Assessment, Research & Evaluation, 18, 1-14.

```
library (psych)
library (GPArotation)
library (nFactors)
library(corpcor)
library(ICS)
library(R.utils)
# Reading data from SPSS
mdata <- spssdata.GetDataFromSPSS(variables=c("SWSRS_prä_aa_1 SWSRS_prä_aa_2
SWSRS_prä_aa_3 SWSRS_prä_aa_4 SWSRS_prä_aa_5 SWSRS_prä_aa_6
SWSRS_prä_sm_1
R_SWSRS_prä_sm_2 SWSRS_prä_sm_3 SWSRS_prä_sm_4 SWSRS_prä_sm_5
SWSRS_prä_sm_6 SWSRS_prä_sm_7 SWSRS_prä_sm_8 SWSRS_prä_sm_9
SWSRS_prä_sm_10
SWSRS_prä_sm_11 SWSRS_prä_sm_12 SWSRS_akt_sk_1 SWSRS_akt_sk_2
SWSRS_akt_sk_3 SWSRS_akt_sk_4 SWSRS_akt_sk_5 SWSRS_akt_sk_6
SWSRS_akt_sk_7
SWSRS_akt_sk_8 SWSRS_akt_sk_9 SWSRS_akt_sk_10 SWSRS_akt_sk_11
SWSRS_akt_sk_12 SWSRS_akt_sk_13 SWSRS_akt_sk_14 SWSRS_akt_sk_15
SWSRS_akt_sk_16
SWSRS_akt_sk_17 SWSRS_akt_sk_18 SWSRS_akt_sk_19 SWSRS_akt_sk_20
SWSRS_akt_sk_21 SWSRS_akt_sk_22 SWSRS_akt_sk_23 SWSRS_akt_sk_24
SWSRS_akt_sk_25
SWSRS_akt_sk_26 SWSRS_akt_sk_27 SWSRS_akt_sk_28 SWSRS_akt_sb_1
SWSRS_akt_sb_2 SWSRS_akt_sb_3 SWSRS_akt_sb_4 SWSRS_akt_sb_5
SWSRS_akt_sb_6
SWSRS_akt_sb_7 SWSRS_akt_sb_8 SWSRS_akt_sb_9 SWSRS_akt_sb_10
SWSRS_post_sb_1 SWSRS_post_sb_2 SWSRS_post_sb_3 SWSRS_post_sb_4
SWSRS_post_sb_5
SWSRS_post_sb_6 SWSRS_post_sb_7 SWSRS_post_sr_1 SWSRS_post_sr_2
SWSRS_post_sr_3 SWSRS_post_sr_4 SWSRS_post_sr_5 SWSRS_post_sr_6
SWSRS_post_sr_7"))
scal <- spssdictionary.GetDictionaryFromSPSS(variables=c("SWSRS_prä_aa_1
SWSRS_prä_aa_2 SWSRS_prä_aa_3 SWSRS_prä_aa_4 SWSRS_prä_aa_5
SWSRS_prä_aa_6 SWSRS_prä_sm_1
R_SWSRS_prä_sm_2 SWSRS_prä_sm_3 SWSRS_prä_sm_4 SWSRS_prä_sm_5
SWSRS_prä_sm_6 SWSRS_prä_sm_7 SWSRS_prä_sm_8 SWSRS_prä_sm_9
SWSRS_prä_sm_10
SWSRS_prä_sm_11 SWSRS_prä_sm_12 SWSRS_akt_sk_1 SWSRS_akt_sk_2
SWSRS_akt_sk_3 SWSRS_akt_sk_4 SWSRS_akt_sk_5 SWSRS_akt_sk_6
SWSRS_akt_sk_7
SWSRS_akt_sk_8 SWSRS_akt_sk_9 SWSRS_akt_sk_10 SWSRS_akt_sk_11
SWSRS_akt_sk_12 SWSRS_akt_sk_13 SWSRS_akt_sk_14 SWSRS_akt_sk_15
SWSRS_akt_sk_16
SWSRS_akt_sk_17 SWSRS_akt_sk_18 SWSRS_akt_sk_19 SWSRS_akt_sk_20
SWSRS_akt_sk_21 SWSRS_akt_sk_22 SWSRS_akt_sk_23 SWSRS_akt_sk_24
SWSRS_akt_sk_25
SWSRS_akt_sk_26 SWSRS_akt_sk_27 SWSRS_akt_sk_28 SWSRS_akt_sb_1
SWSRS_akt_sb_2 SWSRS_akt_sb_3 SWSRS_akt_sb_4 SWSRS_akt_sb_5
SWSRS_akt_sb_6
```

```
SWSRS_akt_sb_7 SWSRS_akt_sb_8 SWSRS_akt_sb_9 SWSRS_akt_sb_10
SWSRS_post_sb_1 SWSRS_post_sb_2 SWSRS_post_sb_3 SWSRS_post_sb_4
SWSRS_post_sb_5
SWSRS_post_sb_6 SWSRS_post_sb_7 SWSRS_post_sr_1 SWSRS_post_sr_2
SWSRS_post_sr_3 SWSRS_post_sr_4 SWSRS_post_sr_5 SWSRS_post_sr_6
SWSRS_post_sr_7"))
is.na(mdata) <- is.na(mdata)
m1 <- 0; m2 <- 0; m3 <- 0; m4 <- 0; m5<-0; m6<-0
nam <- names(mdata)
# Missing values (pairwise or listwise)
n <- ncol(mdata)
ncase <- nrow(mdata)
ncase2 <- ncase
 # counting listwise cases
cont <- 0
for (i in 1:ncase) {
ind <- 0
j <- 0
while (j<n && ind ==0) {
j <- j+1
if (is.na(mdata[i,j])) {
cont <- cont+1
ind <- 1 } } }
ncase_list <- ncase-cont
if ("complete.obs"=="complete.obs") {     # counting listwise cases
ncase <- ncase_list
junto <- data.frame(ncase, cont)
names(junto) <- c("Number of cases", "Number of cases excluded")
spsspivottable.Display(junto,
title="Listwise deletion", hiderowdimlabel=TRUE, format=6) }
if ("complete.obs"=="pairwise.complete.obs") {
spsspivottable.Display(count.pairwise(mdata),
title="Number of valid cases for each pairwise correlation", format=6) }
da <- mdata
dah <- mdata
for (i in 1:n) { da[,i]<-ordered(mdata[,i])}  # for calculation of polychoric correlations
for (i in 1:n) {
if  ( scal["varMeasurementLevel",i]=="nominal") {dah[,i]<-factor(mdata[,i]) }
else { if  (scal["varMeasurementLevel",i]=="ordinal") { dah[,i]<-ordered(mdata[,i]) } } }
#for calculation of correlations according to their scales
# Multivariate normality tests
if ("n"=="y") {
mn <- mvnorm.skew.test(mdata, na.action = na.omit)
mn2 <- mvnorm.kur.test(mdata, method = "satterthwaite", n.simu = 1000, na.action=na.omit)
mm <- c("Skewness","Kurtosis")
mn3 <- c(mn$statistic, mn2$statistic)
mn4 <- c(mn$p.value, mn2$p.value)
junto <- data.frame(mm, mn3, mn4)
names(junto) <- c("Moment","Test statistic", "p-value")
spsspivottable.Display(junto,
```

```
title="Test of Multivariate Normality based on Skewness and Kurtosis",
hiderowdimlabel=TRUE) }
if ("n"=="y") { co <- corr.test(mdata, use="complete.obs", method="pearson"); m1 <-1
cor1 <- co$r
test <- co$p
spsspivottable.Display(cor1,
title="Pearson correlations")
spsspivottable.Display(test,
title="p-values for Pearson correlations") }
if ("n"=="y") { co <- hetcor(da,  ML = FALSE, std.err = FALSE, pd=TRUE,
use="complete.obs", digits=6); m2 <-1
cor2 <- co$correlations
spsspivottable.Display(cor2,
title="Polychoric correlations - Two Step Estimation") }
if ("n"=="y") { co <- hetcor(da,  ML = TRUE, std.err = FALSE, pd=TRUE,
use="complete.obs", digits=6); m3 <-1
cor3 <- co$correlations
spsspivottable.Display(cor3,
title="Polychoric correlations - Max. Likelihood Estimation") }
if ("n"=="y") { co <- corr.test(mdata, use="complete.obs", method="spearman"); m4 <-1
cor4 <- co$r
test <- co$p
spsspivottable.Display(cor4,
title="Spearman correlations")
spsspivottable.Display(test,
title="p-values for Spearman correlations") }
if ("n"=="y") { co <- hetcor(dah,  ML =FALSE , std.err = FALSE, pd=TRUE,
use="complete.obs", digits=6)
m5 <-1
cor5 <- co$correlations
ctype <- as.data.frame(co$type)
names(ctype) <- nam
het<- rbind(signif(cor5, digits=3), ctype)
row.names(het) <- c(nam,paste("Correlation Type",nam))
spsspivottable.Display(het,
title="Heterogenous correlations (Two Step)") }
if ("n"=="y") { co <- hetcor(dah,  ML =TRUE , std.err = FALSE, pd=TRUE,
use="complete.obs", digits=6)
m6 <-1
cor6 <- co$correlations
ctype <- as.data.frame(co$type)
names(ctype) <- nam
het<- rbind(signif(cor6, digits=3), ctype)
row.names(het) <- c(nam,paste("Correlation Type",nam))
spsspivottable.Display(het,
title="Heterogenous correlations (Max. Likelihood)") }
spsspkg.EndProcedure()
END PROGRAM.
BEGIN PROGRAM R.
# Correlation differences
spsspkg.StartProcedure ("Correlation differences")
```

```
if ("n"=="y")  {
if (m1==0)  {cor1 <- cor(mdata, use="complete.obs", method="pearson"); m1 <-1 }
if (m2==0)  { co <- hetcor(da,  ML = FALSE, std.err = FALSE, pd=TRUE,
use="complete.obs", digits=6); m2 <-1
cor2 <- co$correlations }
d12 <- cor1-cor2
spsspivottable.Display(d12,
title="Pearson - Polychoric (Two Step)") }
if ("n"=="y")  {
if (m1==0)  {cor1 <- cor(mdata, use="complete.obs", method="pearson"); m1 <-1 }
if (m3==0)  { co <- hetcor(da,  ML = TRUE, std.err = FALSE, pd=TRUE,
use="complete.obs", digits=6); m3 <-1
cor3 <- co$correlations }
d13 <- cor1-cor3
spsspivottable.Display(d13,
title="Pearson - Polychoric (Max. Lik.)")  }
if ("n"=="y")  {
if (m2==0)  { co <- hetcor(da,  ML = FALSE, std.err = FALSE, pd=TRUE,
use="complete.obs", digits=6); m2 <-1
cor2 <- co$correlations }
if (m3==0)  { co <- hetcor(da,  ML = TRUE, std.err = FALSE, pd=TRUE,
use="complete.obs", digits=6); m3 <-1
cor3 <- co$correlations }
d23 <- cor2-cor3
spsspivottable.Display(d23,
title="Polychoric (Two Step) - Polychoric (Max. Lik.)")   }
if ("n"=="y")  {
if (m1==0)  { cor1 <- cor(mdata, use="complete.obs", method="pearson"); m1 <-1 }
if (m4==0)  { cor4 <- cor(mdata, use="complete.obs", method="spearman"); m4 <-1 }
d14 <- cor1-cor4
spsspivottable.Display(d14,
title="Pearson - Spearman") }
if ("n"=="y")  {
if (m2==0)  { co <- hetcor(da,  ML = FALSE, std.err = FALSE, pd=TRUE,
use="complete.obs", digits=6); m2 <-1
cor2 <- co$correlations }
if (m4==0)  { cor4 <- cor(mdata, use="complete.obs", method="spearman"); m4 <-1 }
d24 <- cor2-cor4
spsspivottable.Display(d24,
title="Polychoric (Two Step) - Spearman") }
if ("n"=="y")  {
if (m3==0)  { co <- hetcor(da,  ML = TRUE, std.err = FALSE, pd=TRUE,
use="complete.obs", digits=6); m3 <-1
cor3 <- co$correlations }
if (m4==0)  { cor4 <- cor(mdata, use="complete.obs", method="spearman"); m4 <-1 }
d34 <- cor3-cor4
spsspivottable.Display(d34,
title="Polychoric (Max. Lik.) - Spearman") }
spsspkg.EndProcedure()
END PROGRAM.
BEGIN PROGRAM R.
```

```
# Principal Components Analysis extracting PCs from the correlation matrix (function
princomp)
if ("no"=="yes")  {
if (""=="a") {spsspkg.StartProcedure ("Principal Component Analysis - Pearson") }
else { if (""=="b") {spsspkg.StartProcedure ("Principal Component Analysis - Polychoric
(Two Step)") }
else { if (""=="c") {spsspkg.StartProcedure ("Principal Component Analysis - Polychoric
(Max. Likelihood)") }
else { if (""=="d") {spsspkg.StartProcedure ("Principal Component Analysis - Spearman") }
else { if (""=="e") {spsspkg.StartProcedure ("Principal Component Analysis - Covariance") }
else { if (""=="f") {spsspkg.StartProcedure ("Principal Component Analysis - Heterogenous
(Two Step)") }
else { if (""=="g") {spsspkg.StartProcedure ("Principal Component Analysis - Heterogenous
(Max. Likelihood)") }
} } } } } }
if (""=="a")  {
if (m1==0) {cor1 <- cor(mdata, use="complete.obs", method="pearson"); m1 <-1 }
ad <-princomp(cor = FALSE, covmat = cor1)  }
else { if (""=="b")  {
if (m2==0) { co <- hetcor(da,  ML = FALSE, std.err = FALSE, pd=TRUE,
use="complete.obs", digits=6); m2 <-1
cor2 <- co$correlations }
ad <-princomp(cor = FALSE, covmat = cor2)  }
else { if (""=="c")  {
if (m3==0) { co <- hetcor(da,  ML = TRUE, std.err = FALSE, pd=TRUE,
use="complete.obs", digits=6); m3 <-1
cor3 <- co$correlations }
ad <-princomp(cor = FALSE, covmat = cor3)  }
else { if (""=="d") {
if (m4==0) { cor4 <- cor(mdata, use="complete.obs", method="spearman"); m4 <-1 }
ad <-princomp(cor = FALSE, covmat = cor4)  }
else { if (""=="e") { cor <- cov(mdata, use="complete.obs")
ad <-princomp(cor = FALSE, covmat = cor) }
else { if (""=="f")  {
if (m5==0) { co <- hetcor(dah,  ML = FALSE, std.err = FALSE, pd=TRUE,
use="complete.obs", digits=6); m5 <-1
cor5 <- co$correlations }
ad <-princomp(cor = FALSE, covmat = cor5)  }
else { if (""=="g")  {
if (m6==0) { co <- hetcor(dah,  ML = TRUE, std.err = FALSE, pd=TRUE,
use="complete.obs", digits=6); m6 <-1
cor6 <- co$correlations }
ad <-princomp(cor = FALSE, covmat = cor6)  }
 } } } } } }
spsspivottable.Display(ad$loadings[,], title="Eigenvectors")
pro <- ad$sdev^2
loa <- ad$loadings
loa <- loa%*%diag(ad$sdev)
# Sorting or not loadings
if (""=="no") {
spsspivottable.Display(loa[,],
```

```
title="Component Loadings",
collabels=paste ("Comp.",1:n, sep="") ) }
else {
# To correct a little problem when dealing with matrices for some versions of ICLUST.sort
loab <- list()
loab$loadings <- loa
loab$pattern <- loa
p <- ICLUST.sort(loab)
p <- p$sorted[,4:(n+3)]
spsspivottable.Display(p,
title="Sorted Component Loadings", collabels=paste ("Comp.",1:n, sep="")) }
sume <- 0
sum2 <- rep(0,n)
for (i in 1:n) { sume <- sume + pro[i]
sum2[i] = sum2[i] + sume}
junto <- data.frame(ad$sdev,pro,pro/sume*100,sum2/sume*100)
names(junto) <- c("Stdev","Eigenvalues","% of Variance","Cumulative %")
spsspivottable.Display(junto,
title="Variance Explained")
plot(ad,main="Scree Plot",type="lines")
spsspkg.EndProcedure() }
END PROGRAM.
BEGIN PROGRAM R.
#Factor Analysis (functions "principal", "fa", "iterativePrincipalAxis", "PrincipalAxis")
if ("no"=="yes")  {
suprime <-
if ("a"=="a") {
if (m1==0) { cor1 <- cor(mdata, use="complete.obs", method="pearson"); m1 <-1 }
co <- cor1 }
else { if ("a"=="b") {
if (m2==0) { co <- hetcor(da,  ML = FALSE, std.err = FALSE, pd=TRUE,
use="complete.obs", digits=6); m2 <-1
cor2 <- co$correlations }
co <- cor2 }
else { if ("a"=="c") {
if (m3==0) { co <- hetcor(da,  ML = TRUE, std.err = FALSE, pd=TRUE,
use="complete.obs", digits=6); m3 <-1
cor3 <- co$correlations }
co <- cor3 }
else { if ("a"=="d") {
if (m4==0) { cor4 <- cor(mdata, use="complete.obs", method="spearman"); m4 <-1 }
co <- cor4 }
else { if ("a"=="e") {
if (m5==0) { co <- hetcor(dah,  ML = FALSE, std.err = FALSE, pd=TRUE,
use="complete.obs", digits=6); m5 <-1
cor5 <- co$correlations }
co <- cor5 }
else { if ("a"=="f") {
if (m6==0) { co <- hetcor(dah,  ML = TRUE, std.err = FALSE, pd=TRUE,
use="complete.obs", digits=6); m6 <-1
cor6 <- co$correlations }
```

```
co <- cor6 }
 } } } } }
spsspkg.StartProcedure ("Factor Analysis")
pr <-
if (""=="pca" && pr > n) { pr <- n }
if (""!="pca" && pr > (n-1)) { pr <- n-1 }
if (""=="pca") {
ad<-principal(co, nfactors = pr, residuals = FALSE, rotate="none", n.obs=ncase,
scores=FALSE) }
else { if (""=="ipa") {
ad<-principal(co, nfactors = pr, rotate="none", n.obs=ncase, scores=FALSE)
ad3 <- iterativePrincipalAxis(co, nFactors=pr, communalities="", iterations=200,
tolerance=1e-6)
ad$loadings <- ad3$loadings }
else { if (""=="nipa") {
ad<-principal(co, nfactors = pr, rotate="none", n.obs=ncase, scores=FALSE)
ad3 <- principalAxis(co, nFactors=pr, communalities="")
ad$loadings <- ad3$loadings }
else {ad <- fa(co, nfactors=pr, residuals = FALSE, rotate = "none", n.obs=ncase_list,
SMC=TRUE, min.err = 1e-6, digits =8, max.iter=200, symmetric=TRUE, warnings=TRUE,
fm="")
ad$loadings <- ad$loadings[,order(colnames(ad$loadings))]
 } } }
if ("a"=="a") { mat <- "Pearson" }
else { if ("a"=="b") { mat <- "Polychoric (Two Step)" }
else { if ("a"=="c") { mat <- "Polychoric (Max. Lik.)" }
else { if ("a"=="d") { mat <- "Spearman" }
else { if ("a"=="e") { mat <- "Heterogenous (Two Step)" }
else { if ("a"=="f") { mat <- "Heterogenous (Max. Lik.)" }
 } } } } }
if ("complete.obs"=="pairwise.complete.obs") { caso <- "pairwise"}
if ("complete.obs"=="complete.obs") { caso <- "listwise"}
junto <- data.frame(mat, "", caso)
names(junto) <- c("Correlation Matrix", "Factor Extraction", "Missing values")
spsspivottable.Display(junto,
title="Factor Analysis", hiderowdimlabel=TRUE)
ad$loadings <- as.matrix(ad$loadings)
row.names(ad$loadings) <- nam
load <- ad$loadings
ConFac <- load
if (""=="ipa" || ""=="nipa") {
sum2 <- rep(0,n)
for(i in 1:pr) { sum2 <- load[,i]^2 + sum2 }
ad$communality <- sum2
}
extracted <- ad$communality
b <- data.frame(extracted, row.names=nam)
spsspivottable.Display(b, title="Communalities")
# Sorting or not loadings
if (""=="no") {
supre <- load
```

```
for (j in 1:pr) {
for (i in 1:nrow(load)) {
if (abs(load[i,j]) < suprime) { supre[i,j] <- NA } } }
spsspivottable.Display(supre[,],
title="Factor Loadings (unrotated)", collabels=paste ("F",1:pr, sep="")) }
else {
ad <- fa.sort(ad)
load2 <- ad$loadings
supre <- load2
for (j in 1:pr) {
for (i in 1:nrow(load2)) {
if (abs(load2[i,j]) < suprime) { supre[i,j] <- NA } } }
spsspivottable.Display(supre[,],
title="Sorted Factor Loadings (unrotated)", collabels=paste ("F",1:pr, sep="") ) }
# Initial Eigenvalues
if (""=="pca" || ""=="ipa" || ""=="nipa") {
Eigenvalues <- ad$values }
else { Eigenvalues <- ad$e.values }
# Scree plot
if ("n"=="y") {
plotuScree(Eigenvalues, ylab = "Eigenvalues",
xlab = "Components",
main = "Scree Plot") }
sume  <- 0
sum2 <- rep(0,n)
for (i in 1:n) { sume <- sume  + Eigenvalues[i]
sum2[i] = sum2[i] + sume }
junto <- data.frame(Eigenvalues,Eigenvalues/sume*100,sum2/sume*100)
names(junto) <- c("Eigenvalues","% of Variance","Cumulative %")
spsspivottable.Display(junto,
title=" Variance Explained (initial)")
# Calculation of Sums of Squared Loadings
pro <- rep(0,pr)
for (j in 1:pr) { sum3 <- 0
for (i in 1:n) { sum3<- sum3 + load[i,j]^2 }
pro[j] <- sum3 }
sum3 <- 0
sum2 <- rep(0,pr)
for (i in 1:pr) { sum3<- sum3 + pro[i]
sum2[i] = sum2[i] + sum3 }
junto <- data.frame(pro,pro/sume*100,sum2/sume*100)
names(junto) <- c("Sums of Squared Loadings","% of Variance","Cumulative %")
spsspivottable.Display(junto,
title="Variance Explained (extracted)")
# Factor plot unrotated
if ("n"=="y") {
l1 <-
l2 <-
ll1 <- min(l1,l2); ll2 <- max(l1,l2)
if (ll1 < ll2 && ll2 < (pr+1)) {
x <-matrix(c(load[,ll1],load[,ll2]),ncol=2)
```

```
x1 <- paste("Factor", ll1, sep=" ")
y1 <- paste("Factor", ll2, sep=" ")
factor.plot(x, cut = , labels=nam, xlim = c(-1, 1), ylim = c(-1, 1), xlab=x1, ylab=y1, title =
"Factor Plot (initial solution)")  } }
# Factor diagram unrotated
if ("n"=="y") {
fa.diagram(ad, sort=, labels=NULL, cut=, simple=, errors=TRUE, digits=2, e.size=0.05,
rsize=0.15, side=2, main="Factor Diagram (initial solution)", cex=NULL) }
# Rotations
if (""!="none" || ""=="yes") {
if (""=="varimax") {ad2 <- Varimax(load, Tmat=diag(ncol(load)), normalize=, eps=1e-5,
maxit=500) }
else { if (""=="quartimax") {ad2 <- quartimax(load, Tmat=diag(ncol(load)), normalize=,
eps=1e-5, maxit=500) }
else { if (""=="quartimin") {ad2 <- quartimin(load, Tmat=diag(ncol(load)), normalize=,
eps=1e-5, maxit=500) }
else { if (""=="equamax") {ad2 <- cfT(load, Tmat=diag(ncol(load)),
kappa=pr/(2*nrow(load)), normalize=, eps=1e-5, maxit=500) }
else { if (""=="parsimax") {ad2 <- cfT(load, Tmat=diag(ncol(load)), kappa=(pr-
1)/(nrow(load)+pr-2), normalize=, eps=1e-5, maxit=500) }
else { if (""=="factor.parsimony") {ad2 <- cfT(load, Tmat=diag(ncol(load)), kappa=1,
normalize=, eps=1e-5, maxit=500) }
else { if (""=="biquartimin") { ad2 <- oblimin(load, Tmat=diag(ncol(load)), gam=0.5,
normalize=, eps=1e-5, maxit=500) }
else { if (""=="covarimin") { ad2 <- oblimin(load, Tmat=diag(ncol(load)), gam=1,
normalize=, eps=1e-5, maxit=500) }
else { if (""=="oblimax") {ad2 <- oblimax(load, Tmat=diag(ncol(load)), normalize=, eps=1e-
5, maxit=500) }
else { if (""=="entropy") {
kk <-
if (""=="") { kk <- ncol(load)*nrow(load) - nrow(load) }
ad2 <- entropy(load, Tmat=diag(ncol(load)), normalize=, eps=1e-5, maxit=500) }
else { if (""=="simplimax") {ad2 <- simplimax(load, Tmat=diag(ncol(load)), k=, normalize=,
eps=1e-5, maxit=500) }
else { if (""=="bentlerT") {ad2 <- bentlerT(load, Tmat=diag(ncol(load)), normalize=, eps=1e-
5, maxit=500) }
else { if (""=="bentlerQ") {ad2 <- bentlerQ(load, Tmat=diag(ncol(load)), normalize=,
eps=1e-5, maxit=500) }
else { if (""=="tandemI") {ad2 <- tandemI(load, Tmat=diag(ncol(load)), normalize=, eps=1e-
5, maxit=500) }
else { if (""=="tandemII") {ad2 <- tandemII(load, Tmat=diag(ncol(load)), normalize=,
eps=1e-5, maxit=500) }
else { if (""=="geominT") {ad2 <- geominT(load, Tmat=diag(ncol(load)), delta=0.01,
normalize=, eps=1e-5, maxit=500) }
else { if (""=="geominQ") {ad2 <- geominQ(load, Tmat=diag(ncol(load)), delta=0.01,
normalize=, eps=1e-5, maxit=500) }
else { if (""=="infomaxT") {ad2 <- infomaxT(load, Tmat=diag(ncol(load)), normalize=,
eps=1e-5, maxit=500) }
else { if (""=="infomaxQ") {ad2 <- infomaxQ(load, Tmat=diag(ncol(load)), normalize=,
eps=1e-5, maxit=500) }
```

```
else { if (""=="mccammon") {ad2 <- mccammon(load, Tmat=diag(ncol(load)), normalize=,
eps=1e-5, maxit=500) }
} } } } } } } } } } } } } } } } } } } }
prom<-0
# Promax rotation (adapted from Promax function from psych package)
if (""!="none") {  promai <- ad2$orthogonal } else { promai <- FALSE }
if ((promai) || (""=="none")) {
if (""=="yes") {
m <-
if (ncol(load) < 2) { return(load)}
dn <- dimnames(load)
if (""!="none") { load <- ad2$loadings }
Q <- load * abs(load)^(m - 1)
U <- lm.fit(load, Q)$coefficients
d <- diag(solve(t(U) %*% U))
U <- U %*% diag(sqrt(d))
dimnames(U) <- NULL
z <- load %*% U
if (""!="none") {U <- ad2$Th %*% U}
ui <- solve(U)
Phi <- ui %*% t(ui)
dimnames(z) <- dn
prom <-1
ad2<- list(loadings = z, Th = U%*%Phi, Phi = Phi, orthogonal = FALSE, method ="promax",
initial="") } }
load <- ad2$loadings   # if oblique rotation, this is the pattern matrix
 # Sorting or not loadings after rotation
if (""=="no") {
if (ad2$orthogonal == FALSE) {
supre <- load
for (j in 1:pr) {
for (i in 1:nrow(load)) {
if (abs(load[i,j]) < suprime) { supre[i,j] <- NA } } }
spsspivottable.Display(supre[,], title="Pattern Matrix",
collabels=paste ("F",1:pr, sep="") )
p <- as.matrix(load[,])%*%ad2$Phi
supre <- p
for (j in 1:pr) {
for (i in 1:nrow(p)) {
if (abs(p[i,j]) < suprime) { supre[i,j] <- NA } } }
spsspivottable.Display(supre[,],
title="Structure Matrix") }
else {
supre <- load
for (j in 1:pr) {
for (i in 1:nrow(load)) {
if (abs(load[i,j]) < suprime) { supre[i,j] <- NA } } }
spsspivottable.Display(supre[,],
title="Rotated Factor Loadings",
collabels=paste ("F",1:pr, sep="") ) } }
else {
```

```
ad$loadings <- ad2$loadings
ad <- fa.sort(ad)
load2 <- ad$loadings
supre <- load2
for (j in 1:pr) {
for (i in 1:nrow(load2)) {
if (abs(load2[i,j]) < suprime) { supre[i,j] <- NA } } }
if ( ! ad2$orthogonal) { spsspivottable.Display(supre[,],
title="Sorted Pattern Matrix",
collabels=paste ("F",1:pr, sep="") )
p <- as.matrix(load2[,])%*%ad2$Phi
supre <- p
for (j in 1:pr) {
for (i in 1:nrow(p)) {
if (abs(p[i,j]) < suprime) { supre[i,j] <- NA } } }
spsspivottable.Display(supre[,],
title="Sorted Structure Matrix",
collabels=paste ("F",1:pr, sep="") ) }
else {
supre <- load2
for (j in 1:pr) {
for (i in 1:nrow(load2)) {
if (abs(load2[i,j]) < suprime) { supre[i,j] <- NA } } }
spsspivottable.Display(supre[,],
title="Sorted Rotated Factor Loadings",
collabels=paste ("F",1:pr, sep="") ) } }
# Calculation of Sums of Squared Loadings after rotation
if (ad2$orthogonal) {
pro <- rep(0,pr)
for (j in 1:pr) { sum3 <- 0
for (i in 1:n) { sum3<- sum3 + load[i,j]^2 }
pro[j] <- sum3 }
sum3 <- 0
sum2 <- rep(0,pr)
for (i in 1:pr) { sum3<- sum3 + pro[i]
sum2[i] = sum2[i] + sum3 }
junto <- data.frame(pro,pro/sume*100,sum2/sume*100)
names(junto) <- c("Sums of Squared Loadings","% of Variance","Cumulative %")
spsspivottable.Display(junto,
title="Rotated Variance Explained (extracted)") }
else {
pro <- rep(0,pr)
for (j in 1:pr) { sum3 <- 0
for (i in 1:n) { sum3<- sum3 + p[i,j]^2 }
pro[j] <- sum3 }
junto <- data.frame(pro)
names(junto) <- c("Sums of Squared Loadings")
spsspivottable.Display(junto,
title="Rotation Sums of Squared Loadings from Structure Matrix") }
spsspivottable.Display(ad2$Th,
title="Rotation Matrix",
```

```
collabels=paste ("F",1:pr, sep=""),
rowlabels=paste ("F",1:pr, sep="") )
spsspivottable.Display(ad2$Phi,
title="Correlation matrix of rotated factors",
collabels=paste ("F",1:pr, sep=""),
rowlabels=paste ("F",1:pr, sep="") )
if (prom==0) {
junto <- data.frame(ad2$method,ad2$orthogonal,ad2$convergence)
names(junto) <- c("method","orthogonal","convergence")
spsspivottable.Display(junto,
title="Rotation", hiderowdimlabel=TRUE) }
else { junto <- data.frame(ad2$method,ad2$orthogonal, ad2$initial)
names(junto) <- c("method","orthogonal","initial rotation")
spsspivottable.Display(junto,
title="Rotation", hiderowdimlabel=TRUE) }
# Factor plot after rotation
if ("n"=="y") {
l1 <-
l2 <-
ll1 <- min(l1,l2); ll2 <- max(l1,l2)
if (ll1 < ll2 && ll2 < (pr+1)) {
x <-matrix(c(load[,ll1],load[,ll2]), ncol=2)
x1 <- paste("Factor", ll1, sep=" ")
y1 <- paste("Factor", ll2, sep=" ")
factor.plot(x, cut = , labels=nam,  xlim = c(-1, 1), ylim = c(-1, 1), xlab=x1, ylab=y1, title =
"Factor Plot (rotated solution)")  } }
# Factor diagram after rotation
if ("n"=="y") {
ad$loadings <- ad2$loadings
colnames(ad$loadings) <- paste("F", 1:pr, sep = "")
fa.diagram(ad, sort=, labels=NULL, cut=, simple=, errors=TRUE, digits=2, e.size=0.05,
rsize=0.15, side=2, main="Factor Diagram (rotated solution)", cex=NULL) }
 }   #end of rotation
# KMO
if ("n"=="y") {
g <- diag(0,n)
par <- cor2pcor(co)
cco <- co
cco[upper.tri(cco, TRUE)] <- g[upper.tri(cco, TRUE)]
par[upper.tri(par, TRUE)] <- g[upper.tri(par, TRUE)]
cco <- sum(cco^2)
kmo <- cco/(sum(par^2)+cco)
KMO <- c(kmo)
b <- data.frame(KMO)
spsspivottable.Display(b,
title="KMO - Kaiser-Meyer-Olkin measure of sampling adequacy",
hiderowdimlabel=TRUE) }
# MSA
if ("n"=="y") {
par <- cor2pcor(co)
ms <- rep(0,n)
```

```
for (i in 1:n) { ms[i] <- ( sum(co[i,]^2)-1 )  / (sum(co[i,]^2)+sum(par[i,]^2)-2) }
MSA <- c(ms)
b <- data.frame(MSA, row.names=nam)
spsspivottable.Display(b,
title="MSA - Measures of sampling adequacy") }
# Residual matrix for extraction using NFactors package
if (""=="ipa" || ""=="nipa") {
res <- diag(0,n)
for (i in 2:n) {
for (j in 1:(i-1)) {
res [i,j] <- co[i,j] - sum(ConFac[i,]*ConFac[j,]) } }
junto <- res + t(res) + diag(1 - ad$communality)
row.names(junto) <- nam
names(junto) <- nam
ad$residual <- junto
junto <- data.frame(junto) }
# GFI and AGFI
if ("n"=="y") {
s <- ad$residual
som <- sum(diag(s%*%s))
som2 <- sum(diag(co%*%co))
som <- 1-som/som2
b <- data.frame(som)
names(b) <- c("GFI (ULS)")
spsspivottable.Display(b,
title="GFI (Goodness of Fit)", hiderowdimlabel=TRUE) }
if ("n"=="y") {
aus <- matrix(0,n,n)
for (i in 1:(n-1)) {
for (j in (i+1):n) {
aus[i,j] <- sum(ConFac[i,]*ConFac[j,]) } }
aus2 <- t(aus)
diag(aus2) <- 1
aus <- aus + aus2
som <- sum(diag((solve(aus)%*%co-diag(n))%*%(solve(aus)%*%co-diag(n))))
som2 <- sum(diag(solve(aus)%*%co%*%solve(aus)%*%co))
som <- 1-som/som2
b <- data.frame(som)
names(b) <- c("GFI (ML)")
spsspivottable.Display(b,
title="GFI (Goodness of Fit)", hiderowdimlabel=TRUE) }
# RMSR
if ("n"=="y") {
g <- diag(0,n)
s <- ad$residual
s[upper.tri(s, TRUE)] <- g[upper.tri(s, TRUE)]
rmsr <- sqrt(2*sum(s^2)/(n*(n-1)))
RMSR <- c(rmsr)
b <- data.frame(RMSR)
spsspivottable.Display(b,
title="RMSR (Root Mean Square Residual)",
```

```
hiderowdimlabel=TRUE) }
# Root Mean Square Partials Correlations Controlling Factors
if ("n"=="y") {
cc <- co-(ConFac%*%t(ConFac))
ca <- sqrt(diag(cc))
d <- diag(1/ca)
ea <- d%*%cc%*%d   #partial correlations controlling factors
rmsp <- sqrt((sum(ea^2)-n)/(n*(n-1)))
RMSP <- c(rmsp)
b <- data.frame(RMSP)
spsspivottable.Display(b,
title="Root mean square partial correlations controlling factors",
hiderowdimlabel=TRUE) }
# Partial Correlations controlling all other variables
if ("n"=="y") {
par <- cor2pcor(co)
row.names(par) <- nam
par <- data.frame(par)
names(par) <- nam
spsspivottable.Display(par,
title="Partial correlations controlling all other variables") }
# Partial Correlations controlling Factors
if ("n"=="y") {
cc <- co-(ConFac%*%t(ConFac))
ca <- sqrt(diag(cc))
d <- diag(1/ca)
ea <- d%*%cc%*%d   #partial correlations controlling factors
row.names(ea) <- nam
ea <- data.frame(ea)
names(ea) <- nam
spsspivottable.Display(ea,
title="Partial correlations controlling factors") }
# Residual correlations
if (FALSE==TRUE) {
spsspivottable.Display(ad$residual,
title="Residual correlations with uniqueness on the diagonal (computed between observed and
reproduced correlations)")
cont <- 0
s <- ad$residual
for (i in 2:n) { for (j in 1:(i-1)) { if (abs(s[i,j])> 0.05) { cont <- cont+1 } } }
p <- 2*cont/(n*(n-1))*100
table <- spss.BasePivotTable("Fit of the model to the correlation matrix","OMS")
rowdim <- BasePivotTable.Append(table,Dimension.Place.row,"")
coldim <- BasePivotTable.Append(table,Dimension.Place.column,"Residual fit values")
row1 <- spss.CellText.String("Values")
col1 <- spss.CellText.String("Residuals > 0.05")
col2 <- spss.CellText.String("% residuals > 0.05")
BasePivotTable.SetCategories(table,rowdim,list(row1))
BasePivotTable.SetCategories(table,coldim,list(col1,col2))
BasePivotTable.SetCellsByColumn(table,col1,list(spss.CellText.Number(cont,6)))
BasePivotTable.SetCellsByColumn(table,col2,list(spss.CellText.Number(p))) }
```

```
# Determinant
if ("n"=="y") {
b <- determinant(co, logarithm = FALSE)
junto <- data.frame(b$modulus)
names(junto) <- c("Determinant")
spsspivottable.Display(junto,
title="Determinant (modulus)",
hiderowdimlabel=TRUE) }
# Tests to correlation matrix
if ("n"=="y") {
bar <- cortest.bartlett(co, n = ncase)
bar2 <- cortest.normal(co,n1=ncase, fisher = FALSE)
bar3 <- cortest.jennrich(co, diag(rep(1,n)), n1 = ncase, n2=ncase)
table <- spss.BasePivotTable("Tests of whether a correlation matrix is an identity
matrix","OMS")
rowdim <- BasePivotTable.Append(table,Dimension.Place.row,"")
coldim <- BasePivotTable.Append(table,Dimension.Place.column,"Chisquare tests")
row1 <- spss.CellText.String("Bartlett's Test")
row2 <- spss.CellText.String("Steiger Test")
row3 <- spss.CellText.String("Jennrich Test")
col1 <- spss.CellText.String("Chisquare")
col2 <- spss.CellText.String("Degrees of freedom")
col3<- spss.CellText.String("p-value")
BasePivotTable.SetCategories(table,rowdim,list(row1,row2, row3))
BasePivotTable.SetCategories(table,coldim,list(col1,col2,col3))
BasePivotTable.SetCellsByColumn(table,col1,list(spss.CellText.Number(bar$chisq),spss.Cell
Text.Number(bar2$chi2),spss.CellText.Number(bar3$chi2)))
BasePivotTable.SetCellsByColumn(table,col2,list(spss.CellText.Number(bar$df,6),spss.CellT
ext.Number(bar2$df,6),spss.CellText.Number(bar2$df,6)))
BasePivotTable.SetCellsByColumn(table,col3,list(spss.CellText.Number(bar$p.value),spss.C
ellText.Number(bar2$prob),spss.CellText.Number(bar3$prob))) }
# Chisquare test for Maximum Likelihood
if (""=="ml") {
table <- spss.BasePivotTable("Chisquare test for Maximum Likelihood procedure","OMS")
rowdim <- BasePivotTable.Append(table,Dimension.Place.row,"")
coldim <- BasePivotTable.Append(table,Dimension.Place.column,"Chisquare test")
row1 <- spss.CellText.String("Test")
col1 <- spss.CellText.String("Chisquare")
col2 <- spss.CellText.String("Degrees of freedom")
col3<- spss.CellText.String("p-value")
BasePivotTable.SetCategories(table,rowdim,list(row1))
BasePivotTable.SetCategories(table,coldim,list(col1,col2,col3))
BasePivotTable.SetCellsByColumn(table,col1,list(spss.CellText.Number(ad$STATISTIC)))
BasePivotTable.SetCellsByColumn(table,col2,list(spss.CellText.Number(ad$dof,6)))
if (is.numeric(ad$PVAL)) {
BasePivotTable.SetCellsByColumn(table,col3,list(spss.CellText.Number(ad$PVAL))) }
else { BasePivotTable.SetCellsByColumn(table,col3,list(spss.CellText.String(ad$PVAL))) }
}
spsspkg.EndProcedure() }
END PROGRAM.
BEGIN PROGRAM R.
```

```
# Number of Factors
if ("no"=="yes" || "y"=="y" || "n"=="y") {
if ("e"=="a") {
if (m1==0) { cor1 <- cor(mdata, use="complete.obs", method="pearson"); m1 <-1 }
co <- cor1 }
if ("e"=="b") {
if (m2==0) { co <- hetcor(da,  ML = FALSE, std.err = FALSE, pd=TRUE,
use="complete.obs", digits=6); m2 <-1
cor2 <- co$correlations }
co <- cor2 }
if ("e"=="c") {
if (m3==0) { co <- hetcor(da,  ML = TRUE, std.err = FALSE, pd=TRUE,
use="complete.obs", digits=6); m3 <-1
cor3 <- co$correlations }
co <- cor3 }
if ("e"=="d") {
if (m4==0) { cor4 <- cor(mdata, use="complete.obs", method="spearman"); m4 <-1 }
co <- cor4 }
if ("e"=="e") {
if (m5==0) { co <- hetcor(dah,  ML = FALSE, std.err = FALSE, pd=TRUE,
use="complete.obs", digits=6); m5 <-1
cor5 <- co$correlations }
co <- cor5 }
if ("e"=="f") {
if (m6==0) { co <- hetcor(dah,  ML = TRUE, std.err = FALSE, pd=TRUE,
use="complete.obs", digits=6); m6 <-1
cor6 <- co$correlations }
co <- cor6 }
spsspkg.StartProcedure ("Number of Components/Factors")
if ("e"=="a") { mat <- "Pearson" }
if ("e"=="b") { mat <- "Polychoric (Two Step)" }
if ("e"=="c") { mat <- "Polychoric (Max. Lik.)" }
if ("e"=="d") { mat <- "Spearman" }
if ("e"=="e") { mat <- "Heterogenous (Two Step)" }
if ("e"=="f") { mat <- "Heterogenous (Max. Lik.)" }
if ("complete.obs"=="pairwise.complete.obs") { caso <- "pairwise"}
if ("complete.obs"=="complete.obs") { caso <- "listwise"}
junto <- data.frame(mat, caso)
names(junto) <- c("Correlation Matrix", "Missing values")
spsspivottable.Display(junto,
title="Number of Components/Factors estimated for the following Correlation Matrix",
hiderowdimlabel=TRUE) }
# Cattel Scree Test
if ("no"=="yes" && n >= 3) {
if (""!="") { set.seed() }
repi <-
evpea <- matrix(NA, ncol = n, nrow = repi)
evpea_b <- matrix(NA, ncol = n, nrow = repi)
# Eigenvalues from components or factors
if (""=="components") {
evt <- eigen(co, only.values = TRUE) }
```

```
else { evt <- eigen(co - ginv(diag(diag(ginv(co)))), only.values=TRUE) }
# Simulations
quant <- function(x, sprobs = sprobs) { return(as.vector(quantile(x, probs = c()))) }
# standardized normal
if ("" == "normal") {
co_used <- "Pearson"
for (k in 1:repi) {
y <- mvrnorm(ncase, rep(0, n), diag(1,n), empirical = FALSE)
corY <- cor(y, use="complete.obs", method="pearson")
if (""=="factors") { corY <- corY - ginv(diag(diag(ginv(corY)))) }
ev <- eigen(corY, only.values = TRUE)
evpea[k, ] <- ev$values }
mevpea <- sapply(as.data.frame(evpea), mean)
qevpea <- sapply(as.data.frame(evpea), quant)
ap <- list(eigen = data.frame(mevpea, qevpea)) }
# permutation
if ("" == "permutation") {
sdata <- matrix (0,ncol = ncol(mdata), nrow =ncase_list)
if (""=="y") {
mdata2 <- as.matrix(mdata)
lcor <- 0
for (i in 1:ncase2) {
ind <- 0
j <- 0
while (j<n && ind ==0) {
j <- j+1
if (is.na(mdata[i,j])) { ind <- 1 } }
if (ind==0) { lcor <- lcor+1; sdata[lcor,] <- mdata2[i,] } } }
else { sdata <- mdata }
for (k in 1:repi) {
perm <- apply(sdata[,2:n], 2, sample, replace = )
perm <- data.frame(sdata[,1],perm)
if (""=="a") { corY <- cor(perm, use="complete.obs", method="pearson"); co_used <-
"Pearson" }
else { if (""=="b") {
co_used <- "Polychoric (Two Step)"
for (i in 1:n) { perm[,i]<-ordered(perm[,i]) }
corY <-  hetcor(perm,  ML = FALSE, std.err = FALSE, pd=TRUE, use="complete.obs",
digits=4)
corY <- corY$correlations }
else { if (""=="c") {
co_used <- "Polychoric (Max. Likelihood)"
for (i in 1:n) { perm[,i]<-ordered(perm[,i]) }
corY <-  hetcor(perm,  ML = TRUE, std.err = FALSE, pd=TRUE, use="complete.obs",
digits=4)
corY <- corY$correlations }
else { if (""=="e") {
co_used <- "Heterogeneous (Two Step)"
for (i in 1:n) {
if  ( scal["varMeasurementLevel",i]=="nominal") { perm[,i]<-factor(perm[,i]) }
else { if  (scal["varMeasurementLevel",i]=="ordinal") { perm[,i]<-ordered(perm[,i]) } } }
```

```
corY <-  hetcor(perm,  ML = FALSE, std.err = FALSE, pd=TRUE, use="complete.obs",
digits=4)
corY <- corY$correlations }
else { if (""=="f") {
co_used <- "Heterogeneous (Max. Likelihood)"
for (i in 1:n) {
if  ( scal["varMeasurementLevel",i]=="nominal") { perm[,i]<-factor(perm[,i]) }
else { if  (scal["varMeasurementLevel",i]=="ordinal") { perm[,i]<-ordered(perm[,i]) } } }
corY <-  hetcor(perm,  ML = TRUE, std.err = FALSE, pd=TRUE, use="complete.obs",
digits=4)
corY <- corY$correlations }
else { corY <- cor(perm, use="complete.obs", method="spearman"); co_used <- "Spearman"
} } } } }
if (""=="factors") { corY <- corY - ginv(diag(diag(ginv(corY)))) }
ev <- eigen(corY, only.values = TRUE)
evpea[k, ] <- ev$values }
mevpea <- sapply(as.data.frame(evpea), mean)
qevpea <- sapply(as.data.frame(evpea), quant)
ap <- list(eigen = data.frame(mevpea, qevpea) ) }
# parallel analysis and scree
nS <- nScree(evt$values, aparallel=)
s <- ""
if ( s=="ap$eigen$mevpea") { ss <- "mean" } else { ss <- "quantile" }
if (""=="factors") { criteria <- mean(evt$values)
nS$Components$nkaiser <- sum(evt$values >= rep(criteria, n))
p.vec <- which(evt$values >= ap$eigen$qevpea)
nS$Components$nparallel <- sum(p.vec == (1:length(p.vec))) }
junto <- data.frame(repi, "", "", ss, mat, co_used)
names(junto) <- c("Number of Samples","Model","Data","Measure",
"Correlation matrix","Matrix for simulations")
spsspivottable.Display(junto,
title="Parameters for Parallel Analysis",
hiderowdimlabel=TRUE, format=6)
junto <- data.frame(paste("",1:n, rep=""), ap$eigen)
names(junto) <- c("Order", "Mean", "Quantile selected")
spsspivottable.Display(junto,
title="Distribution of the eigenvalues computed",
hiderowdimlabel=TRUE)
junto <- data.frame(nS$Components)
names(junto) <- c("Optimal coordinates","Acceleration factor",
"Parallel analysis","Kaiser rule")
spsspivottable.Display(junto,
title="Number of components/factors to retain according to different rules",
hiderowdimlabel=TRUE, format=6)
junto <- data.frame(nS$Analysis)
names(junto) <- c("Eigenvalues","Proportion of variance", "Cumulative", "Parallel analysis",
"Predicted eigenvalues", "OC", "Acceleration factor", "AF")
spsspivottable.Display(junto,
title="Data linked to the different rules")
if (""=="components") {
if (""=="normal") {
```

```
plotnScree(nS, ylab = "Eigenvalues", xlab = "Components",
main = "Parallel analysis on random uncorrelated standardized normal") }
if (""=="permutation") {
plotnScree(nS, ylab = "Eigenvalues", xlab = "Components",
main = "Parallel analysis on data permutation") } }
else {
if (""=="normal") {
plotnScree(nS, ylab = "Eigenvalues", xlab = "Factors",
main = "Parallel analysis on random uncorrelated standardized normal") }
if (""=="permutation") {
plotnScree(nS, ylab = "Eigenvalues", xlab = "Factors",
main = "Parallel analysis on data permutation") } } }
# Velicer MAP
if ("y"=="y") {
ev <- eigen(co, symmetric=TRUE)
load2 <- ev$vectors%*%sqrt(diag(ev$values))
f <- rep(0,(n-1)); fq <- rep(0,(n-1))
som <- sum(co^2); somq <- sum(co^4)
m <- n*(n-1); f[1] <- (som-n)/m; fq[1] <- (somq-n)/m
for (i in 1:(n-2)) { b <- load2[,1:i]; cc <- co-(b%*%t(b))
d <- diag(1/sqrt(diag(cc))); ea <- d%*%cc%*%d      #partial correlation matrix controlling
components
som <- sum(ea^2); somq <- sum(ea^4)
f[i+1] <- (som-n)/m; fq[i+1] <- (somq-n)/m }
matt <- replace(f, f == "NaN", 2)
matt2 <- replace(fq, fq == "NaN", 2)
fm <- min(matt); fqm <- min(matt2)
for (i in 1:(n-1)) {
if (f[i]==fm) { fma <- i-1; break } }
for (i in 1:(n-1)) {
if (fq[i]==fqm) { fqma <- i-1; break } }
junto <- data.frame(f[1:(n-1)],fq[1:(n-1)],row.names=0:(n-2))
names(junto) <- c("Squared average partial correlations", "4th average partial correlations")
spsspivottable.Display(junto,
title="Velicer's MAP values")
table <- spss.BasePivotTable("Velicer's Minimum Average Partial Test","OMS")
rowdim <- BasePivotTable.Append(table,Dimension.Place.row,"")
coldim <- BasePivotTable.Append(table,Dimension.Place.column,"Velicer's Minimum")
row1 <- spss.CellText.String("Squared MAP")
row2 <- spss.CellText.String("4th power MAP")
col1 <- spss.CellText.String("Minimum")
col2 <- spss.CellText.String("Components to retain")
BasePivotTable.SetCategories(table,rowdim,list(row1, row2))
BasePivotTable.SetCategories(table,coldim,list(col1,col2))
BasePivotTable.SetCellsByRow(table,row1,list(spss.CellText.Number(fm),
spss.CellText.Number(fma,6)))
BasePivotTable.SetCellsByRow(table,row2,list(spss.CellText.Number(fqm),
spss.CellText.Number(fqma,6))) }
# Very Simple Structure
if ("n"=="y") {
v <- VSS(co, n = , rotate = "", diagonal = , fm = "", n.obs=ncase, plot=TRUE)
```

```
junto <- data.frame(v$cfit.1, v$cfit.2)
names(junto) <- c("Complexity 1","Complexity 2")
spsspivottable.Display(junto,
title="Very Simple Structure")
nu <-
vss1 <- 0
for (i in 1:nu) {
vss1 <- vss1+1
if ( v$cfit.1[i]==max(v$cfit.1) ) { break }
 }
vss2 <- 0
for (i in 1:nu) {
vss2 <- vss2+1
if ( v$cfit.2[i]==max(v$cfit.2) ) { break }
 }
table <- spss.BasePivotTable("Very Simple Structure (Number of factors)","OMS")
rowdim <- BasePivotTable.Append(table,Dimension.Place.row,"")
coldim <- BasePivotTable.Append(table,Dimension.Place.column,"Very Simple Structure
(VSS)")
row1 <- spss.CellText.String("Values")
col1 <- spss.CellText.String("Rotation")
col2 <- spss.CellText.String("Factoring method")
col3 <- spss.CellText.String("Max VSS complexity 1")
col4 <- spss.CellText.String("N factors complexity 1")
col5 <- spss.CellText.String("Max VSS complexity 2")
col6 <- spss.CellText.String("N factors complexity 2")
BasePivotTable.SetCategories(table,rowdim,list(row1))
BasePivotTable.SetCategories(table,coldim,list(col1,col2,col3,col4,col5,col6))
BasePivotTable.SetCellsByColumn(table,col1,list(spss.CellText.String("")))
BasePivotTable.SetCellsByColumn(table,col2,list(spss.CellText.String("")))
BasePivotTable.SetCellsByColumn(table,col3,list(spss.CellText.Number(max(v$cfit.1))))
BasePivotTable.SetCellsByColumn(table,col4,list(spss.CellText.Number(vss1,6)))
BasePivotTable.SetCellsByColumn(table,col5,list(spss.CellText.Number(max(v$cfit.2))))
BasePivotTable.SetCellsByColumn(table,col6,list(spss.CellText.Number(vss2,6)))
 }
spsspkg.EndProcedure()
END PROGRAM.
BEGIN PROGRAM R.
# More Number of Factors
if("no"=="yes") {
spsspkg.StartProcedure ("Comparison Data")
N.Pop <-
sdata <- matrix (0,ncol = ncol(mdata), nrow =ncase_list)
if (""=="yes") {
mdata2 <- as.matrix(mdata)
lcor <- 0
for (i in 1:ncase2) {
ind <- 0
j <- 0
while (j<n && ind ==0) {
j <- j+1
```

```
if (is.na(mdata[i,j])) { ind <- 1 } }
if (ind==0) { lcor <- lcor+1; sdata[lcor,] <- mdata2[i,] } } }
else { sdata <- mdata }
# Matrix to store each variable score distribution
Distributions <- matrix(0, nrow = N.Pop, ncol = dim(sdata)[2])
# Generate distribution for each variable
if (""!="") { set.seed() }
b2 <- matrix(0, nrow = dim(sdata)[2], ncol = 1)
for (i in 1:dim(sdata)[2]) {
bb <- sort(sample(sdata[,i], size = N.Pop, replace = T) )
b2[i] <- length(bb)
ac <- N.Pop-b2[i]
if (b2[i] < N.Pop) {
for (j in 1:ac) {
qw <- sample(1: (b2[i] + j),1)
bb <- insert (bb, qw, NA) } }
Distributions[,i] <- as.matrix(bb) }
EFA.Comp.Data <- function(Data, F.Max, N.Samples=500, Alpha = 0.3) {
# Data = N by k data matrix
# F.Max = largest number of factors to consider
# N.Pop = size of finite populations of comparison data
# N.Samples = number of samples drawn from each population
# Alpha = alpha level testing significance of improvement with adding factor
sig2 <- data.frame(1)
N <- dim(Data)[1]
k <- dim(Data)[2]
Data2 <- as.data.frame(Data)
if (""=="a") { cor.Data <- cor(Data,use="complete.obs") }
if (""=="b") {
for (i in 1:k) { Data2[,i]<-ordered(Data[,i])  }
corY <-  hetcor(Data2,  ML = FALSE, std.err = FALSE, pd=TRUE, use="complete.obs",
digits=4)
cor.Data <- corY$correlations }
if (""=="c") {
for (i in 1:k) { Data2[,i]<-ordered(Data[,i])  }
corY <-  hetcor(Data2,  ML = TRUE, std.err = FALSE, pd=TRUE, use="complete.obs",
digits=4)
cor.Data <- corY$correlations }
if (""=="d") { cor.Data <- cor(Data,use="complete.obs",method="spearman") }
if (""=="e") {
for (i in 1:k) {
if  ( scal["varMeasurementLevel",i]=="nominal") { Data2[,i]<-factor(Data[,i]) }
else { if  (scal["varMeasurementLevel",i]=="ordinal") { Data2[,i]<-ordered(Data[,i]) } } }
corY <-  hetcor(Data2,  ML = FALSE, std.err = FALSE, pd=TRUE, use="complete.obs",
digits=4)
cor.Data <- corY$correlations }
if (""=="f") {
for (i in 1:k) {
if  ( scal["varMeasurementLevel",i]=="nominal") { Data2[,i]<-factor(Data[,i]) }
else { if  (scal["varMeasurementLevel",i]=="ordinal") { Data2[,i]<-ordered(Data[,i]) } } }
```

```
corY <-  hetcor(Data2,  ML = TRUE, std.err = FALSE, pd=TRUE, use="complete.obs",
digits=4)
cor.Data <- corY$correlations }
Eigs.Data <- eigen(cor.Data)$values
RMSR.Eigs <- matrix(0, nrow = N.Samples, ncol = F.Max)
Sig <- T
F.CD <- 1
nf <- -1
while (F.CD <= F.Max) {
Pop <- GenData(Data, N.Factors = F.CD, N = N.Pop, Target.Corr = cor.Data)
for (j in 1:N.Samples) {
Samp <- Pop[sample(1:N.Pop, size = N, replace = T),]
Samp2 <- as.data.frame(Samp)
if (""=="a") { cor.Samp <- cor(Samp,use="complete.obs") }
if (""=="b") {
for (i in 1:k) { Samp2[,i]<-ordered(Samp[,i])  }
corY <-  hetcor(Samp2,  ML = FALSE, std.err = FALSE, pd=TRUE, use="complete.obs",
digits=4)
cor.Samp <- corY$correlations }
if (""=="c") {
for (i in 1:k) { Samp2[,i]<-ordered(Samp[,i])  }
corY <-  hetcor(Samp2,  ML = TRUE, std.err = FALSE, pd=TRUE, use="complete.obs",
digits=4)
cor.Samp <- corY$correlations }
if (""=="d") { cor.Samp <- cor(Samp,use="complete.obs",method="spearman") }
if (""=="e") {
for (i in 1:k) {
if  ( scal["varMeasurementLevel",i]=="nominal") { Samp2[,i]<-factor(Samp[,i]) }
else { if  (scal["varMeasurementLevel",i]=="ordinal") { Samp2[,i]<-ordered(Samp[,i]) } } }
corY <-  hetcor(Samp2,  ML = FALSE, std.err = FALSE, pd=TRUE, use="complete.obs",
digits=4)
cor.Samp <- corY$correlations }
if (""=="f") {
for (i in 1:k) {
if  ( scal["varMeasurementLevel",i]=="nominal") { Samp2[,i]<-factor(Samp[,i]) }
else { if  (scal["varMeasurementLevel",i]=="ordinal") { Samp2[,i]<-ordered(Samp[,i]) } } }
corY <-  hetcor(Samp2,  ML = TRUE, std.err = FALSE, pd=TRUE, use="complete.obs",
digits=4)
cor.Samp <- corY$correlations }
Eigs.Samp <- eigen(cor.Samp)$values
RMSR.Eigs[j,F.CD] <- sqrt(sum((Eigs.Samp - Eigs.Data) * (Eigs.Samp - Eigs.Data)) / k) }
if (F.CD > 1) {
sig2[F.CD, ] <- wilcox.test(RMSR.Eigs[,F.CD], RMSR.Eigs[,(F.CD - 1)], "less")$p.value }
else { sig2[1,] <- NA }
if (F.CD > 1)  {
Sig <- (wilcox.test(RMSR.Eigs[,F.CD], RMSR.Eigs[,(F.CD - 1)], "less")$p.value < Alpha) }
F.CD <- F.CD + 1
if (Sig == FALSE && nf == -1) {
nf = F.CD - 2
if (""=="yes") { break }
}
```

```
 }
if (nf == -1) { nf = F.Max }
# graph
if (""=="yes") { x.max <- min(nf+1, F.Max) }
else { x.max <- F.Max }
ys <- apply(RMSR.Eigs[,1:x.max], 2, mean)
plot(x = 1:x.max, y = ys, ylim = c(0, max(ys)), xlab = "Factor",
ylab = "RMSR Eigenvalue", type = "b", main = "Fit to Comparison Data")
abline(v = nf, lty = 3)
lis <- list(ys = ys, nf = nf, sig2 = sig2, x.max = x.max)
return(lis)
 }
GenData <- function(Supplied.Data, N.Factors, N, Max.Trials = 5, Initial.Multiplier = 1,
Target.Corr)
{
# Ruscio, J., & Kaczetow, W. (2008).
# Simulating multivariate nonnormal data using an iterative algorithm.
# Multivariate Behavioral Research, 43(3), 355-381.
k <- dim(Supplied.Data)[2]
Iteration <- 0   # Iteration counter
Best.RMSR <- 1   # Lowest RMSR correlation
Trials.Without.Improvement <- 0   # Trial counter
Data <- matrix(0, nrow = N, ncol = k)   # Matrix to store the simulated data
# Calculate and store a copy of the target correlation matrix
Intermediate.Corr <- Target.Corr
# Generate random normal data, initialize factor loadings
Shared.Comp <- matrix(rnorm(N * N.Factors, 0, 1), nrow = N, ncol = N.Factors)
Unique.Comp <- matrix(rnorm(N.Pop * dim(sdata)[2], 0, 1), nrow = N.Pop, ncol =
dim(sdata)[2])
Shared.Load <- matrix(0, nrow = k, ncol = N.Factors)
Unique.Load <- matrix(0, nrow = k, ncol = 1)
# Begin loop that ends when specified n of iterations pass without improvement in RMSR
correlation
while (Trials.Without.Improvement < Max.Trials)  {
Iteration <- Iteration + 1
# Calculate factor loadings and apply to reproduce desired correlations
Fact.Anal <- Factor.Analysis(Intermediate.Corr, N.Factors = N.Factors)
if (N.Factors == 1) { Shared.Load[,1] <- Fact.Anal$loadings }
else {
for (i in 1:N.Factors) { Shared.Load[,i] <- Fact.Anal$loadings[,i] } }
Shared.Load[Shared.Load > 1] <- 1
Shared.Load[Shared.Load < -1] <- -1
if (Shared.Load[1,1] < 0) { Shared.Load <- Shared.Load * -1 }
for (i in 1:k) {
if (sum(Shared.Load[i,] * Shared.Load[i,]) < 1) {
Unique.Load[i,1] <- (1 - sum(Shared.Load[i,] * Shared.Load[i,])) }
else { Unique.Load[i,1] <- 0 } }
Unique.Load <- sqrt(Unique.Load)
for (i in 1:k) {
Data[,i] <- (Shared.Comp %*% t(Shared.Load))[,i] + Unique.Comp[,i] * Unique.Load[i,1] }
# Replace normal with nonnormal distributions
```

```
for (i in 1:k) {
Data <- Data[sort.list(Data[,i]),]
Data[,i] <- Distributions[,i] }
# Calculate RMSR correlation, compare to lowest value, take appropriate action
Data2 <- as.data.frame(Data)
if (""=="a") { Reproduced.Corr <- cor(Data,use="complete.obs") }
if (""=="b") {
for (i in 1:k) { Data2[,i]<-ordered(Data[,i])  }
corY <-  hetcor(Data2,  ML = FALSE, std.err = FALSE, pd=TRUE, use="complete.obs",
digits=4)
Reproduced.Corr <- corY$correlations }
if (""=="c") {
for (i in 1:k) { Data2[,i]<-ordered(Data[,i])  }
corY <-  hetcor(Data2,  ML = TRUE, std.err = FALSE, pd=TRUE, use="complete.obs",
digits=4)
Reproduced.Corr <- corY$correlations }
if (""=="d") { Reproduced.Corr <- cor(Data,use="complete.obs",method="spearman") }
if (""=="e") {
for (i in 1:k) {
if  ( scal["varMeasurementLevel",i]=="nominal") { Data2[,i]<-factor(Data[,i]) }
else { if  (scal["varMeasurementLevel",i]=="ordinal") { Data2[,i]<-ordered(Data[,i]) } } }
corY <-  hetcor(Data2,  ML = FALSE, std.err = FALSE, pd=TRUE, use="complete.obs",
digits=4)
Reproduced.Corr <- corY$correlations }
if (""=="f") {
for (i in 1:k) {
if  ( scal["varMeasurementLevel",i]=="nominal") { Data2[,i]<-factor(Data[,i]) }
else { if  (scal["varMeasurementLevel",i]=="ordinal") { Data2[,i]<-ordered(Data[,i]) } } }
corY <-  hetcor(Data2,  ML = TRUE, std.err = FALSE, pd=TRUE, use="complete.obs",
digits=4)
Reproduced.Corr <- corY$correlations }
Residual.Corr <- Target.Corr - Reproduced.Corr
RMSR <-
sqrt(sum(Residual.Corr[lower.tri(Residual.Corr)]*Residual.Corr[lower.tri(Residual.Corr)]) /
(.5* (k*k - k)))
if (RMSR < Best.RMSR)  {
Best.RMSR <- RMSR
Best.Corr <- Intermediate.Corr
Best.Res <- Residual.Corr
Intermediate.Corr <- Intermediate.Corr + Initial.Multiplier * Residual.Corr
Trials.Without.Improvement <- 0 }
else  {
Trials.Without.Improvement <- Trials.Without.Improvement + 1
Current.Multiplier <- Initial.Multiplier * .5 ^ Trials.Without.Improvement
Intermediate.Corr <- Best.Corr + Current.Multiplier * Best.Res }  }
# Construct the data set with the lowest RMSR correlation
Fact.Anal <- Factor.Analysis(Best.Corr, N.Factors = N.Factors)
if (N.Factors == 1) { Shared.Load[,1] <- Fact.Anal$loadings }
else {
for (i in 1:N.Factors) {
Shared.Load[,i] <- Fact.Anal$loadings[,i] } }
```

```
Shared.Load[Shared.Load > 1] <- 1
Shared.Load[Shared.Load < -1] <- -1
if (Shared.Load[1,1] < 0) { Shared.Load <- Shared.Load * -1 }
for (i in 1:k) {
if (sum(Shared.Load[i,] * Shared.Load[i,]) < 1)  {
Unique.Load[i,1] <- (1 - sum(Shared.Load[i,] * Shared.Load[i,])) }
else { Unique.Load[i,1] <- 0 } }
Unique.Load <- sqrt(Unique.Load)
for (i in 1:k) {
Data[,i] <- (Shared.Comp %*% t(Shared.Load))[,i] + Unique.Comp[,i] * Unique.Load[i,1] }
Data <- apply(Data, 2, scale) # standardizes each variable in the matrix
for (i in 1:k) {
Data <- Data[sort.list(Data[,i]),]
Data[,i] <- Distributions[,i]  }
# Return the simulated data set
return(Data) }
Factor.Analysis <- function(Data, Max.Iter = 50, N.Factors = 0)
{
Data <- as.matrix(Data)
k <- dim(Data)[2]
if (N.Factors == 0) {
N.Factors <- k
Determine <- T  }
else { Determine <- F }
Cor.Matrix <- Data
Criterion <- .001
Old.H2 <- rep(99, k)
H2 <- rep(0, k)
Change <- 1
Iter <- 0
Factor.Loadings <- matrix(nrow = k, ncol = N.Factors)
while ((Change >= Criterion) & (Iter < Max.Iter)) {
Iter <- Iter + 1
Eig <- eigen(Cor.Matrix)
L <- sqrt(Eig$values[1:N.Factors])
for (i in 1:N.Factors) {
Factor.Loadings[,i] <- Eig$vectors[,i] * L[i] }
for (i in 1:k) {
H2[i] <- sum(Factor.Loadings[i,] * Factor.Loadings[i,]) }
Change <- max(abs(Old.H2 - H2))
Old.H2 <- H2
diag(Cor.Matrix) <- H2
   }
if (Determine) { N.Factors <- sum(Eig$values > 1) }
return(list(loadings = Factor.Loadings[,1:N.Factors], factors = N.Factors))
}
ysa<-EFA.Comp.Data(Data = sdata, F.Max =  ,
N.Samples = , Alpha = )
junto <- data.frame(paste("",1:ysa$x.max, rep = "factor"), ysa$ys, ysa$sig2)
names(junto) <- c("Number of factors","RMSR Eigenvalue","p-value")
spsspivottable.Display(junto,
```

```
title="Fit to Comparison Data", hiderowdimlabel=TRUE )
if (""=="a") { mat <- "Pearson" }
else { if (""=="b") { mat <- "Polychoric (Two Step)" }
else { if (""=="c") { mat <- "Polychoric (Max. Lik.)" }
else { if (""=="d") { mat <- "Spearman" }
else { if (""=="e") { mat <- "Heterogenous (Two Step)" }
else { if (""=="f") { mat <- "Heterogenous (Max. Lik.)" }
 } } } } }
junto <- data.frame(mat,ysa$nf)
names(junto) <- c("Correlations","Number of factors to retain")
spsspivottable.Display(junto,
title="Comparison Data", hiderowdimlabel=TRUE, format=6)
spsspkg.EndProcedure()
 }
END PROGRAM.
BEGIN PROGRAM R.
# Items, Scales and Realibility
ind <- 0
spsspkg.StartProcedure ("Items, Scales and Realibility")
if ("n"=="y" || "n"=="y") {
if ("no"=="yes" && "a"=="a") { w <- factor2cluster(load, cut = 0);nu <- ncol(w); ind <- 1; w
<- as.data.frame(w); names(w) <- paste ("S",1:nu, rep="") }
if ("a"=="m") { ke <- list(c(1,3,-4),c(6,7))
w <- make.keys(n, ke, key.labels = NULL, item.labels = nam);nu <- ncol(w); ind <- 1; w <-
as.data.frame(w); names(w) <- paste ("S",1:nu, rep="") }
if ("a"=="all") { S1 <- rep(1,n); w <- data.frame(S1, row.names=nam); ind <- 1 }
if (ind==1) {
nu <- ncol(w)
spsspivottable.Display(w,
title="Assigned items to clusters (scores are adjusted for reverse scored items)", format=6)
if ("n"=="y") {
see <- scoreItems(w, mdata, totals = , ilabels = nam, missing = , impute="", delete=FALSE,
digits=6) }
else { see <- scoreItems(w, mdata, totals = TRUE, ilabels = nam, missing = TRUE,
impute="median", delete=FALSE, digits=6) }
junto <- data.frame(see$n.items)
names(junto) <- c("N Items")
spsspivottable.Display(junto,
title="Number of items for each scale",
format=6)
junto <- data.frame(see$av.r)
names(junto) <- paste ("S",1:nu, rep="")
spsspivottable.Display(junto, hiderowdimlabel=TRUE,
title="Average correlation")
spsspivottable.Display(see$cor, title="Intercorrelation of scales")
see$corrected[lower.tri(see$corrected)] <- t(see$corrected)[lower.tri(t(see$corrected))]
spsspivottable.Display(see$corrected,
title="Unattenuated correlations of scales (alpha on the diagonal)")
spsspivottable.Display(see$item.cor,
title="Correlation of items with scales (not corrected)")
spsspivottable.Display(see$item.corrected,
```

```
title=" Correlation of items with scales (corrected for item overlap)")
# Beggining Cronbach alpha
if (""=="y") {
cc <- cov(mdata, use="complete.obs", method="pearson")
alpha.u <- rep(0,nu)
alpha.s <- rep(0,nu)
if (m1==0) { co <- cor(mdata, use="complete.obs", method="pearson"); cor1 <- co; m1 <- 1 }
else { co <- cor1 }
for (i in 1:nu) {
ke <- diag(w[,i]) %*% co %*% diag(w[,i])
ke2 <- diag(w[,i]) %*% cc %*% diag(w[,i])
s <- sum(abs(w[,i]))
alpha.s[i] <- (1 - tr(ke)/sum(ke)) * (s/(s - 1))
alpha.u[i] <- (1 - tr(ke2)/sum(ke2)) * (s/(s - 1)) }
junto <- data.frame(t(alpha.u))
names(junto) <- paste ("S",1:nu, rep="")
spsspivottable.Display(junto, hiderowdimlabel=TRUE,
title="Raw Cronbach alpha")
junto <- data.frame(t(alpha.s))
names(junto) <- paste ("S",1:nu, rep="")
spsspivottable.Display(junto, hiderowdimlabel=TRUE,
title="Standardized Cronbach alpha")
w2 <- w
alp.u <- matrix (nrow=n,ncol=nu)
alp.s <- matrix (nrow=n,ncol=nu)
for (i in 1:n) {
for (j in 1:nu) {
if (w[i,j] != 0) { w2[i,j] = 0
ke <- diag(w2[,j]) %*% co %*% diag(w2[,j])
ke2 <- diag(w2[,j]) %*% cc %*% diag(w2[,j])
s <- sum(abs(w2[,j]))
alp.s[i,j] <- (1 - tr(ke)/sum(ke)) * (s/(s - 1))
alp.u[i,j] <- (1 - tr(ke2)/sum(ke2)) * (s/(s - 1))
w2 <- w } } }
junto <- data.frame(alp.u, row.names=nam)
names(junto) <- paste ("S",1:nu, rep="")
spsspivottable.Display(junto,
title="Raw Cronbach alpha if item deleted")
junto <- data.frame(alp.s, row.names=nam)
names(junto) <- paste ("S",1:nu, rep="")
spsspivottable.Display(junto,
title="Standardized Cronbach alpha if item deleted") } # end Cronbach alpha
# Beginning ordinal Cronbach alpha
if (""=="y") {
if (""=="a") {
if (m2==0) { co <- hetcor(da,  ML = FALSE, std.err = FALSE, pd=TRUE,
use="complete.obs", digits=6)
co <- co$correlations; cor2 <- co; m2 <- 1 }
else { co <- cor2 } }
if (""=="b") {
```

```
if (m3==0) { co <- hetcor(da,  ML = TRUE, std.err = FALSE, pd=TRUE,
use="complete.obs", digits=6)
co <- co$correlations; cor3<- co; m3 <- 1 }
else { co <- cor3 } }
or.al <- rep(0,nu)
for (i in 1:nu) {
ke <- diag(w[,i]) %*% co %*% diag(w[,i])
s <- sum(abs(w[,i]))
or.al[i] <- (1 - tr(ke)/sum(ke)) * (s/(s - 1)) }
junto <- data.frame(t(or.al))
names(junto) <- paste ("S",1:nu, rep="")
spsspivottable.Display(junto, hiderowdimlabel=TRUE,
title="Ordinal coefficient alpha")
w2 <- w
or.al <- matrix (nrow=n,ncol=nu)
for (i in 1:n) {
for (j in 1:nu) {
if (w[i,j] != 0) { w2[i,j] = 0
ke <- diag(w2[,j]) %*% co %*% diag(w2[,j])
s <- sum(abs(w2[,j]))
or.al[i,j] <- (1 - tr(ke)/sum(ke)) * (s/(s - 1))
w2 <- w } } }
junto <- data.frame(or.al, row.names=nam)
names(junto) <- paste ("S",1:nu, rep="")
spsspivottable.Display(junto,
title="Ordinal coefficient alpha if item deleted") } # end ordinal Cronbach alpha
# Beginning Armor's reliability theta
if (""=="y")  {
armo <- rep(0,nu)
co3 <- list()
if (m1==0) { co <- cor(mdata, use="complete.obs", method="pearson") }
else { co <- cor1 }
for (j in 1:nu) {
con <- 0
co2 <- co
for (i in 1:n) {
if (w[i,j] == 0) { co2 <- co2[-(i-con),-(i-con)]; con <- con+1 }
else { if (w[i,j] == -1) { co2[i-con,] <- (-1)*co2[i-con,]; co2[,i-con] <- (-1)*co2[,i-con] } } }
co3[[j]] <- co2
ei <- eigen(co2, symmetric=TRUE, only.values=TRUE)
ei <- ei$values
s <- dim(as.matrix(co2))[1]
armo[j] <- ((ei[1]-1)/ei[1]) * (s/(s - 1)) }
junto <- data.frame(t(armo))
names(junto) <- paste ("S",1:nu, rep="")
spsspivottable.Display(junto, hiderowdimlabel=TRUE,
title="Armor's reliability theta")
armo <- matrix (nrow=n,ncol=nu)
indi <- list()
for (j in 1:nu) {
con <- 1
```

```
for (i in 1:n) { if (w[i,j] != 0) {indi[[con]] <- i; con <- con+1} }
s <- dim(as.matrix(co3[[j]]))[1]
for (i in 1:s) {
co2 <- as.matrix(co3[[j]])
if (dim(co2)[1] > 1) {
co2 <- co2[-i,-i]
ei <- eigen(co2, symmetric=TRUE, only.values=TRUE)
ei <- ei$values
armo[indi[[i]],j] <- ((ei[1]-1)/ei[1]) * ((s-1)/(s - 2)) } } }
junto <- data.frame(armo, row.names=nam)
names(junto) <- paste ("S",1:nu, rep="")
spsspivottable.Display(junto,
title="Armor's reliability theta if item deleted") } # end Armor's reliability theta
# Beginning ordinal Armor's reliability theta
if (""=="y")  {
armo <- rep(0,nu)
co3 <- list()
if (""=="a") {
if (m2==0) {
co <- hetcor(da,  ML = FALSE, std.err = FALSE, pd=TRUE, use="complete.obs", digits=6)
co <- co$correlations; cor2 <- co; m2 <- 1 }
else { co <- cor2 } }
if (""=="b") {
if (m3==0) {
co <- hetcor(da,  ML = TRUE, std.err = FALSE, pd=TRUE, use="complete.obs", digits=6)
co <- co$correlations; cor3 <- co; m3 <-1 }
else { co <- cor3 } }
for (j in 1:nu) {
con <- 0
co2 <- co
for (i in 1:n) {
if (w[i,j] == 0) { co2 <- co2[-(i-con),-(i-con)]; con <- con+1 }
else { if (w[i,j] == -1) { co2[i-con,] <- (-1)*co2[i-con,]; co2[,i-con] <- (-1)*co2[,i-con] } } }
co3[[j]] <- co2
ei <- eigen(co2, symmetric=TRUE, only.values=TRUE)
ei <- ei$values
s <- dim(as.matrix(co2))[1]
armo[j] <- ((ei[1]-1)/ei[1]) * (s/(s - 1)) }
junto <- data.frame(t(armo))
names(junto) <- paste ("S",1:nu, rep="")
spsspivottable.Display(junto, hiderowdimlabel=TRUE,
title="Ordinal coefficient theta")
armo <- matrix (nrow=n,ncol=nu)
indi <- list()
for (j in 1:nu) {
con <- 1
for (i in 1:n) { if (w[i,j] != 0) {indi[[con]] <- i; con <- con+1} }
s <- dim(as.matrix(co3[[j]]))[1]
for (i in 1:s) {
co2 <- as.matrix(co3[[j]])
if (dim(co2)[1]>1) {
```

```
co2 <- co2[-i,-i]
ei <- eigen(co2, symmetric=TRUE, only.values=TRUE)
ei <- ei$values
armo[indi[[i]],j] <- ((ei[1]-1)/ei[1]) * ((s-1)/(s - 2)) } } }
junto <- data.frame(armo, row.names=nam)
names(junto) <- paste ("S",1:nu, rep="")
spsspivottable.Display(junto,
title="Ordinal coefficient theta if item deleted") } # end ordinal Armor's reliability theta
 } #end (ind=1)
 }
spsspkg.EndProcedure()
# Saving scores to SPSS
if ("n"=="y") {
if (""=="y" && ind==1) {
rm(mdata)
dict1 <- spssdictionary.GetDictionaryFromSPSS()
mdata <- spssdata.GetDataFromSPSS()
g <- list()
com <- ncol(w)
for (i in 1:com) { h <- as.character(i)
h <- paste("factor", h, sep="")
g[[i]] <- c(h,"",0,"F8.4","scale") }
dict <- spssdictionary.CreateSPSSDictionary(g)
dict <- data.frame(dict1,dict)
spssdictionary.SetDictionaryToSPSS("",dict)
fa=nrow(mdata)
fb=row.names(as.data.frame(see$scores))
ult=fb[nrow(see$scores)]
dife=fa-as.numeric(ult)
line=rep(NA,ncol(see$scores))
if (com==1) {
if (fb[1] != 1)  {
see$scores=rbind(as.matrix(line),as.matrix(see$scores[1:nrow(see$scores),]))
fb=row.names(as.data.frame(see$scores)) }
for (i in 2:(fa-dife-1)) {
if  (fb[i] != i) {
see$scores=rbind(as.matrix(see$scores[1:(i-
1),]),as.matrix(line),as.matrix(see$scores[i:nrow(see$scores),]))
fb=row.names(as.data.frame(see$scores)) } }
if (dife>0) {
for (i in 1:dife) {
see$scores=rbind(as.matrix(see$scores),as.matrix(line))  } } }
else{
if (fb[1] != 1)  {
see$scores=rbind(line,as.data.frame(see$scores[1:nrow(see$scores),]))
fb=row.names(as.data.frame(see$scores)) }
for (i in 2:(fa-dife-1)) {
if  (fb[i] != i) {
see$scores=rbind(as.data.frame(see$scores[1:(i-
1),]),line,as.data.frame(see$scores[i:nrow(see$scores),]))
fb=row.names(as.data.frame(see$scores)) } }
```

```
if (dife>0) {
for (i in 1:dife) {
see$scores=rbind(as.data.frame(see$scores),line)  } } }
see$scores=data.frame(see$scores,row.names=1:fa)
casedata <- data.frame(mdata,see$scores)
spssdata.SetDataToSPSS("",casedata)
spssdictionary.EndDataStep() } }
END PROGRAM.
set printback on.
```

Step 2: Factor analysis with 29 remaining items of the SSAW scale.

```
FACTOR
  /VARIABLES SWSRS_prä_aa_2 SWSRS_prä_aa_5 SWSRS_prä_aa_6 SWSRS_prä_sm_1
SWSRS_prä_sm_3 SWSRS_prä_sm_7 SWSRS_prä_sm_8 SWSRS_akt_sk_1
SWSRS_akt_sk_7 SWSRS_akt_sk_11 SWSRS_akt_sk_12 SWSRS_akt_sk_15
SWSRS_akt_sk_17 SWSRS_akt_sk_18 SWSRS_akt_sk_19 SWSRS_akt_sb_2
SWSRS_akt_sb_4 SWSRS_akt_sb_7 SWSRS_akt_sb_9 SWSRS_akt_sb_10
SWSRS_post_sb_1 SWSRS_post_sb_3 SWSRS_post_sb_4 SWSRS_post_sr_2
SWSRS_post_sr_3 SWSRS_post_sr_4 SWSRS_post_sr_5 SWSRS_post_sr_6
SWSRS_post_sr_7
  /MISSING LISTWISE
  /ANALYSIS SWSRS_prä_aa_2 SWSRS_prä_aa_5 SWSRS_prä_aa_6 SWSRS_prä_sm_1
SWSRS_prä_sm_3 SWSRS_prä_sm_7 SWSRS_prä_sm_8 SWSRS_akt_sk_1
SWSRS_akt_sk_7 SWSRS_akt_sk_11 SWSRS_akt_sk_12 SWSRS_akt_sk_15
SWSRS_akt_sk_17 SWSRS_akt_sk_18 SWSRS_akt_sk_19 SWSRS_akt_sb_2
SWSRS_akt_sb_4 SWSRS_akt_sb_7 SWSRS_akt_sb_9 SWSRS_akt_sb_10
SWSRS_post_sb_1 SWSRS_post_sb_3 SWSRS_post_sb_4 SWSRS_post_sr_2
SWSRS_post_sr_3 SWSRS_post_sr_4 SWSRS_post_sr_5 SWSRS_post_sr_6
SWSRS_post_sr_7
  /PRINT UNIVARIATE INITIAL CORRELATION KMO REPR AIC EXTRACTION
ROTATION
  /PLOT EIGEN
  /CRITERIA MINEIGEN(1) ITERATE(100)
  /EXTRACTION PAF
  /CRITERIA ITERATE(100)
  /ROTATION PROMAX(4)
  /METHOD=CORRELATION.

* Parallel Analysis program.

set mxloops=9000 printback=off width=80  seed = 1953125.
matrix.

* enter your specifications here.
compute ncases   = 121.
compute nvars    = 29.
compute ndatsets = 100.
compute percent  = 95.

* Specify the desired kind of parallel analysis, where:
  1 = principal components analysis
```

```
  2 = principal axis/common factor analysis.
compute kind = 2 .

****************** End of user specifications. ******************

* principal components analysis.
do if (kind = 1).
compute evals = make(nvars,ndatsets,-9999).
compute nm1 = 1 / (ncases-1).
loop #nds = 1 to ndatsets.
compute x = sqrt(2 * (ln(uniform(ncases,nvars)) * -1) ) &*
          cos(6.283185 * uniform(ncases,nvars) ).
compute vcv = nm1 * (sscp(x) - ((t(csum(x))*csum(x))/ncases)).
compute d = inv(mdiag(sqrt(diag(vcv)))).
compute evals(:,#nds) = eval(d * vcv * d).
end loop.
end if.

* principal axis / common factor analysis with SMCs on the diagonal.
do if (kind = 2).
compute evals = make(nvars,ndatsets,-9999).
compute nm1 = 1 / (ncases-1).
loop #nds = 1 to ndatsets.
compute x = sqrt(2 * (ln(uniform(ncases,nvars)) * -1) ) &*
          cos(6.283185 * uniform(ncases,nvars) ).
compute vcv = nm1 * (sscp(x) - ((t(csum(x))*csum(x))/ncases)).
compute d = inv(mdiag(sqrt(diag(vcv)))).
compute r = d * vcv * d.
compute smc = 1 - (1 &/ diag(inv(r)) ).
call setdiag(r,smc).
compute evals(:,#nds) = eval(r).
end loop.
end if.

* identifying the eigenvalues corresponding to the desired percentile.
compute num = rnd((percent*ndatsets)/100).
compute results = { t(1:nvars), t(1:nvars), t(1:nvars) }.
loop #root = 1 to nvars.
compute ranks = rnkorder(evals(#root,:)).
loop #col = 1 to ndatsets.
do if (ranks(1,#col) = num).
compute results(#root,3) = evals(#root,#col).
break.
end if.
end loop.
end loop.
compute results(:,2) = rsum(evals) / ndatsets.

print /title="PARALLEL ANALYSIS:".
do if   (kind = 1).
print /title="Principal Components".
```

```
else if (kind = 2).
print /title="Principal Axis / Common Factor Analysis".
end if.
compute specifs = {ncases; nvars; ndatsets; percent}.
print specifs /title="Specifications for this Run:"
 /rlabels="Ncases" "Nvars" "Ndatsets" "Percent".
print results /title="Random Data Eigenvalues"
 /clabels="Root" "Means" "Prcntyle"  /format "f12.6".

do if   (kind = 2).
print / space = 1.
print /title="Compare the random data eigenvalues to the".
print /title="real-data eigenvalues that are obtained from a".
print /title="Common Factor Analysis in which the # of factors".
print /title="extracted equals the # of variables/items, and the".
print /title="number of iterations is fixed at zero;".
print /title="To obtain these real-data values using SPSS, see the".
print /title="sample commands at the end of the parallel.sps program,".
print /title="or use the rawpar.sps program.".
print / space = 1.
print /title="Warning: Parallel analyses of adjusted correlation matrices".
print /title="eg, with SMCs on the diagonal, tend to indicate more factors".
print /title="than warranted (Buja, A., & Eyuboglu, N., 1992, Remarks on parallel".
print /title="analysis. Multivariate Behavioral Research, 27, 509-540.).".
print /title="The eigenvalues for trivial, negligible factors in the real".
print /title="data commonly surpass corresponding random data eigenvalues".
print /title="for the same roots. The eigenvalues from parallel analyses".
print /title="can be used to determine the real data eigenvalues that are".
print /title="beyond chance, but additional procedures should then be used".
print /title="to trim trivial factors.".
print / space = 1.
print /title="Principal components eigenvalues are often used to determine".
print /title="the number of common factors. This is the default in most".
print /title="statistical software packages, and it is the primary practice".
print /title="in the literature. It is also the method used by many factor".
print /title="analysis experts, including Cattell, who often examined".
print /title="principal components eigenvalues in his scree plots to determine".
print /title="the number of common factors. But others believe this common".
print /title="practice is wrong. Principal components eigenvalues are based".
print /title="on all of the variance in correlation matrices, including both".
print /title="the variance that is shared among variables and the variances".
print /title="that are unique to the variables. In contrast, principal".
print /title="axis eigenvalues are based solely on the shared variance".
print /title="among the variables. The two procedures are qualitatively".
print /title="different. Some therefore claim that the eigenvalues from one".
print /title="extraction method should not be used to determine".
print /title="the number of factors for the other extraction method.".
print /title="The issue remains neglected and unsettled.".

end if.
```

```
end matrix.

corr SWSRS_prä_aa_2 SWSRS_prä_aa_5
   SWSRS_prä_aa_6 SWSRS_prä_sm_1 SWSRS_prä_sm_3 SWSRS_prä_sm_7
SWSRS_prä_sm_8
   SWSRS_akt_sk_1 SWSRS_akt_sk_7 SWSRS_akt_sk_11
   SWSRS_akt_sk_12 SWSRS_akt_sk_15 SWSRS_akt_sk_17
   SWSRS_akt_sk_18 SWSRS_akt_sk_19
   SWSRS_akt_sb_2 SWSRS_akt_sb_4 SWSRS_akt_sb_7
   SWSRS_akt_sb_9 SWSRS_akt_sb_10 SWSRS_post_sb_1 SWSRS_post_sb_3
   SWSRS_post_sb_4 SWSRS_post_sr_2
   SWSRS_post_sr_3 SWSRS_post_sr_4 SWSRS_post_sr_5 SWSRS_post_sr_6
SWSRS_post_sr_7 / matrix out ('C:\Users\Cici\Desktop\SWSRS_EJPA
Hauptachsen\Datensatz_SWSRS') / missing = listwise.
matrix.
MGET /type= corr /file='C:\Users\Cici\Desktop\SWSRS_EJPA
Hauptachsen\Datensatz_SWSRS' .
compute smc = 1 - (1 &/ diag(inv(cr)) ).
call setdiag(cr,smc).
compute evals = eval(cr).
print { t(1:nrow(cr)) , evals }
 /title="Raw Data Eigenvalues"
 /clabels="Root" "Eigen."  /format "f12.6".
end matrix.


* Commands for obtaining the necessary real-data eigenvalues for
  principal axis / common factor analysis using SPSS;
  make sure to insert valid filenames/locations, and
  remove the '*' from the first columns.
* corr var1 to var20 / matrix out ('filename') / missing = listwise.
* matrix.
* MGET /type= corr /file='filename' .
* compute smc = 1 - (1 &/ diag(inv(cr)) ).
* call setdiag(cr,smc).
* compute evals = eval(cr).
* print { t(1:nrow(cr)) , evals }
 /title="Raw Data Eigenvalues"
 /clabels="Root" "Eigen."  /format "f12.6".
* end matrix.

* MAP-Test program.
* Encoding: UTF-8.

*Mário Basto, José Manuel Pereira, IPCA
*Required: SPSS 21 and R Integration Plugin
*R Packages required: psych, polycor, GPArotation, nFactors, corpcor, ICS, R.utils.
set printback off.
BEGIN PROGRAM R.
# Correlations and descriptives
spsspkg.StartProcedure ("Correlation")
```

```
library (polycor)
library (psych)
library (GPArotation)
library (nFactors)
library(corpcor)
library(ICS)
library(R.utils)
# Reading data from SPSS
mdata <- spssdata.GetDataFromSPSS(variables=c("SWSRS_prä_aa_2 SWSRS_prä_aa_5
   SWSRS_prä_aa_6 SWSRS_prä_sm_1 SWSRS_prä_sm_3 SWSRS_prä_sm_7
SWSRS_prä_sm_8
   SWSRS_akt_sk_1 SWSRS_akt_sk_7 SWSRS_akt_sk_11
   SWSRS_akt_sk_12 SWSRS_akt_sk_15 SWSRS_akt_sk_17
   SWSRS_akt_sk_18 SWSRS_akt_sk_19
   SWSRS_akt_sb_2 SWSRS_akt_sb_4 SWSRS_akt_sb_7
   SWSRS_akt_sb_9 SWSRS_akt_sb_10 SWSRS_post_sb_1 SWSRS_post_sb_3
   SWSRS_post_sb_4 SWSRS_post_sr_2
   SWSRS_post_sr_3 SWSRS_post_sr_4 SWSRS_post_sr_5 SWSRS_post_sr_6
SWSRS_post_sr_7"))
scal <- spssdictionary.GetDictionaryFromSPSS(variables=c("SWSRS_prä_aa_2
SWSRS_prä_aa_5
   SWSRS_prä_aa_6 SWSRS_prä_sm_1 SWSRS_prä_sm_3 SWSRS_prä_sm_7
SWSRS_prä_sm_8
   SWSRS_akt_sk_1 SWSRS_akt_sk_7 SWSRS_akt_sk_11
   SWSRS_akt_sk_12 SWSRS_akt_sk_15 SWSRS_akt_sk_17
   SWSRS_akt_sk_18 SWSRS_akt_sk_19
   SWSRS_akt_sb_2 SWSRS_akt_sb_4 SWSRS_akt_sb_7
   SWSRS_akt_sb_9 SWSRS_akt_sb_10 SWSRS_post_sb_1 SWSRS_post_sb_3
   SWSRS_post_sb_4 SWSRS_post_sr_2
   SWSRS_post_sr_3 SWSRS_post_sr_4 SWSRS_post_sr_5 SWSRS_post_sr_6
SWSRS_post_sr_7"))
is.na(mdata) <- is.na(mdata)
m1 <- 0; m2 <- 0; m3 <- 0; m4 <- 0; m5<-0; m6<-0
nam <- names(mdata)
# Missing values (pairwise or listwise)
n <- ncol(mdata)
ncase <- nrow(mdata)
ncase2 <- ncase
 # counting listwise cases
cont <- 0
for (i in 1:ncase) {
ind <- 0
j <- 0
while (j<n && ind ==0) {
j <- j+1
if (is.na(mdata[i,j])) {
cont <- cont+1
ind <- 1 } } }
ncase_list <- ncase-cont
if ("complete.obs"=="complete.obs") {     # counting listwise cases
ncase <- ncase_list
```

```
junto <- data.frame(ncase, cont)
names(junto) <- c("Number of cases", "Number of cases excluded")
spsspivottable.Display(junto,
title="Listwise deletion", hiderowdimlabel=TRUE, format=6) }
if ("complete.obs"=="pairwise.complete.obs") {
spsspivottable.Display(count.pairwise(mdata),
title="Number of valid cases for each pairwise correlation", format=6) }
da <- mdata
dah <- mdata
for (i in 1:n) { da[,i]<-ordered(mdata[,i])}  # for calculation of polychoric correlations
for (i in 1:n) {
if  ( scal["varMeasurementLevel",i]=="nominal") {dah[,i]<-factor(mdata[,i]) }
else { if  (scal["varMeasurementLevel",i]=="ordinal") { dah[,i]<-ordered(mdata[,i]) } } }
#for calculation of correlations according to their scales
# Multivariate normality tests
if ("n"=="y") {
mn <- mvnorm.skew.test(mdata, na.action = na.omit)
mn2 <- mvnorm.kur.test(mdata, method = "satterthwaite", n.simu = 1000, na.action=na.omit)
mm <- c("Skewness","Kurtosis")
mn3 <- c(mn$statistic, mn2$statistic)
mn4 <- c(mn$p.value, mn2$p.value)
junto <- data.frame(mm, mn3, mn4)
names(junto) <- c("Moment","Test statistic", "p-value")
spsspivottable.Display(junto,
title="Test of Multivariate Normality based on Skewness and Kurtosis",
hiderowdimlabel=TRUE) }
if ("n"=="y") { co <- corr.test(mdata, use="complete.obs", method="pearson"); m1 <-1
cor1 <- co$r
test <- co$p
spsspivottable.Display(cor1,
title="Pearson correlations")
spsspivottable.Display(test,
title="p-values for Pearson correlations") }
if ("n"=="y") { co <- hetcor(da,  ML = FALSE, std.err = FALSE, pd=TRUE,
use="complete.obs", digits=6); m2 <-1
cor2 <- co$correlations
spsspivottable.Display(cor2,
title="Polychoric correlations - Two Step Estimation") }
if ("n"=="y") { co <- hetcor(da,  ML = TRUE, std.err = FALSE, pd=TRUE,
use="complete.obs", digits=6); m3 <-1
cor3 <- co$correlations
spsspivottable.Display(cor3,
title="Polychoric correlations - Max. Likelihood Estimation") }
if ("n"=="y") { co <- corr.test(mdata, use="complete.obs", method="spearman"); m4 <-1
cor4 <- co$r
test <- co$p
spsspivottable.Display(cor4,
title="Spearman correlations")
spsspivottable.Display(test,
title="p-values for Spearman correlations") }
```

```
if ("n"=="y") { co <- hetcor(dah,  ML =FALSE , std.err = FALSE, pd=TRUE,
use="complete.obs", digits=6)
m5 <-1
cor5 <- co$correlations
ctype <- as.data.frame(co$type)
names(ctype) <- nam
het<- rbind(signif(cor5, digits=3), ctype)
row.names(het) <- c(nam,paste("Correlation Type",nam))
spsspivottable.Display(het,
title="Heterogenous correlations (Two Step)") }
if ("n"=="y") { co <- hetcor(dah,  ML =TRUE , std.err = FALSE, pd=TRUE,
use="complete.obs", digits=6)
m6 <-1
cor6 <- co$correlations
ctype <- as.data.frame(co$type)
names(ctype) <- nam
het<- rbind(signif(cor6, digits=3), ctype)
row.names(het) <- c(nam,paste("Correlation Type",nam))
spsspivottable.Display(het,
title="Heterogenous correlations (Max. Likelihood)") }
spsspkg.EndProcedure()
END PROGRAM.
BEGIN PROGRAM R.
# Correlation differences
spsspkg.StartProcedure ("Correlation differences")
if ("n"=="y")  {
if (m1==0)  {cor1 <- cor(mdata, use="complete.obs", method="pearson"); m1 <-1 }
if (m2==0)  { co <- hetcor(da,  ML = FALSE, std.err = FALSE, pd=TRUE,
use="complete.obs", digits=6); m2 <-1
cor2 <- co$correlations }
d12 <- cor1-cor2
spsspivottable.Display(d12,
title="Pearson - Polychoric (Two Step)") }
if ("n"=="y")  {
if (m1==0)  {cor1 <- cor(mdata, use="complete.obs", method="pearson"); m1 <-1 }
if (m3==0)  { co <- hetcor(da,  ML = TRUE, std.err = FALSE, pd=TRUE,
use="complete.obs", digits=6); m3 <-1
cor3 <- co$correlations }
d13 <- cor1-cor3
spsspivottable.Display(d13,
title="Pearson - Polychoric (Max. Lik.)")  }
if ("n"=="y")  {
if (m2==0)  { co <- hetcor(da,  ML = FALSE, std.err = FALSE, pd=TRUE,
use="complete.obs", digits=6); m2 <-1
cor2 <- co$correlations }
if (m3==0)  { co <- hetcor(da,  ML = TRUE, std.err = FALSE, pd=TRUE,
use="complete.obs", digits=6); m3 <-1
cor3 <- co$correlations }
d23 <- cor2-cor3
spsspivottable.Display(d23,
title="Polychoric (Two Step) - Polychoric (Max. Lik.)")   }
```

```
if ("n"=="y")  {
if (m1==0)  { cor1 <- cor(mdata, use="complete.obs", method="pearson"); m1 <-1 }
if (m4==0)  { cor4 <- cor(mdata, use="complete.obs", method="spearman"); m4 <-1 }
d14 <- cor1-cor4
spsspivottable.Display(d14,
title="Pearson - Spearman") }
if ("n"=="y")  {
if (m2==0)  { co <- hetcor(da,  ML = FALSE, std.err = FALSE, pd=TRUE,
use="complete.obs", digits=6); m2 <-1
cor2 <- co$correlations }
if (m4==0)  { cor4 <- cor(mdata, use="complete.obs", method="spearman"); m4 <-1 }
d24 <- cor2-cor4
spsspivottable.Display(d24,
title="Polychoric (Two Step) - Spearman") }
if ("n"=="y")  {
if (m3==0)  { co <- hetcor(da,  ML = TRUE, std.err = FALSE, pd=TRUE,
use="complete.obs", digits=6); m3 <-1
cor3 <- co$correlations }
if (m4==0)  { cor4 <- cor(mdata, use="complete.obs", method="spearman"); m4 <-1 }
d34 <- cor3-cor4
spsspivottable.Display(d34,
title="Polychoric (Max. Lik.) - Spearman") }
spsspkg.EndProcedure()
END PROGRAM.
BEGIN PROGRAM R.
# Principal Components Analysis extracting PCs from the correlation matrix (function
princomp)
if ("no"=="yes")  {
if (""=="a") {spsspkg.StartProcedure ("Principal Component Analysis - Pearson") }
else { if (""=="b") {spsspkg.StartProcedure ("Principal Component Analysis - Polychoric
(Two Step)") }
else { if (""=="c") {spsspkg.StartProcedure ("Principal Component Analysis - Polychoric
(Max. Likelihood)") }
else { if (""=="d") {spsspkg.StartProcedure ("Principal Component Analysis - Spearman") }
else { if (""=="e") {spsspkg.StartProcedure ("Principal Component Analysis - Covariance") }
else { if (""=="f") {spsspkg.StartProcedure ("Principal Component Analysis - Heterogenous
(Two Step)") }
else { if (""=="g") {spsspkg.StartProcedure ("Principal Component Analysis - Heterogenous
(Max. Likelihood)") }
} } } } } }
if (""=="a")  {
if (m1==0) {cor1 <- cor(mdata, use="complete.obs", method="pearson"); m1 <-1 }
ad <-princomp(cor = FALSE, covmat = cor1)  }
else { if (""=="b")  {
if (m2==0) { co <- hetcor(da,  ML = FALSE, std.err = FALSE, pd=TRUE,
use="complete.obs", digits=6); m2 <-1
cor2 <- co$correlations }
ad <-princomp(cor = FALSE, covmat = cor2)  }
else { if (""=="c")  {
if (m3==0) { co <- hetcor(da,  ML = TRUE, std.err = FALSE, pd=TRUE,
use="complete.obs", digits=6); m3 <-1
```

```
cor3 <- co$correlations }
ad <-princomp(cor = FALSE, covmat = cor3)  }
else { if (""=="d") {
if (m4==0) { cor4 <- cor(mdata, use="complete.obs", method="spearman"); m4 <-1 }
ad <-princomp(cor = FALSE, covmat = cor4)  }
else { if (""=="e") { cor <- cov(mdata, use="complete.obs")
ad <-princomp(cor = FALSE, covmat = cor) }
else { if (""=="f")  {
if (m5==0) { co <- hetcor(dah,  ML = FALSE, std.err = FALSE, pd=TRUE,
use="complete.obs", digits=6); m5 <-1
cor5 <- co$correlations }
ad <-princomp(cor = FALSE, covmat = cor5)  }
else { if (""=="g")  {
if (m6==0) { co <- hetcor(dah,  ML = TRUE, std.err = FALSE, pd=TRUE,
use="complete.obs", digits=6); m6 <-1
cor6 <- co$correlations }
ad <-princomp(cor = FALSE, covmat = cor6)  }
 } } } } } }
spsspivottable.Display(ad$loadings[,], title="Eigenvectors")
pro <- ad$sdev^2
loa <- ad$loadings
loa <- loa%*%diag(ad$sdev)
# Sorting or not loadings
if (""=="no") {
spsspivottable.Display(loa[,],
title="Component Loadings",
collabels=paste ("Comp.",1:n, sep="") ) }
else {
# To correct a little problem when dealing with matrices for some versions of ICLUST.sort
loab <- list()
loab$loadings <- loa
loab$pattern <- loa
p <- ICLUST.sort(loab)
p <- p$sorted[,4:(n+3)]
spsspivottable.Display(p,
title="Sorted Component Loadings", collabels=paste ("Comp.",1:n, sep="")) }
sume <- 0
sum2 <- rep(0,n)
for (i in 1:n) { sume <- sume + pro[i]
sum2[i] = sum2[i] + sume}
junto <- data.frame(ad$sdev,pro,pro/sume*100,sum2/sume*100)
names(junto) <- c("Stdev","Eigenvalues","% of Variance","Cumulative %")
spsspivottable.Display(junto,
title="Variance Explained")
plot(ad,main="Scree Plot",type="lines")
spsspkg.EndProcedure() }
END PROGRAM.
BEGIN PROGRAM R.
#Factor Analysis (functions "principal", "fa", "iterativePrincipalAxis", "PrincipalAxis")
if ("no"=="yes")  {
suprime <-
```

```
if ("a"=="a") {
if (m1==0) { cor1 <- cor(mdata, use="complete.obs", method="pearson"); m1 <-1 }
co <- cor1 }
else { if ("a"=="b") {
if (m2==0) { co <- hetcor(da,  ML = FALSE, std.err = FALSE, pd=TRUE,
use="complete.obs", digits=6); m2 <-1
cor2 <- co$correlations }
co <- cor2 }
else { if ("a"=="c") {
if (m3==0) { co <- hetcor(da,  ML = TRUE, std.err = FALSE, pd=TRUE,
use="complete.obs", digits=6); m3 <-1
cor3 <- co$correlations }
co <- cor3 }
else { if ("a"=="d") {
if (m4==0) { cor4 <- cor(mdata, use="complete.obs", method="spearman"); m4 <-1 }
co <- cor4 }
else { if ("a"=="e") {
if (m5==0) { co <- hetcor(dah,  ML = FALSE, std.err = FALSE, pd=TRUE,
use="complete.obs", digits=6); m5 <-1
cor5 <- co$correlations }
co <- cor5 }
else { if ("a"=="f") {
if (m6==0) { co <- hetcor(dah,  ML = TRUE, std.err = FALSE, pd=TRUE,
use="complete.obs", digits=6); m6 <-1
cor6 <- co$correlations }
co <- cor6 }
 } } } } }
spsspkg.StartProcedure ("Factor Analysis")
pr <-
if (""=="pca" && pr > n) { pr <- n }
if (""!="pca" && pr > (n-1)) { pr <- n-1 }
if (""=="pca") {
ad<-principal(co, nfactors = pr, residuals = FALSE, rotate="none", n.obs=ncase,
scores=FALSE) }
else { if (""=="ipa") {
ad<-principal(co, nfactors = pr, rotate="none", n.obs=ncase, scores=FALSE)
ad3 <- iterativePrincipalAxis(co, nFactors=pr, communalities="", iterations=200,
tolerance=1e-6)
ad$loadings <- ad3$loadings }
else { if (""=="nipa") {
ad<-principal(co, nfactors = pr, rotate="none", n.obs=ncase, scores=FALSE)
ad3 <- principalAxis(co, nFactors=pr, communalities="")
ad$loadings <- ad3$loadings }
else {ad <- fa(co, nfactors=pr, residuals = FALSE, rotate = "none", n.obs=ncase_list,
SMC=TRUE, min.err = 1e-6, digits =8, max.iter=200, symmetric=TRUE, warnings=TRUE,
fm="")
ad$loadings <- ad$loadings[,order(colnames(ad$loadings))]
 } } }
if ("a"=="a") { mat <- "Pearson" }
else { if ("a"=="b") { mat <- "Polychoric (Two Step)" }
else { if ("a"=="c") { mat <- "Polychoric (Max. Lik.)" }
```

```
else { if ("a"=="d") { mat <- "Spearman" }
else { if ("a"=="e") { mat <- "Heterogenous (Two Step)" }
else { if ("a"=="f") { mat <- "Heterogenous (Max. Lik.)" }
 } } } } }
if ("complete.obs"=="pairwise.complete.obs") { caso <- "pairwise"}
if ("complete.obs"=="complete.obs") { caso <- "listwise"}
junto <- data.frame(mat, "", caso)
names(junto) <- c("Correlation Matrix", "Factor Extraction", "Missing values")
spsspivottable.Display(junto,
title="Factor Analysis", hiderowdimlabel=TRUE)
ad$loadings <- as.matrix(ad$loadings)
row.names(ad$loadings) <- nam
load <- ad$loadings
ConFac <- load
if (""=="ipa" || ""=="nipa") {
sum2 <- rep(0,n)
for(i in 1:pr) { sum2 <- load[,i]^2 + sum2 }
ad$communality <- sum2
}
extracted <- ad$communality
b <- data.frame(extracted, row.names=nam)
spsspivottable.Display(b, title="Communalities")
# Sorting or not loadings
if (""=="no") {
supre <- load
for (j in 1:pr) {
for (i in 1:nrow(load)) {
if (abs(load[i,j]) < suprime) { supre[i,j] <- NA } } }
spsspivottable.Display(supre[,],
title="Factor Loadings (unrotated)", collabels=paste ("F",1:pr, sep="")) }
else {
ad <- fa.sort(ad)
load2 <- ad$loadings
supre <- load2
for (j in 1:pr) {
for (i in 1:nrow(load2)) {
if (abs(load2[i,j]) < suprime) { supre[i,j] <- NA } } }
spsspivottable.Display(supre[,],
title="Sorted Factor Loadings (unrotated)", collabels=paste ("F",1:pr, sep="") ) }
# Initial Eigenvalues
if (""=="pca" || ""=="ipa" || ""=="nipa") {
Eigenvalues <- ad$values }
else { Eigenvalues <- ad$e.values }
# Scree plot
if ("n"=="y") {
plotuScree(Eigenvalues, ylab = "Eigenvalues",
xlab = "Components",
main = "Scree Plot") }
sume  <- 0
sum2 <- rep(0,n)
for (i in 1:n) { sume <- sume  + Eigenvalues[i]
```

```
sum2[i] = sum2[i] + sume }
junto <- data.frame(Eigenvalues,Eigenvalues/sume*100,sum2/sume*100)
names(junto) <- c("Eigenvalues","% of Variance","Cumulative %")
spsspivottable.Display(junto,
title=" Variance Explained (initial)")
# Calculation of Sums of Squared Loadings
pro <- rep(0,pr)
for (j in 1:pr) { sum3 <- 0
for (i in 1:n) { sum3<- sum3 + load[i,j]^2 }
pro[j] <- sum3 }
sum3 <- 0
sum2 <- rep(0,pr)
for (i in 1:pr) { sum3<- sum3 + pro[i]
sum2[i] = sum2[i] + sum3 }
junto <- data.frame(pro,pro/sume*100,sum2/sume*100)
names(junto) <- c("Sums of Squared Loadings","% of Variance","Cumulative %")
spsspivottable.Display(junto,
title="Variance Explained (extracted)")
# Factor plot unrotated
if ("n"=="y") {
l1 <-
l2 <-
ll1 <- min(l1,l2); ll2 <- max(l1,l2)
if (ll1 < ll2 && ll2 < (pr+1)) {
x <-matrix(c(load[,ll1],load[,ll2]),ncol=2)
x1 <- paste("Factor", ll1, sep=" ")
y1 <- paste("Factor", ll2, sep=" ")
factor.plot(x, cut = , labels=nam, xlim = c(-1, 1), ylim = c(-1, 1), xlab=x1, ylab=y1, title =
"Factor Plot (initial solution)")  } }
# Factor diagram unrotated
if ("n"=="y") {
fa.diagram(ad, sort=, labels=NULL, cut=, simple=, errors=TRUE, digits=2, e.size=0.05,
rsize=0.15, side=2, main="Factor Diagram (initial solution)", cex=NULL) }
# Rotations
if (""!="none" || ""=="yes") {
if (""=="varimax") {ad2 <- Varimax(load, Tmat=diag(ncol(load)), normalize=, eps=1e-5,
maxit=500) }
else { if (""=="quartimax") {ad2 <- quartimax(load, Tmat=diag(ncol(load)), normalize=,
eps=1e-5, maxit=500) }
else { if (""=="quartimin") {ad2 <- quartimin(load, Tmat=diag(ncol(load)), normalize=,
eps=1e-5, maxit=500) }
else { if (""=="equamax") {ad2 <- cfT(load, Tmat=diag(ncol(load)),
kappa=pr/(2*nrow(load)), normalize=, eps=1e-5, maxit=500) }
else { if (""=="parsimax") {ad2 <- cfT(load, Tmat=diag(ncol(load)), kappa=(pr-
1)/(nrow(load)+pr-2), normalize=, eps=1e-5, maxit=500) }
else { if (""=="factor.parsimony") {ad2 <- cfT(load, Tmat=diag(ncol(load)), kappa=1,
normalize=, eps=1e-5, maxit=500) }
else { if (""=="biquartimin") { ad2 <- oblimin(load, Tmat=diag(ncol(load)), gam=0.5,
normalize=, eps=1e-5, maxit=500) }
else { if (""=="covarimin") { ad2 <- oblimin(load, Tmat=diag(ncol(load)), gam=1,
normalize=, eps=1e-5, maxit=500) }
```

```
else { if (""=="oblimax") {ad2 <- oblimax(load, Tmat=diag(ncol(load)), normalize=, eps=1e-5, maxit=500) }
else { if (""=="entropy") {
kk <-
if (""=="") { kk <- ncol(load)*nrow(load) - nrow(load) }
ad2 <- entropy(load, Tmat=diag(ncol(load)), normalize=, eps=1e-5, maxit=500) }
else { if (""=="simplimax") {ad2 <- simplimax(load, Tmat=diag(ncol(load)), k=, normalize=, eps=1e-5, maxit=500) }
else { if (""=="bentlerT") {ad2 <- bentlerT(load, Tmat=diag(ncol(load)), normalize=, eps=1e-5, maxit=500) }
else { if (""=="bentlerQ") {ad2 <- bentlerQ(load, Tmat=diag(ncol(load)), normalize=, eps=1e-5, maxit=500) }
else { if (""=="tandemI") {ad2 <- tandemI(load, Tmat=diag(ncol(load)), normalize=, eps=1e-5, maxit=500) }
else { if (""=="tandemII") {ad2 <- tandemII(load, Tmat=diag(ncol(load)), normalize=, eps=1e-5, maxit=500) }
else { if (""=="geominT") {ad2 <- geominT(load, Tmat=diag(ncol(load)), delta=0.01, normalize=, eps=1e-5, maxit=500) }
else { if (""=="geominQ") {ad2 <- geominQ(load, Tmat=diag(ncol(load)), delta=0.01, normalize=, eps=1e-5, maxit=500) }
else { if (""=="infomaxT") {ad2 <- infomaxT(load, Tmat=diag(ncol(load)), normalize=, eps=1e-5, maxit=500) }
else { if (""=="infomaxQ") {ad2 <- infomaxQ(load, Tmat=diag(ncol(load)), normalize=, eps=1e-5, maxit=500) }
else { if (""=="mccammon") {ad2 <- mccammon(load, Tmat=diag(ncol(load)), normalize=, eps=1e-5, maxit=500) }
} } } } } } } } } } } } } } } } } } }
prom<-0
# Promax rotation (adapted from Promax function from psych package)
if (""!="none") {  promai <- ad2$orthogonal } else { promai <- FALSE }
if ((promai) || (""=="none")) {
if (""=="yes") {
m <-
if (ncol(load) < 2) { return(load)}
dn <- dimnames(load)
if (""!="none") { load <- ad2$loadings }
Q <- load * abs(load)^(m - 1)
U <- lm.fit(load, Q)$coefficients
d <- diag(solve(t(U) %*% U))
U <- U %*% diag(sqrt(d))
dimnames(U) <- NULL
z <- load %*% U
if (""!="none") {U <- ad2$Th %*% U}
ui <- solve(U)
Phi <- ui %*% t(ui)
dimnames(z) <- dn
prom <-1
ad2<- list(loadings = z, Th = U%*%Phi, Phi = Phi, orthogonal = FALSE, method ="promax", initial="") } }
load <- ad2$loadings   # if oblique rotation, this is the pattern matrix
 # Sorting or not loadings after rotation
```

```
if (""=="no") {
if (ad2$orthogonal == FALSE) {
supre <- load
for (j in 1:pr) {
for (i in 1:nrow(load)) {
if (abs(load[i,j]) < suprime) { supre[i,j] <- NA } } }
spsspivottable.Display(supre[,], title="Pattern Matrix",
collabels=paste ("F",1:pr, sep="") )
p <- as.matrix(load[,])%*%ad2$Phi
supre <- p
for (j in 1:pr) {
for (i in 1:nrow(p)) {
if (abs(p[i,j]) < suprime) { supre[i,j] <- NA } } }
spsspivottable.Display(supre[,],
title="Structure Matrix") }
else {
supre <- load
for (j in 1:pr) {
for (i in 1:nrow(load)) {
if (abs(load[i,j]) < suprime) { supre[i,j] <- NA } } }
spsspivottable.Display(supre[,],
title="Rotated Factor Loadings",
collabels=paste ("F",1:pr, sep="") ) } }
else {
ad$loadings <- ad2$loadings
ad <- fa.sort(ad)
load2 <- ad$loadings
supre <- load2
for (j in 1:pr) {
for (i in 1:nrow(load2)) {
if (abs(load2[i,j]) < suprime) { supre[i,j] <- NA } } }
if ( ! ad2$orthogonal) { spsspivottable.Display(supre[,],
title="Sorted Pattern Matrix",
collabels=paste ("F",1:pr, sep="") )
p <- as.matrix(load2[,])%*%ad2$Phi
supre <- p
for (j in 1:pr) {
for (i in 1:nrow(p)) {
if (abs(p[i,j]) < suprime) { supre[i,j] <- NA } } }
spsspivottable.Display(supre[,],
title="Sorted Structure Matrix",
collabels=paste ("F",1:pr, sep="") ) }
else {
supre <- load2
for (j in 1:pr) {
for (i in 1:nrow(load2)) {
if (abs(load2[i,j]) < suprime) { supre[i,j] <- NA } } }
spsspivottable.Display(supre[,],
title="Sorted Rotated Factor Loadings",
collabels=paste ("F",1:pr, sep="") ) } }
# Calculation of Sums of Squared Loadings after rotation
```

```
if (ad2$orthogonal) {
pro <- rep(0,pr)
for (j in 1:pr) { sum3 <- 0
for (i in 1:n) { sum3<- sum3 + load[i,j]^2 }
pro[j] <- sum3 }
sum3 <- 0
sum2 <- rep(0,pr)
for (i in 1:pr) { sum3<- sum3 + pro[i]
sum2[i] = sum2[i] + sum3 }
junto <- data.frame(pro,pro/sume*100,sum2/sume*100)
names(junto) <- c("Sums of Squared Loadings","% of Variance","Cumulative %")
spsspivottable.Display(junto,
title="Rotated Variance Explained (extracted)") }
else {
pro <- rep(0,pr)
for (j in 1:pr) { sum3 <- 0
for (i in 1:n) { sum3<- sum3 + p[i,j]^2 }
pro[j] <- sum3 }
junto <- data.frame(pro)
names(junto) <- c("Sums of Squared Loadings")
spsspivottable.Display(junto,
title="Rotation Sums of Squared Loadings from Structure Matrix") }
spsspivottable.Display(ad2$Th,
title="Rotation Matrix",
collabels=paste ("F",1:pr, sep=""),
rowlabels=paste ("F",1:pr, sep="") )
spsspivottable.Display(ad2$Phi,
title="Correlation matrix of rotated factors",
collabels=paste ("F",1:pr, sep=""),
rowlabels=paste ("F",1:pr, sep="") )
if (prom==0) {
junto <- data.frame(ad2$method,ad2$orthogonal,ad2$convergence)
names(junto) <- c("method","orthogonal","convergence")
spsspivottable.Display(junto,
title="Rotation", hiderowdimlabel=TRUE) }
else { junto <- data.frame(ad2$method,ad2$orthogonal, ad2$initial)
names(junto) <- c("method","orthogonal","initial rotation")
spsspivottable.Display(junto,
title="Rotation", hiderowdimlabel=TRUE) }
# Factor plot after rotation
if ("n"=="y") {
l1 <-
l2 <-
ll1 <- min(l1,l2); ll2 <- max(l1,l2)
if (ll1 < ll2 && ll2 < (pr+1)) {
x <-matrix(c(load[,ll1],load[,ll2]), ncol=2)
x1 <- paste("Factor", ll1, sep=" ")
y1 <- paste("Factor", ll2, sep=" ")
factor.plot(x, cut = , labels=nam,  xlim = c(-1, 1), ylim = c(-1, 1), xlab=x1, ylab=y1, title =
"Factor Plot (rotated solution)")  } }
# Factor diagram after rotation
```

```
if ("n"=="y") {
ad$loadings <- ad2$loadings
colnames(ad$loadings) <- paste("F", 1:pr, sep = "")
fa.diagram(ad, sort=, labels=NULL, cut=, simple=, errors=TRUE, digits=2, e.size=0.05,
rsize=0.15, side=2, main="Factor Diagram (rotated solution)", cex=NULL) }
 }   #end of rotation
# KMO
if ("n"=="y") {
g <- diag(0,n)
par <- cor2pcor(co)
cco <- co
cco[upper.tri(cco, TRUE)] <- g[upper.tri(cco, TRUE)]
par[upper.tri(par, TRUE)] <- g[upper.tri(par, TRUE)]
cco <- sum(cco^2)
kmo <- cco/(sum(par^2)+cco)
KMO <- c(kmo)
b <- data.frame(KMO)
spsspivottable.Display(b,
title="KMO - Kaiser-Meyer-Olkin measure of sampling adequacy",
hiderowdimlabel=TRUE) }
# MSA
if ("n"=="y") {
par <- cor2pcor(co)
ms <- rep(0,n)
for (i in 1:n) { ms[i] <- ( sum(co[i,]^2)-1 )  / (sum(co[i,]^2)+sum(par[i,]^2)-2) }
MSA <- c(ms)
b <- data.frame(MSA, row.names=nam)
spsspivottable.Display(b,
title="MSA - Measures of sampling adequacy") }
# Residual matrix for extraction using NFactors package
if (""=="ipa" || ""=="nipa") {
res <- diag(0,n)
for (i in 2:n) {
for (j in 1:(i-1)) {
res [i,j] <- co[i,j] - sum(ConFac[i,]*ConFac[j,]) } }
junto <- res + t(res) + diag(1 - ad$communality)
row.names(junto) <- nam
names(junto) <- nam
ad$residual <- junto
junto <- data.frame(junto) }
# GFI and AGFI
if ("n"=="y") {
s <- ad$residual
som <- sum(diag(s%*%s))
som2 <- sum(diag(co%*%co))
som <- 1-som/som2
b <- data.frame(som)
names(b) <- c("GFI (ULS)")
spsspivottable.Display(b,
title="GFI (Goodness of Fit)", hiderowdimlabel=TRUE) }
if ("n"=="y") {
```

```
aus <- matrix(0,n,n)
for (i in 1:(n-1)) {
for (j in (i+1):n) {
aus[i,j] <- sum(ConFac[i,]*ConFac[j,]) } }
aus2 <- t(aus)
diag(aus2) <- 1
aus <- aus + aus2
som <- sum(diag((solve(aus)%*%co-diag(n))%*%(solve(aus)%*%co-diag(n))))
som2 <- sum(diag(solve(aus)%*%co%*%solve(aus)%*%co))
som <- 1-som/som2
b <- data.frame(som)
names(b) <- c("GFI (ML)")
spsspivottable.Display(b,
title="GFI (Goodness of Fit)", hiderowdimlabel=TRUE) }
# RMSR
if ("n"=="y") {
g <- diag(0,n)
s <- ad$residual
s[upper.tri(s, TRUE)] <- g[upper.tri(s, TRUE)]
rmsr <- sqrt(2*sum(s^2)/(n*(n-1)))
RMSR <- c(rmsr)
b <- data.frame(RMSR)
spsspivottable.Display(b,
title="RMSR (Root Mean Square Residual)",
hiderowdimlabel=TRUE) }
# Root Mean Square Partials Correlations Controlling Factors
if ("n"=="y") {
cc <- co-(ConFac%*%t(ConFac))
ca <- sqrt(diag(cc))
d <- diag(1/ca)
ea <- d%*%cc%*%d   #partial correlations controlling factors
rmsp <- sqrt((sum(ea^2)-n)/(n*(n-1)))
RMSP <- c(rmsp)
b <- data.frame(RMSP)
spsspivottable.Display(b,
title="Root mean square partial correlations controlling factors",
hiderowdimlabel=TRUE) }
# Partial Correlations controlling all other variables
if ("n"=="y") {
par <- cor2pcor(co)
row.names(par) <- nam
par <- data.frame(par)
names(par) <- nam
spsspivottable.Display(par,
title="Partial correlations controlling all other variables") }
# Partial Correlations controlling Factors
if ("n"=="y") {
cc <- co-(ConFac%*%t(ConFac))
ca <- sqrt(diag(cc))
d <- diag(1/ca)
ea <- d%*%cc%*%d   #partial correlations controlling factors
```

```
row.names(ea) <- nam
ea <- data.frame(ea)
names(ea) <- nam
spsspivottable.Display(ea,
title="Partial correlations controlling factors") }
# Residual correlations
if (FALSE==TRUE) {
spsspivottable.Display(ad$residual,
title="Residual correlations with uniqueness on the diagonal (computed between observed and
reproduced correlations)")
cont <- 0
s <- ad$residual
for (i in 2:n) { for (j in 1:(i-1)) { if (abs(s[i,j])> 0.05) { cont <- cont+1 } } }
p <- 2*cont/(n*(n-1))*100
table <- spss.BasePivotTable("Fit of the model to the correlation matrix","OMS")
rowdim <- BasePivotTable.Append(table,Dimension.Place.row,"")
coldim <- BasePivotTable.Append(table,Dimension.Place.column,"Residual fit values")
row1 <- spss.CellText.String("Values")
col1 <- spss.CellText.String("Residuals > 0.05")
col2 <- spss.CellText.String("% residuals > 0.05")
BasePivotTable.SetCategories(table,rowdim,list(row1))
BasePivotTable.SetCategories(table,coldim,list(col1,col2))
BasePivotTable.SetCellsByColumn(table,col1,list(spss.CellText.Number(cont,6)))
BasePivotTable.SetCellsByColumn(table,col2,list(spss.CellText.Number(p))) }
# Determinant
if ("n"=="y") {
b <- determinant(co, logarithm = FALSE)
junto <- data.frame(b$modulus)
names(junto) <- c("Determinant")
spsspivottable.Display(junto,
title="Determinant (modulus)",
hiderowdimlabel=TRUE) }
# Tests to correlation matrix
if ("n"=="y") {
bar <- cortest.bartlett(co, n = ncase)
bar2 <- cortest.normal(co,n1=ncase, fisher = FALSE)
bar3 <- cortest.jennrich(co, diag(rep(1,n)), n1 = ncase, n2=ncase)
table <- spss.BasePivotTable("Tests of whether a correlation matrix is an identity
matrix","OMS")
rowdim <- BasePivotTable.Append(table,Dimension.Place.row,"")
coldim <- BasePivotTable.Append(table,Dimension.Place.column,"Chisquare tests")
row1 <- spss.CellText.String("Bartlett's Test")
row2 <- spss.CellText.String("Steiger Test")
row3 <- spss.CellText.String("Jennrich Test")
col1 <- spss.CellText.String("Chisquare")
col2 <- spss.CellText.String("Degrees of freedom")
col3<- spss.CellText.String("p-value")
BasePivotTable.SetCategories(table,rowdim,list(row1,row2, row3))
BasePivotTable.SetCategories(table,coldim,list(col1,col2,col3))
BasePivotTable.SetCellsByColumn(table,col1,list(spss.CellText.Number(bar$chisq),spss.Cell
Text.Number(bar2$chi2),spss.CellText.Number(bar3$chi2)))
```

```
BasePivotTable.SetCellsByColumn(table,col2,list(spss.CellText.Number(bar$df,6),spss.CellT
ext.Number(bar2$df,6),spss.CellText.Number(bar2$df,6)))
BasePivotTable.SetCellsByColumn(table,col3,list(spss.CellText.Number(bar$p.value),spss.C
ellText.Number(bar2$prob),spss.CellText.Number(bar3$prob))) }
# Chisquare test for Maximum Likelihood
if (""=="ml") {
table <- spss.BasePivotTable("Chisquare test for Maximum Likelihood procedure","OMS")
rowdim <- BasePivotTable.Append(table,Dimension.Place.row,"")
coldim <- BasePivotTable.Append(table,Dimension.Place.column,"Chisquare test")
row1 <- spss.CellText.String("Test")
col1 <- spss.CellText.String("Chisquare")
col2 <- spss.CellText.String("Degrees of freedom")
col3<- spss.CellText.String("p-value")
BasePivotTable.SetCategories(table,rowdim,list(row1))
BasePivotTable.SetCategories(table,coldim,list(col1,col2,col3))
BasePivotTable.SetCellsByColumn(table,col1,list(spss.CellText.Number(ad$STATISTIC)))
BasePivotTable.SetCellsByColumn(table,col2,list(spss.CellText.Number(ad$dof,6)))
if (is.numeric(ad$PVAL)) {
BasePivotTable.SetCellsByColumn(table,col3,list(spss.CellText.Number(ad$PVAL))) }
else { BasePivotTable.SetCellsByColumn(table,col3,list(spss.CellText.String(ad$PVAL))) }
}
spsspkg.EndProcedure() }
END PROGRAM.
BEGIN PROGRAM R.
# Number of Factors
if ("no"=="yes" || "y"=="y" || "n"=="y") {
if ("e"=="a") {
if (m1==0) { cor1 <- cor(mdata, use="complete.obs", method="pearson"); m1 <-1 }
co <- cor1 }
if ("e"=="b") {
if (m2==0) { co <- hetcor(da,  ML = FALSE, std.err = FALSE, pd=TRUE,
use="complete.obs", digits=6); m2 <-1
cor2 <- co$correlations }
co <- cor2 }
if ("e"=="c") {
if (m3==0) { co <- hetcor(da,  ML = TRUE, std.err = FALSE, pd=TRUE,
use="complete.obs", digits=6); m3 <-1
cor3 <- co$correlations }
co <- cor3 }
if ("e"=="d") {
if (m4==0) { cor4 <- cor(mdata, use="complete.obs", method="spearman"); m4 <-1 }
co <- cor4 }
if ("e"=="e") {
if (m5==0) { co <- hetcor(dah,  ML = FALSE, std.err = FALSE, pd=TRUE,
use="complete.obs", digits=6); m5 <-1
cor5 <- co$correlations }
co <- cor5 }
if ("e"=="f") {
if (m6==0) { co <- hetcor(dah,  ML = TRUE, std.err = FALSE, pd=TRUE,
use="complete.obs", digits=6); m6 <-1
cor6 <- co$correlations }
```

```
co <- cor6 }
spsspkg.StartProcedure ("Number of Components/Factors")
if ("e"=="a") { mat <- "Pearson" }
if ("e"=="b") { mat <- "Polychoric (Two Step)" }
if ("e"=="c") { mat <- "Polychoric (Max. Lik.)" }
if ("e"=="d") { mat <- "Spearman" }
if ("e"=="e") { mat <- "Heterogenous (Two Step)" }
if ("e"=="f") { mat <- "Heterogenous (Max. Lik.)" }
if ("complete.obs"=="pairwise.complete.obs") { caso <- "pairwise"}
if ("complete.obs"=="complete.obs") { caso <- "listwise"}
junto <- data.frame(mat, caso)
names(junto) <- c("Correlation Matrix", "Missing values")
spsspivottable.Display(junto,
title="Number of Components/Factors estimated for the following Correlation Matrix",
hiderowdimlabel=TRUE) }
# Cattel Scree Test
if ("no"=="yes" && n >= 3) {
if (""!="") { set.seed() }
repi <-
evpea <- matrix(NA, ncol = n, nrow = repi)
evpea_b <- matrix(NA, ncol = n, nrow = repi)
# Eigenvalues from components or factors
if (""=="components") {
evt <- eigen(co, only.values = TRUE) }
else { evt <- eigen(co - ginv(diag(diag(ginv(co)))), only.values=TRUE) }
# Simulations
quant <- function(x, sprobs = sprobs) { return(as.vector(quantile(x, probs = c()))) }
# standardized normal
if ("" == "normal") {
co_used <- "Pearson"
for (k in 1:repi) {
y <- mvrnorm(ncase, rep(0, n), diag(1,n), empirical = FALSE)
corY <- cor(y, use="complete.obs", method="pearson")
if (""=="factors") { corY <- corY - ginv(diag(diag(ginv(corY)))) }
ev <- eigen(corY, only.values = TRUE)
evpea[k, ] <- ev$values }
mevpea <- sapply(as.data.frame(evpea), mean)
qevpea <- sapply(as.data.frame(evpea), quant)
ap <- list(eigen = data.frame(mevpea, qevpea)) }
# permutation
if ("" == "permutation") {
sdata <- matrix (0,ncol = ncol(mdata), nrow =ncase_list)
if (""=="y") {
mdata2 <- as.matrix(mdata)
lcor <- 0
for (i in 1:ncase2) {
ind <- 0
j <- 0
while (j<n && ind ==0) {
j <- j+1
if (is.na(mdata[i,j])) { ind <- 1 } }
```

```
if (ind==0) { lcor <- lcor+1; sdata[lcor,] <- mdata2[i,] } } }
else { sdata <- mdata }
for (k in 1:repi) {
perm <- apply(sdata[,2:n], 2, sample, replace = )
perm <- data.frame(sdata[,1],perm)
if (""=="a") { corY <- cor(perm, use="complete.obs", method="pearson"); co_used <-
"Pearson" }
else { if (""=="b") {
co_used <- "Polychoric (Two Step)"
for (i in 1:n) { perm[,i]<-ordered(perm[,i]) }
corY <-  hetcor(perm,  ML = FALSE, std.err = FALSE, pd=TRUE, use="complete.obs",
digits=4)
corY <- corY$correlations }
else { if (""=="c") {
co_used <- "Polychoric (Max. Likelihood)"
for (i in 1:n) { perm[,i]<-ordered(perm[,i]) }
corY <-  hetcor(perm,  ML = TRUE, std.err = FALSE, pd=TRUE, use="complete.obs",
digits=4)
corY <- corY$correlations }
else { if (""=="e") {
co_used <- "Heterogeneous (Two Step)"
for (i in 1:n) {
if  ( scal["varMeasurementLevel",i]=="nominal") { perm[,i]<-factor(perm[,i]) }
else { if  (scal["varMeasurementLevel",i]=="ordinal") { perm[,i]<-ordered(perm[,i]) } } }
corY <-  hetcor(perm,  ML = FALSE, std.err = FALSE, pd=TRUE, use="complete.obs",
digits=4)
corY <- corY$correlations }
else { if (""=="f") {
co_used <- "Heterogeneous (Max. Likelihood)"
for (i in 1:n) {
if  ( scal["varMeasurementLevel",i]=="nominal") { perm[,i]<-factor(perm[,i]) }
else { if  (scal["varMeasurementLevel",i]=="ordinal") { perm[,i]<-ordered(perm[,i]) } } }
corY <-  hetcor(perm,  ML = TRUE, std.err = FALSE, pd=TRUE, use="complete.obs",
digits=4)
corY <- corY$correlations }
else { corY <- cor(perm, use="complete.obs", method="spearman"); co_used <- "Spearman"
} } } } }
if (""=="factors") { corY <- corY - ginv(diag(diag(ginv(corY)))) }
ev <- eigen(corY, only.values = TRUE)
evpea[k, ] <- ev$values }
mevpea <- sapply(as.data.frame(evpea), mean)
qevpea <- sapply(as.data.frame(evpea), quant)
ap <- list(eigen = data.frame(mevpea, qevpea) ) }
# parallel analysis and scree
nS <- nScree(evt$values, aparallel=)
s <- ""
if ( s=="ap$eigen$mevpea") { ss <- "mean" } else { ss <- "quantile" }
if (""=="factors") { criteria <- mean(evt$values)
nS$Components$nkaiser <- sum(evt$values >= rep(criteria, n))
p.vec <- which(evt$values >= ap$eigen$qevpea)
nS$Components$nparallel <- sum(p.vec == (1:length(p.vec))) }
```

```
junto <- data.frame(repi, "", "", ss, mat, co_used)
names(junto) <- c("Number of Samples","Model","Data","Measure",
"Correlation matrix","Matrix for simulations")
spsspivottable.Display(junto,
title="Parameters for Parallel Analysis",
hiderowdimlabel=TRUE, format=6)
junto <- data.frame(paste("",1:n, rep=""), ap$eigen)
names(junto) <- c("Order", "Mean", "Quantile selected")
spsspivottable.Display(junto,
title="Distribution of the eigenvalues computed",
hiderowdimlabel=TRUE)
junto <- data.frame(nS$Components)
names(junto) <- c("Optimal coordinates","Acceleration factor",
"Parallel analysis","Kaiser rule")
spsspivottable.Display(junto,
title="Number of components/factors to retain according to different rules",
hiderowdimlabel=TRUE, format=6)
junto <- data.frame(nS$Analysis)
names(junto) <- c("Eigenvalues","Proportion of variance", "Cumulative", "Parallel analysis",
"Predicted eigenvalues", "OC", "Acceleration factor", "AF")
spsspivottable.Display(junto,
title="Data linked to the different rules")
if (""=="components") {
if (""=="normal") {
plotnScree(nS, ylab = "Eigenvalues", xlab = "Components",
main = "Parallel analysis on random uncorrelated standardized normal") }
if (""=="permutation") {
plotnScree(nS, ylab = "Eigenvalues", xlab = "Components",
main = "Parallel analysis on data permutation") } }
else {
if (""=="normal") {
plotnScree(nS, ylab = "Eigenvalues", xlab = "Factors",
main = "Parallel analysis on random uncorrelated standardized normal") }
if (""=="permutation") {
plotnScree(nS, ylab = "Eigenvalues", xlab = "Factors",
main = "Parallel analysis on data permutation") } } }
# Velicer MAP
if ("y"=="y") {
ev <- eigen(co, symmetric=TRUE)
load2 <- ev$vectors%*%sqrt(diag(ev$values))
f <- rep(0,(n-1)); fq <- rep(0,(n-1))
som <- sum(co^2); somq <- sum(co^4)
m <- n*(n-1); f[1] <- (som-n)/m; fq[1] <- (somq-n)/m
for (i in 1:(n-2)) { b <- load2[,1:i]; cc <- co-(b%*%t(b))
d <- diag(1/sqrt(diag(cc))); ea <- d%*%cc%*%d      #partial correlation matrix controlling
components
som <- sum(ea^2); somq <- sum(ea^4)
f[i+1] <- (som-n)/m; fq[i+1] <- (somq-n)/m }
matt <- replace(f, f == "NaN", 2)
matt2 <- replace(fq, fq == "NaN", 2)
fm <- min(matt); fqm <- min(matt2)
```

```
for (i in 1:(n-1)) {
if (f[i]==fm) { fma <- i-1; break } }
for (i in 1:(n-1)) {
if (fq[i]==fqm) { fqma <- i-1; break } }
junto <- data.frame(f[1:(n-1)],fq[1:(n-1)],row.names=0:(n-2))
names(junto) <- c("Squared average partial correlations", "4th average partial correlations")
spsspivottable.Display(junto,
title="Velicer's MAP values")
table <- spss.BasePivotTable("Velicer's Minimum Average Partial Test","OMS")
rowdim <- BasePivotTable.Append(table,Dimension.Place.row,"")
coldim <- BasePivotTable.Append(table,Dimension.Place.column,"Velicer's Minimum")
row1 <- spss.CellText.String("Squared MAP")
row2 <- spss.CellText.String("4th power MAP")
col1 <- spss.CellText.String("Minimum")
col2 <- spss.CellText.String("Components to retain")
BasePivotTable.SetCategories(table,rowdim,list(row1, row2))
BasePivotTable.SetCategories(table,coldim,list(col1,col2))
BasePivotTable.SetCellsByRow(table,row1,list(spss.CellText.Number(fm),
spss.CellText.Number(fma,6)))
BasePivotTable.SetCellsByRow(table,row2,list(spss.CellText.Number(fqm),
spss.CellText.Number(fqma,6))) }
# Very Simple Structure
if ("n"=="y") {
v <- VSS(co, n = , rotate = "", diagonal = , fm = "", n.obs=ncase, plot=TRUE)
junto <- data.frame(v$cfit.1, v$cfit.2)
names(junto) <- c("Complexity 1","Complexity 2")
spsspivottable.Display(junto,
title="Very Simple Structure")
nu <-
vss1 <- 0
for (i in 1:nu) {
vss1 <- vss1+1
if ( v$cfit.1[i]==max(v$cfit.1) ) { break }
 }
vss2 <- 0
for (i in 1:nu) {
vss2 <- vss2+1
if ( v$cfit.2[i]==max(v$cfit.2) ) { break }
 }
table <- spss.BasePivotTable("Very Simple Structure (Number of factors)","OMS")
rowdim <- BasePivotTable.Append(table,Dimension.Place.row,"")
coldim <- BasePivotTable.Append(table,Dimension.Place.column,"Very Simple Structure
(VSS)")
row1 <- spss.CellText.String("Values")
col1 <- spss.CellText.String("Rotation")
col2 <- spss.CellText.String("Factoring method")
col3 <- spss.CellText.String("Max VSS complexity 1")
col4 <- spss.CellText.String("N factors complexity 1")
col5 <- spss.CellText.String("Max VSS complexity 2")
col6 <- spss.CellText.String("N factors complexity 2")
BasePivotTable.SetCategories(table,rowdim,list(row1))
```

```
BasePivotTable.SetCategories(table,coldim,list(col1,col2,col3,col4,col5,col6))
BasePivotTable.SetCellsByColumn(table,col1,list(spss.CellText.String("")))
BasePivotTable.SetCellsByColumn(table,col2,list(spss.CellText.String("")))
BasePivotTable.SetCellsByColumn(table,col3,list(spss.CellText.Number(max(v$cfit.1))))
BasePivotTable.SetCellsByColumn(table,col4,list(spss.CellText.Number(vss1,6)))
BasePivotTable.SetCellsByColumn(table,col5,list(spss.CellText.Number(max(v$cfit.2))))
BasePivotTable.SetCellsByColumn(table,col6,list(spss.CellText.Number(vss2,6)))
 }
spsspkg.EndProcedure()
END PROGRAM.
BEGIN PROGRAM R.
# More Number of Factors
if("no"=="yes") {
spsspkg.StartProcedure ("Comparison Data")
N.Pop <-
sdata <- matrix (0,ncol = ncol(mdata), nrow =ncase_list)
if (""=="yes") {
mdata2 <- as.matrix(mdata)
lcor <- 0
for (i in 1:ncase2) {
ind <- 0
j <- 0
while (j<n && ind ==0) {
j <- j+1
if (is.na(mdata[i,j])) { ind <- 1 } }
if (ind==0) { lcor <- lcor+1; sdata[lcor,] <- mdata2[i,] } } }
else { sdata <- mdata }
# Matrix to store each variable score distribution
Distributions <- matrix(0, nrow = N.Pop, ncol = dim(sdata)[2])
# Generate distribution for each variable
if (""!="") { set.seed() }
b2 <- matrix(0, nrow = dim(sdata)[2], ncol = 1)
for (i in 1:dim(sdata)[2]) {
bb <- sort(sample(sdata[,i], size = N.Pop, replace = T) )
b2[i] <- length(bb)
ac <- N.Pop-b2[i]
if (b2[i] < N.Pop) {
for (j in 1:ac) {
qw <- sample(1: (b2[i] + j),1)
bb <- insert (bb, qw, NA) } }
Distributions[,i] <- as.matrix(bb) }
EFA.Comp.Data <- function(Data, F.Max, N.Samples=500, Alpha = 0.3) {
# Data = N by k data matrix
# F.Max = largest number of factors to consider
# N.Pop = size of finite populations of comparison data
# N.Samples = number of samples drawn from each population
# Alpha = alpha level testing significance of improvement with adding factor
sig2 <- data.frame(1)
N <- dim(Data)[1]
k <- dim(Data)[2]
Data2 <- as.data.frame(Data)
```

```
if (""=="a") { cor.Data <- cor(Data,use="complete.obs") }
if (""=="b") {
for (i in 1:k) { Data2[,i]<-ordered(Data[,i])  }
corY <-  hetcor(Data2,  ML = FALSE, std.err = FALSE, pd=TRUE, use="complete.obs",
digits=4)
cor.Data <- corY$correlations }
if (""=="c") {
for (i in 1:k) { Data2[,i]<-ordered(Data[,i])  }
corY <-  hetcor(Data2,  ML = TRUE, std.err = FALSE, pd=TRUE, use="complete.obs",
digits=4)
cor.Data <- corY$correlations }
if (""=="d") { cor.Data <- cor(Data,use="complete.obs",method="spearman") }
if (""=="e") {
for (i in 1:k) {
if  ( scal["varMeasurementLevel",i]=="nominal") { Data2[,i]<-factor(Data[,i]) }
else { if  (scal["varMeasurementLevel",i]=="ordinal") { Data2[,i]<-ordered(Data[,i]) } } }
corY <-  hetcor(Data2,  ML = FALSE, std.err = FALSE, pd=TRUE, use="complete.obs",
digits=4)
cor.Data <- corY$correlations }
if (""=="f") {
for (i in 1:k) {
if  ( scal["varMeasurementLevel",i]=="nominal") { Data2[,i]<-factor(Data[,i]) }
else { if  (scal["varMeasurementLevel",i]=="ordinal") { Data2[,i]<-ordered(Data[,i]) } } }
corY <-  hetcor(Data2,  ML = TRUE, std.err = FALSE, pd=TRUE, use="complete.obs",
digits=4)
cor.Data <- corY$correlations }
Eigs.Data <- eigen(cor.Data)$values
RMSR.Eigs <- matrix(0, nrow = N.Samples, ncol = F.Max)
Sig <- T
F.CD <- 1
nf <- -1
while (F.CD <= F.Max) {
Pop <- GenData(Data, N.Factors = F.CD, N = N.Pop, Target.Corr = cor.Data)
for (j in 1:N.Samples) {
Samp <- Pop[sample(1:N.Pop, size = N, replace = T),]
Samp2 <- as.data.frame(Samp)
if (""=="a") { cor.Samp <- cor(Samp,use="complete.obs") }
if (""=="b") {
for (i in 1:k) { Samp2[,i]<-ordered(Samp[,i])  }
corY <-  hetcor(Samp2,  ML = FALSE, std.err = FALSE, pd=TRUE, use="complete.obs",
digits=4)
cor.Samp <- corY$correlations }
if (""=="c") {
for (i in 1:k) { Samp2[,i]<-ordered(Samp[,i])  }
corY <-  hetcor(Samp2,  ML = TRUE, std.err = FALSE, pd=TRUE, use="complete.obs",
digits=4)
cor.Samp <- corY$correlations }
if (""=="d") { cor.Samp <- cor(Samp,use="complete.obs",method="spearman") }
if (""=="e") {
for (i in 1:k) {
if  ( scal["varMeasurementLevel",i]=="nominal") { Samp2[,i]<-factor(Samp[,i]) }
```

```
else { if  (scal["varMeasurementLevel",i]=="ordinal") { Samp2[,i]<-ordered(Samp[,i]) } } }
corY <-  hetcor(Samp2,  ML = FALSE, std.err = FALSE, pd=TRUE, use="complete.obs",
digits=4)
cor.Samp <- corY$correlations }
if (""=="f") {
for (i in 1:k) {
if  ( scal["varMeasurementLevel",i]=="nominal") { Samp2[,i]<-factor(Samp[,i]) }
else { if  (scal["varMeasurementLevel",i]=="ordinal") { Samp2[,i]<-ordered(Samp[,i]) } } }
corY <-  hetcor(Samp2,  ML = TRUE, std.err = FALSE, pd=TRUE, use="complete.obs",
digits=4)
cor.Samp <- corY$correlations }
Eigs.Samp <- eigen(cor.Samp)$values
RMSR.Eigs[j,F.CD] <- sqrt(sum((Eigs.Samp - Eigs.Data) * (Eigs.Samp - Eigs.Data)) / k) }
if (F.CD > 1) {
sig2[F.CD, ] <- wilcox.test(RMSR.Eigs[,F.CD], RMSR.Eigs[,(F.CD - 1)], "less")$p.value }
else { sig2[1,] <- NA }
if (F.CD > 1)  {
Sig <- (wilcox.test(RMSR.Eigs[,F.CD], RMSR.Eigs[,(F.CD - 1)], "less")$p.value < Alpha) }
F.CD <- F.CD + 1
if (Sig == FALSE && nf == -1) {
nf = F.CD - 2
if (""=="yes") { break }
}
 }
if (nf == -1) { nf = F.Max }
# graph
if (""=="yes") { x.max <- min(nf+1, F.Max) }
else { x.max <- F.Max }
ys <- apply(RMSR.Eigs[,1:x.max], 2, mean)
plot(x = 1:x.max, y = ys, ylim = c(0, max(ys)), xlab = "Factor",
ylab = "RMSR Eigenvalue", type = "b", main = "Fit to Comparison Data")
abline(v = nf, lty = 3)
lis <- list(ys = ys, nf = nf, sig2 = sig2, x.max = x.max)
return(lis)
 }
GenData <- function(Supplied.Data, N.Factors, N, Max.Trials = 5, Initial.Multiplier = 1,
Target.Corr)
{
# Ruscio, J., & Kaczetow, W. (2008).
# Simulating multivariate nonnormal data using an iterative algorithm.
# Multivariate Behavioral Research, 43(3), 355-381.
k <- dim(Supplied.Data)[2]
Iteration <- 0   # Iteration counter
Best.RMSR <- 1   # Lowest RMSR correlation
Trials.Without.Improvement <- 0   # Trial counter
Data <- matrix(0, nrow = N, ncol = k)   # Matrix to store the simulated data
# Calculate and store a copy of the target correlation matrix
Intermediate.Corr <- Target.Corr
# Generate random normal data, initialize factor loadings
Shared.Comp <- matrix(rnorm(N * N.Factors, 0, 1), nrow = N, ncol = N.Factors)
```

```
Unique.Comp <- matrix(rnorm(N.Pop * dim(sdata)[2], 0, 1), nrow = N.Pop, ncol =
dim(sdata)[2])
Shared.Load <- matrix(0, nrow = k, ncol = N.Factors)
Unique.Load <- matrix(0, nrow = k, ncol = 1)
# Begin loop that ends when specified n of iterations pass without improvement in RMSR
correlation
while (Trials.Without.Improvement < Max.Trials)  {
Iteration <- Iteration + 1
# Calculate factor loadings and apply to reproduce desired correlations
Fact.Anal <- Factor.Analysis(Intermediate.Corr, N.Factors = N.Factors)
if (N.Factors == 1) { Shared.Load[,1] <- Fact.Anal$loadings }
else {
for (i in 1:N.Factors) { Shared.Load[,i] <- Fact.Anal$loadings[,i] } }
Shared.Load[Shared.Load > 1] <- 1
Shared.Load[Shared.Load < -1] <- -1
if (Shared.Load[1,1] < 0) { Shared.Load <- Shared.Load * -1 }
for (i in 1:k) {
if (sum(Shared.Load[i,] * Shared.Load[i,]) < 1) {
Unique.Load[i,1] <- (1 - sum(Shared.Load[i,] * Shared.Load[i,])) }
else { Unique.Load[i,1] <- 0 } }
Unique.Load <- sqrt(Unique.Load)
for (i in 1:k) {
Data[,i] <- (Shared.Comp %*% t(Shared.Load))[,i] + Unique.Comp[,i] * Unique.Load[i,1] }
# Replace normal with nonnormal distributions
for (i in 1:k) {
Data <- Data[sort.list(Data[,i]),]
Data[,i] <- Distributions[,i] }
# Calculate RMSR correlation, compare to lowest value, take appropriate action
Data2 <- as.data.frame(Data)
if (""=="a") { Reproduced.Corr <- cor(Data,use="complete.obs") }
if (""=="b") {
for (i in 1:k) { Data2[,i]<-ordered(Data[,i])  }
corY <-  hetcor(Data2,  ML = FALSE, std.err = FALSE, pd=TRUE, use="complete.obs",
digits=4)
Reproduced.Corr <- corY$correlations }
if (""=="c") {
for (i in 1:k) { Data2[,i]<-ordered(Data[,i])  }
corY <-  hetcor(Data2,  ML = TRUE, std.err = FALSE, pd=TRUE, use="complete.obs",
digits=4)
Reproduced.Corr <- corY$correlations }
if (""=="d") { Reproduced.Corr <- cor(Data,use="complete.obs",method="spearman") }
if (""=="e") {
for (i in 1:k) {
if  ( scal["varMeasurementLevel",i]=="nominal") { Data2[,i]<-factor(Data[,i]) }
else { if  (scal["varMeasurementLevel",i]=="ordinal") { Data2[,i]<-ordered(Data[,i]) } } }
corY <-  hetcor(Data2,  ML = FALSE, std.err = FALSE, pd=TRUE, use="complete.obs",
digits=4)
Reproduced.Corr <- corY$correlations }
if (""=="f") {
for (i in 1:k) {
if  ( scal["varMeasurementLevel",i]=="nominal") { Data2[,i]<-factor(Data[,i]) }
```

```
else { if  (scal["varMeasurementLevel",i]=="ordinal") { Data2[,i]<-ordered(Data[,i]) } } }
corY <-  hetcor(Data2,  ML = TRUE, std.err = FALSE, pd=TRUE, use="complete.obs",
digits=4)
Reproduced.Corr <- corY$correlations }
Residual.Corr <- Target.Corr - Reproduced.Corr
RMSR <-
sqrt(sum(Residual.Corr[lower.tri(Residual.Corr)]*Residual.Corr[lower.tri(Residual.Corr)]) /
(.5* (k*k - k)))
if (RMSR < Best.RMSR)  {
Best.RMSR <- RMSR
Best.Corr <- Intermediate.Corr
Best.Res <- Residual.Corr
Intermediate.Corr <- Intermediate.Corr + Initial.Multiplier * Residual.Corr
Trials.Without.Improvement <- 0 }
else  {
Trials.Without.Improvement <- Trials.Without.Improvement + 1
Current.Multiplier <- Initial.Multiplier * .5 ^ Trials.Without.Improvement
Intermediate.Corr <- Best.Corr + Current.Multiplier * Best.Res }  }
# Construct the data set with the lowest RMSR correlation
Fact.Anal <- Factor.Analysis(Best.Corr, N.Factors = N.Factors)
if (N.Factors == 1) { Shared.Load[,1] <- Fact.Anal$loadings }
else {
for (i in 1:N.Factors) {
Shared.Load[,i] <- Fact.Anal$loadings[,i] } }
Shared.Load[Shared.Load > 1] <- 1
Shared.Load[Shared.Load < -1] <- -1
if (Shared.Load[1,1] < 0) { Shared.Load <- Shared.Load * -1 }
for (i in 1:k) {
if (sum(Shared.Load[i,] * Shared.Load[i,]) < 1)  {
Unique.Load[i,1] <- (1 - sum(Shared.Load[i,] * Shared.Load[i,])) }
else { Unique.Load[i,1] <- 0 } }
Unique.Load <- sqrt(Unique.Load)
for (i in 1:k) {
Data[,i] <- (Shared.Comp %*% t(Shared.Load))[,i] + Unique.Comp[,i] * Unique.Load[i,1] }
Data <- apply(Data, 2, scale) # standardizes each variable in the matrix
for (i in 1:k) {
Data <- Data[sort.list(Data[,i]),]
Data[,i] <- Distributions[,i]  }
# Return the simulated data set
return(Data) }
Factor.Analysis <- function(Data, Max.Iter = 50, N.Factors = 0)
{
Data <- as.matrix(Data)
k <- dim(Data)[2]
if (N.Factors == 0) {
N.Factors <- k
Determine <- T  }
else { Determine <- F }
Cor.Matrix <- Data
Criterion <- .001
Old.H2 <- rep(99, k)
```

```
H2 <- rep(0, k)
Change <- 1
Iter <- 0
Factor.Loadings <- matrix(nrow = k, ncol = N.Factors)
while ((Change >= Criterion) & (Iter < Max.Iter)) {
Iter <- Iter + 1
Eig <- eigen(Cor.Matrix)
L <- sqrt(Eig$values[1:N.Factors])
for (i in 1:N.Factors) {
Factor.Loadings[,i] <- Eig$vectors[,i] * L[i] }
for (i in 1:k) {
H2[i] <- sum(Factor.Loadings[i,] * Factor.Loadings[i,]) }
Change <- max(abs(Old.H2 - H2))
Old.H2 <- H2
diag(Cor.Matrix) <- H2
   }
if (Determine) { N.Factors <- sum(Eig$values > 1) }
return(list(loadings = Factor.Loadings[,1:N.Factors], factors = N.Factors))
}
ysa<-EFA.Comp.Data(Data = sdata, F.Max =  ,
N.Samples = , Alpha = )
junto <- data.frame(paste("",1:ysa$x.max, rep = "factor"), ysa$ys, ysa$sig2)
names(junto) <- c("Number of factors","RMSR Eigenvalue","p-value")
spsspivottable.Display(junto,
title="Fit to Comparison Data", hiderowdimlabel=TRUE )
if (""=="a") { mat <- "Pearson" }
else { if (""=="b") { mat <- "Polychoric (Two Step)" }
else { if (""=="c") { mat <- "Polychoric (Max. Lik.)" }
else { if (""=="d") { mat <- "Spearman" }
else { if (""=="e") { mat <- "Heterogenous (Two Step)" }
else { if (""=="f") { mat <- "Heterogenous (Max. Lik.)" }
 } } } } }
junto <- data.frame(mat,ysa$nf)
names(junto) <- c("Correlations","Number of factors to retain")
spsspivottable.Display(junto,
title="Comparison Data", hiderowdimlabel=TRUE, format=6)
spsspkg.EndProcedure()
 }
END PROGRAM.
BEGIN PROGRAM R.
# Items, Scales and Realibility
ind <- 0
spsspkg.StartProcedure ("Items, Scales and Realibility")
if ("n"=="y" || "n"=="y") {
if ("no"=="yes" && "a"=="a") { w <- factor2cluster(load, cut = 0);nu <- ncol(w); ind <- 1; w
<- as.data.frame(w); names(w) <- paste ("S",1:nu, rep="") }
if ("a"=="m") { ke <- list(c(1,3,-4),c(6,7))
w <- make.keys(n, ke, key.labels = NULL, item.labels = nam);nu <- ncol(w); ind <- 1; w <-
as.data.frame(w); names(w) <- paste ("S",1:nu, rep="") }
if ("a"=="all") { S1 <- rep(1,n); w <- data.frame(S1, row.names=nam); ind <- 1 }
if (ind==1) {
```

```
nu <- ncol(w)
spsspivottable.Display(w,
title="Assigned items to clusters (scores are adjusted for reverse scored items)", format=6)
if ("n"=="y") {
see <- scoreItems(w, mdata, totals = , ilabels = nam, missing = , impute="", delete=FALSE,
digits=6) }
else { see <- scoreItems(w, mdata, totals = TRUE, ilabels = nam, missing = TRUE,
impute="median", delete=FALSE, digits=6) }
junto <- data.frame(see$n.items)
names(junto) <- c("N Items")
spsspivottable.Display(junto,
title="Number of items for each scale",
format=6)
junto <- data.frame(see$av.r)
names(junto) <- paste ("S",1:nu, rep="")
spsspivottable.Display(junto, hiderowdimlabel=TRUE,
title="Average correlation")
spsspivottable.Display(see$cor, title="Intercorrelation of scales")
see$corrected[lower.tri(see$corrected)] <- t(see$corrected)[lower.tri(t(see$corrected))]
spsspivottable.Display(see$corrected,
title="Unattenuated correlations of scales (alpha on the diagonal)")
spsspivottable.Display(see$item.cor,
title="Correlation of items with scales (not corrected)")
spsspivottable.Display(see$item.corrected,
title=" Correlation of items with scales (corrected for item overlap)")
# Beggining Cronbach alpha
if (""=="y") {
cc <- cov(mdata, use="complete.obs", method="pearson")
alpha.u <- rep(0,nu)
alpha.s <- rep(0,nu)
if (m1==0) { co <- cor(mdata, use="complete.obs", method="pearson"); cor1 <- co; m1 <- 1 }
else { co <- cor1 }
for (i in 1:nu) {
ke <- diag(w[,i]) %*% co %*% diag(w[,i])
ke2 <- diag(w[,i]) %*% cc %*% diag(w[,i])
s <- sum(abs(w[,i]))
alpha.s[i] <- (1 - tr(ke)/sum(ke)) * (s/(s - 1))
alpha.u[i] <- (1 - tr(ke2)/sum(ke2)) * (s/(s - 1)) }
junto <- data.frame(t(alpha.u))
names(junto) <- paste ("S",1:nu, rep="")
spsspivottable.Display(junto, hiderowdimlabel=TRUE,
title="Raw Cronbach alpha")
junto <- data.frame(t(alpha.s))
names(junto) <- paste ("S",1:nu, rep="")
spsspivottable.Display(junto, hiderowdimlabel=TRUE,
title="Standardized Cronbach alpha")
w2 <- w
alp.u <- matrix (nrow=n,ncol=nu)
alp.s <- matrix (nrow=n,ncol=nu)
for (i in 1:n) {
for (j in 1:nu) {
```

```
if (w[i,j] != 0) { w2[i,j] = 0
ke <- diag(w2[,j]) %*% co %*% diag(w2[,j])
ke2 <- diag(w2[,j]) %*% cc %*% diag(w2[,j])
s <- sum(abs(w2[,j]))
alp.s[i,j] <- (1 - tr(ke)/sum(ke)) * (s/(s - 1))
alp.u[i,j] <- (1 - tr(ke2)/sum(ke2)) * (s/(s - 1))
w2 <- w } } }
junto <- data.frame(alp.u, row.names=nam)
names(junto) <- paste ("S",1:nu, rep="")
spsspivottable.Display(junto,
title="Raw Cronbach alpha if item deleted")
junto <- data.frame(alp.s, row.names=nam)
names(junto) <- paste ("S",1:nu, rep="")
spsspivottable.Display(junto,
title="Standardized Cronbach alpha if item deleted") } # end Cronbach alpha
# Beginning ordinal Cronbach alpha
if (""=="y") {
if (""=="a") {
if (m2==0) { co <- hetcor(da,  ML = FALSE, std.err = FALSE, pd=TRUE,
use="complete.obs", digits=6)
co <- co$correlations; cor2 <- co; m2 <- 1 }
else { co <- cor2 } }
if (""=="b") {
if (m3==0) { co <- hetcor(da,  ML = TRUE, std.err = FALSE, pd=TRUE,
use="complete.obs", digits=6)
co <- co$correlations; cor3<- co; m3 <- 1 }
else { co <- cor3 } }
or.al <- rep(0,nu)
for (i in 1:nu) {
ke <- diag(w[,i]) %*% co %*% diag(w[,i])
s <- sum(abs(w[,i]))
or.al[i] <- (1 - tr(ke)/sum(ke)) * (s/(s - 1)) }
junto <- data.frame(t(or.al))
names(junto) <- paste ("S",1:nu, rep="")
spsspivottable.Display(junto, hiderowdimlabel=TRUE,
title="Ordinal coefficient alpha")
w2 <- w
or.al <- matrix (nrow=n,ncol=nu)
for (i in 1:n) {
for (j in 1:nu) {
if (w[i,j] != 0) { w2[i,j] = 0
ke <- diag(w2[,j]) %*% co %*% diag(w2[,j])
s <- sum(abs(w2[,j]))
or.al[i,j] <- (1 - tr(ke)/sum(ke)) * (s/(s - 1))
w2 <- w } } }
junto <- data.frame(or.al, row.names=nam)
names(junto) <- paste ("S",1:nu, rep="")
spsspivottable.Display(junto,
title="Ordinal coefficient alpha if item deleted") } # end ordinal Cronbach alpha
# Beginning Armor's reliability theta
if (""=="y")  {
```

```
armo <- rep(0,nu)
co3 <- list()
if (m1==0) { co <- cor(mdata, use="complete.obs", method="pearson") }
else { co <- cor1 }
for (j in 1:nu) {
con <- 0
co2 <- co
for (i in 1:n) {
if (w[i,j] == 0) { co2 <- co2[-(i-con),-(i-con)]; con <- con+1 }
else { if (w[i,j] == -1) { co2[i-con,] <- (-1)*co2[i-con,]; co2[,i-con] <- (-1)*co2[,i-con] } } }
co3[[j]] <- co2
ei <- eigen(co2, symmetric=TRUE, only.values=TRUE)
ei <- ei$values
s <- dim(as.matrix(co2))[1]
armo[j] <- ((ei[1]-1)/ei[1]) * (s/(s - 1)) }
junto <- data.frame(t(armo))
names(junto) <- paste ("S",1:nu, rep="")
spsspivottable.Display(junto, hiderowdimlabel=TRUE,
title="Armor's reliability theta")
armo <- matrix (nrow=n,ncol=nu)
indi <- list()
for (j in 1:nu) {
con <- 1
for (i in 1:n) { if (w[i,j] != 0) {indi[[con]] <- i; con <- con+1} }
s <- dim(as.matrix(co3[[j]]))[1]
for (i in 1:s) {
co2 <- as.matrix(co3[[j]])
if (dim(co2)[1] > 1) {
co2 <- co2[-i,-i]
ei <- eigen(co2, symmetric=TRUE, only.values=TRUE)
ei <- ei$values
armo[indi[[i]],j] <- ((ei[1]-1)/ei[1]) * ((s-1)/(s - 2)) } } }
junto <- data.frame(armo, row.names=nam)
names(junto) <- paste ("S",1:nu, rep="")
spsspivottable.Display(junto,
title="Armor's reliability theta if item deleted") } # end Armor's reliability theta
# Beginning ordinal Armor's reliability theta
if (""=="y")  {
armo <- rep(0,nu)
co3 <- list()
if (""=="a") {
if (m2==0) {
co <- hetcor(da,  ML = FALSE, std.err = FALSE, pd=TRUE, use="complete.obs", digits=6)
co <- co$correlations; cor2 <- co; m2 <- 1 }
else { co <- cor2 } }
if (""=="b") {
if (m3==0) {
co <- hetcor(da,  ML = TRUE, std.err = FALSE, pd=TRUE, use="complete.obs", digits=6)
co <- co$correlations; cor3 <- co; m3 <-1 }
else { co <- cor3 } }
for (j in 1:nu) {
```

```
con <- 0
co2 <- co
for (i in 1:n) {
if (w[i,j] == 0) { co2 <- co2[-(i-con),-(i-con)]; con <- con+1 }
else { if (w[i,j] == -1) { co2[i-con,] <- (-1)*co2[i-con,]; co2[,i-con] <- (-1)*co2[,i-con] } } }
co3[[j]] <- co2
ei <- eigen(co2, symmetric=TRUE, only.values=TRUE)
ei <- ei$values
s <- dim(as.matrix(co2))[1]
armo[j] <- ((ei[1]-1)/ei[1]) * (s/(s - 1)) }
junto <- data.frame(t(armo))
names(junto) <- paste ("S",1:nu, rep="")
spsspivottable.Display(junto, hiderowdimlabel=TRUE,
title="Ordinal coefficient theta")
armo <- matrix (nrow=n,ncol=nu)
indi <- list()
for (j in 1:nu) {
con <- 1
for (i in 1:n) { if (w[i,j] != 0) {indi[[con]] <- i; con <- con+1} }
s <- dim(as.matrix(co3[[j]]))[1]
for (i in 1:s) {
co2 <- as.matrix(co3[[j]])
if (dim(co2)[1]>1) {
co2 <- co2[-i,-i]
ei <- eigen(co2, symmetric=TRUE, only.values=TRUE)
ei <- ei$values
armo[indi[[i]],j] <- ((ei[1]-1)/ei[1]) * ((s-1)/(s - 2)) } } }
junto <- data.frame(armo, row.names=nam)
names(junto) <- paste ("S",1:nu, rep="")
spsspivottable.Display(junto,
title="Ordinal coefficient theta if item deleted") } # end ordinal Armor's reliability theta
 } #end (ind=1)
 }
spsspkg.EndProcedure()
# Saving scores to SPSS
if ("n"=="y") {
if (""=="y" && ind==1) {
rm(mdata)
dict1 <- spssdictionary.GetDictionaryFromSPSS()
mdata <- spssdata.GetDataFromSPSS()
g <- list()
com <- ncol(w)
for (i in 1:com) { h <- as.character(i)
h <- paste("factor", h, sep="")
g[[i]] <- c(h,"",0,"F8.4","scale") }
dict <- spssdictionary.CreateSPSSDictionary(g)
dict <- data.frame(dict1,dict)
spssdictionary.SetDictionaryToSPSS("",dict)
fa=nrow(mdata)
fb=row.names(as.data.frame(see$scores))
ult=fb[nrow(see$scores)]
```

```
dife=fa-as.numeric(ult)
line=rep(NA,ncol(see$scores))
if (com==1) {
if (fb[1] != 1)  {
see$scores=rbind(as.matrix(line),as.matrix(see$scores[1:nrow(see$scores),]))
fb=row.names(as.data.frame(see$scores)) }
for (i in 2:(fa-dife-1)) {
if  (fb[i] != i) {
see$scores=rbind(as.matrix(see$scores[1:(i-
1),]),as.matrix(line),as.matrix(see$scores[i:nrow(see$scores),]))
fb=row.names(as.data.frame(see$scores)) } }
if (dife>0) {
for (i in 1:dife) {
see$scores=rbind(as.matrix(see$scores),as.matrix(line))  } } }
else{
if (fb[1] != 1)  {
see$scores=rbind(line,as.data.frame(see$scores[1:nrow(see$scores),]))
fb=row.names(as.data.frame(see$scores)) }
for (i in 2:(fa-dife-1)) {
if  (fb[i] != i) {
see$scores=rbind(as.data.frame(see$scores[1:(i-
1),]),line,as.data.frame(see$scores[i:nrow(see$scores),]))
fb=row.names(as.data.frame(see$scores)) } }
if (dife>0) {
for (i in 1:dife) {
see$scores=rbind(as.data.frame(see$scores),line)  } } }
see$scores=data.frame(see$scores,row.names=1:fa)
casedata <- data.frame(mdata,see$scores)
spssdata.SetDataToSPSS("",casedata)
spssdictionary.EndDataStep() } }
END PROGRAM.
set printback on.
```

Step 3: Factor analysis with 22 remaining items of the SSAW scale.

```
FACTOR
  /VARIABLES SWSRS_prä_aa_2 SWSRS_prä_aa_6 SWSRS_prä_sm_1 SWSRS_prä_sm_3
SWSRS_prä_sm_7 SWSRS_prä_sm_8 SWSRS_akt_sk_1 SWSRS_akt_sk_7
SWSRS_akt_sk_15 SWSRS_akt_sk_17 SWSRS_akt_sk_19 SWSRS_akt_sb_2
SWSRS_akt_sb_4 SWSRS_akt_sb_7 SWSRS_akt_sb_9 SWSRS_post_sb_1
SWSRS_post_sb_3 SWSRS_post_sb_4 SWSRS_post_sr_2 SWSRS_post_sr_4
SWSRS_post_sr_6 SWSRS_post_sr_7
  /MISSING LISTWISE
  /ANALYSIS SWSRS_prä_aa_2 SWSRS_prä_aa_6 SWSRS_prä_sm_1 SWSRS_prä_sm_3
SWSRS_prä_sm_7 SWSRS_prä_sm_8 SWSRS_akt_sk_1 SWSRS_akt_sk_7
SWSRS_akt_sk_15 SWSRS_akt_sk_17 SWSRS_akt_sk_19 SWSRS_akt_sb_2
SWSRS_akt_sb_4 SWSRS_akt_sb_7 SWSRS_akt_sb_9 SWSRS_post_sb_1
SWSRS_post_sb_3 SWSRS_post_sb_4 SWSRS_post_sr_2 SWSRS_post_sr_4
SWSRS_post_sr_6 SWSRS_post_sr_7
  /PRINT UNIVARIATE INITIAL CORRELATION KMO REPR AIC EXTRACTION
ROTATION
  /PLOT EIGEN
```

```
  /CRITERIA MINEIGEN(1) ITERATE(100)
  /EXTRACTION PAF
  /CRITERIA ITERATE(100)
  /ROTATION PROMAX(4)
  /METHOD=CORRELATION.

* Parallel Analysis program.

set mxloops=9000 printback=off width=80  seed = 1953125.
matrix.

* enter your specifications here.
compute ncases   = 121.
compute nvars    = 22.
compute ndatsets = 100.
compute percent  = 95.

* Specify the desired kind of parallel analysis, where:
  1 = principal components analysis
  2 = principal axis/common factor analysis.
compute kind = 2 .

****************** End of user specifications. ******************

* principal components analysis.
do if (kind = 1).
compute evals = make(nvars,ndatsets,-9999).
compute nm1 = 1 / (ncases-1).
loop #nds = 1 to ndatsets.
compute x = sqrt(2 * (ln(uniform(ncases,nvars)) * -1) ) &*
          cos(6.283185 * uniform(ncases,nvars) ).
compute vcv = nm1 * (sscp(x) - ((t(csum(x))*csum(x))/ncases)).
compute d = inv(mdiag(sqrt(diag(vcv)))).
compute evals(:,#nds) = eval(d * vcv * d).
end loop.
end if.

* principal axis / common factor analysis with SMCs on the diagonal.
do if (kind = 2).
compute evals = make(nvars,ndatsets,-9999).
compute nm1 = 1 / (ncases-1).
loop #nds = 1 to ndatsets.
compute x = sqrt(2 * (ln(uniform(ncases,nvars)) * -1) ) &*
          cos(6.283185 * uniform(ncases,nvars) ).
compute vcv = nm1 * (sscp(x) - ((t(csum(x))*csum(x))/ncases)).
compute d = inv(mdiag(sqrt(diag(vcv)))).
compute r = d * vcv * d.
compute smc = 1 - (1 &/ diag(inv(r)) ).
call setdiag(r,smc).
compute evals(:,#nds) = eval(r).
end loop.
```

```
end if.

* identifying the eigenvalues corresponding to the desired percentile.
compute num = rnd((percent*ndatsets)/100).
compute results = { t(1:nvars), t(1:nvars), t(1:nvars) }.
loop #root = 1 to nvars.
compute ranks = rnkorder(evals(#root,:)).
loop #col = 1 to ndatsets.
do if (ranks(1,#col) = num).
compute results(#root,3) = evals(#root,#col).
break.
end if.
end loop.
end loop.
compute results(:,2) = rsum(evals) / ndatsets.

print /title="PARALLEL ANALYSIS:".
do if   (kind = 1).
print /title="Principal Components".
else if (kind = 2).
print /title="Principal Axis / Common Factor Analysis".
end if.
compute specifs = {ncases; nvars; ndatsets; percent}.
print specifs /title="Specifications for this Run:"
 /rlabels="Ncases" "Nvars" "Ndatsets" "Percent".
print results /title="Random Data Eigenvalues"
 /clabels="Root" "Means" "Prcntyle"  /format "f12.6".

do if   (kind = 2).
print / space = 1.
print /title="Compare the random data eigenvalues to the".
print /title="real-data eigenvalues that are obtained from a".
print /title="Common Factor Analysis in which the # of factors".
print /title="extracted equals the # of variables/items, and the".
print /title="number of iterations is fixed at zero;".
print /title="To obtain these real-data values using SPSS, see the".
print /title="sample commands at the end of the parallel.sps program,".
print /title="or use the rawpar.sps program.".
print / space = 1.
print /title="Warning: Parallel analyses of adjusted correlation matrices".
print /title="eg, with SMCs on the diagonal, tend to indicate more factors".
print /title="than warranted (Buja, A., & Eyuboglu, N., 1992, Remarks on parallel".
print /title="analysis. Multivariate Behavioral Research, 27, 509-540.).".
print /title="The eigenvalues for trivial, negligible factors in the real".
print /title="data commonly surpass corresponding random data eigenvalues".
print /title="for the same roots. The eigenvalues from parallel analyses".
print /title="can be used to determine the real data eigenvalues that are".
print /title="beyond chance, but additional procedures should then be used".
print /title="to trim trivial factors.".
print / space = 1.
print /title="Principal components eigenvalues are often used to determine".
```

```
print /title="the number of common factors. This is the default in most".
print /title="statistical software packages, and it is the primary practice".
print /title="in the literature. It is also the method used by many factor".
print /title="analysis experts, including Cattell, who often examined".
print /title="principal components eigenvalues in his scree plots to determine".
print /title="the number of common factors. But others believe this common".
print /title="practice is wrong. Principal components eigenvalues are based".
print /title="on all of the variance in correlation matrices, including both".
print /title="the variance that is shared among variables and the variances".
print /title="that are unique to the variables. In contrast, principal".
print /title="axis eigenvalues are based solely on the shared variance".
print /title="among the variables. The two procedures are qualitatively".
print /title="different. Some therefore claim that the eigenvalues from one".
print /title="extraction method should not be used to determine".
print /title="the number of factors for the other extraction method.".
print /title="The issue remains neglected and unsettled.".

end if.

end matrix.

corr SWSRS_prä_aa_2
   SWSRS_prä_aa_6 SWSRS_prä_sm_1 SWSRS_prä_sm_3 SWSRS_prä_sm_7
SWSRS_prä_sm_8
   SWSRS_akt_sk_1 SWSRS_akt_sk_7
   SWSRS_akt_sk_15 SWSRS_akt_sk_17
   SWSRS_akt_sk_19
   SWSRS_akt_sb_2 SWSRS_akt_sb_4 SWSRS_akt_sb_7
   SWSRS_akt_sb_9 SWSRS_post_sb_1 SWSRS_post_sb_3
   SWSRS_post_sb_4 SWSRS_post_sr_2
   SWSRS_post_sr_4 SWSRS_post_sr_6 SWSRS_post_sr_7 / matrix out
('C:\Users\Cici\Desktop\SWSRS_EJPA Hauptachsen\Datensatz_SWSRS') / missing =
listwise.
matrix.
MGET /type= corr /file='C:\Users\Cici\Desktop\SWSRS_EJPA
Hauptachsen\Datensatz_SWSRS' .
compute smc = 1 - (1 &/ diag(inv(cr)) ).
call setdiag(cr,smc).
compute evals = eval(cr).
print { t(1:nrow(cr)) , evals }
 /title="Raw Data Eigenvalues"
 /clabels="Root" "Eigen."  /format "f12.6".
end matrix.


* Commands for obtaining the necessary real-data eigenvalues for
  principal axis / common factor analysis using SPSS;
  make sure to insert valid filenames/locations, and
  remove the '*' from the first columns.
* corr var1 to var20 / matrix out ('filename') / missing = listwise.
* matrix.
```

```
* MGET /type= corr /file='filename' .
* compute smc = 1 - (1 &/ diag(inv(cr)) ).
* call setdiag(cr,smc).
* compute evals = eval(cr).
* print { t(1:nrow(cr)) , evals }
 /title="Raw Data Eigenvalues"
 /clabels="Root" "Eigen."  /format "f12.6".
* end matrix.

* MAP-Test program.
*Mário Basto, José Manuel Pereira, IPCA
*Required: SPSS 21 and R Integration Plugin
*R Packages required: psych, polycor, GPArotation, nFactors, corpcor, ICS, R.utils.
set printback off.
BEGIN PROGRAM R.
# Correlations and descriptives
spsspkg.StartProcedure ("Correlation")
library (polycor)
library (psych)
library (GPArotation)
library (nFactors)
library(corpcor)
library(ICS)
library(R.utils)
# Reading data from SPSS
mdata <- spssdata.GetDataFromSPSS(variables=c("SWSRS_prä_aa_2
   SWSRS_prä_aa_6 SWSRS_prä_sm_1 SWSRS_prä_sm_3 SWSRS_prä_sm_7
SWSRS_prä_sm_8
   SWSRS_akt_sk_1 SWSRS_akt_sk_7
   SWSRS_akt_sk_15 SWSRS_akt_sk_17
   SWSRS_akt_sk_19
   SWSRS_akt_sb_2 SWSRS_akt_sb_4 SWSRS_akt_sb_7
   SWSRS_akt_sb_9 SWSRS_post_sb_1 SWSRS_post_sb_3
   SWSRS_post_sb_4 SWSRS_post_sr_2
   SWSRS_post_sr_4 SWSRS_post_sr_6 SWSRS_post_sr_7"))
scal <- spssdictionary.GetDictionaryFromSPSS(variables=c("SWSRS_prä_aa_2
   SWSRS_prä_aa_6 SWSRS_prä_sm_1 SWSRS_prä_sm_3 SWSRS_prä_sm_7
SWSRS_prä_sm_8
   SWSRS_akt_sk_1 SWSRS_akt_sk_7
   SWSRS_akt_sk_15 SWSRS_akt_sk_17
   SWSRS_akt_sk_19
   SWSRS_akt_sb_2 SWSRS_akt_sb_4 SWSRS_akt_sb_7
   SWSRS_akt_sb_9 SWSRS_post_sb_1 SWSRS_post_sb_3
   SWSRS_post_sb_4 SWSRS_post_sr_2
   SWSRS_post_sr_4 SWSRS_post_sr_6 SWSRS_post_sr_7"))
is.na(mdata) <- is.na(mdata)
m1 <- 0; m2 <- 0; m3 <- 0; m4 <- 0; m5<-0; m6<-0
nam <- names(mdata)
# Missing values (pairwise or listwise)
n <- ncol(mdata)
ncase <- nrow(mdata)
```

```
ncase2 <- ncase
 # counting listwise cases
cont <- 0
for (i in 1:ncase) {
ind <- 0
j <- 0
while (j<n && ind ==0) {
j <- j+1
if (is.na(mdata[i,j])) {
cont <- cont+1
ind <- 1 } } }
ncase_list <- ncase-cont
if ("complete.obs"=="complete.obs") {     # counting listwise cases
ncase <- ncase_list
junto <- data.frame(ncase, cont)
names(junto) <- c("Number of cases", "Number of cases excluded")
spsspivottable.Display(junto,
title="Listwise deletion", hiderowdimlabel=TRUE, format=6) }
if ("complete.obs"=="pairwise.complete.obs") {
spsspivottable.Display(count.pairwise(mdata),
title="Number of valid cases for each pairwise correlation", format=6) }
da <- mdata
dah <- mdata
for (i in 1:n) { da[,i]<-ordered(mdata[,i])}  # for calculation of polychoric correlations
for (i in 1:n) {
if  ( scal["varMeasurementLevel",i]=="nominal") {dah[,i]<-factor(mdata[,i]) }
else { if  (scal["varMeasurementLevel",i]=="ordinal") { dah[,i]<-ordered(mdata[,i]) } } }
#for calculation of correlations according to their scales
# Multivariate normality tests
if ("n"=="y") {
mn <- mvnorm.skew.test(mdata, na.action = na.omit)
mn2 <- mvnorm.kur.test(mdata, method = "satterthwaite", n.simu = 1000, na.action=na.omit)
mm <- c("Skewness","Kurtosis")
mn3 <- c(mn$statistic, mn2$statistic)
mn4 <- c(mn$p.value, mn2$p.value)
junto <- data.frame(mm, mn3, mn4)
names(junto) <- c("Moment","Test statistic", "p-value")
spsspivottable.Display(junto,
title="Test of Multivariate Normality based on Skewness and Kurtosis",
hiderowdimlabel=TRUE) }
if ("n"=="y") { co <- corr.test(mdata, use="complete.obs", method="pearson"); m1 <-1
cor1 <- co$r
test <- co$p
spsspivottable.Display(cor1,
title="Pearson correlations")
spsspivottable.Display(test,
title="p-values for Pearson correlations") }
if ("n"=="y") { co <- hetcor(da,  ML = FALSE, std.err = FALSE, pd=TRUE,
use="complete.obs", digits=6); m2 <-1
cor2 <- co$correlations
spsspivottable.Display(cor2,
```

```
title="Polychoric correlations - Two Step Estimation") }
if ("n"=="y") { co <- hetcor(da,  ML = TRUE, std.err = FALSE, pd=TRUE,
use="complete.obs", digits=6); m3 <-1
cor3 <- co$correlations
spsspivottable.Display(cor3,
title="Polychoric correlations - Max. Likelihood Estimation") }
if ("n"=="y") { co <- corr.test(mdata, use="complete.obs", method="spearman"); m4 <-1
cor4 <- co$r
test <- co$p
spsspivottable.Display(cor4,
title="Spearman correlations")
spsspivottable.Display(test,
title="p-values for Spearman correlations") }
if ("n"=="y") { co <- hetcor(dah,  ML =FALSE , std.err = FALSE, pd=TRUE,
use="complete.obs", digits=6)
m5 <-1
cor5 <- co$correlations
ctype <- as.data.frame(co$type)
names(ctype) <- nam
het<- rbind(signif(cor5, digits=3), ctype)
row.names(het) <- c(nam,paste("Correlation Type",nam))
spsspivottable.Display(het,
title="Heterogenous correlations (Two Step)") }
if ("n"=="y") { co <- hetcor(dah,  ML =TRUE , std.err = FALSE, pd=TRUE,
use="complete.obs", digits=6)
m6 <-1
cor6 <- co$correlations
ctype <- as.data.frame(co$type)
names(ctype) <- nam
het<- rbind(signif(cor6, digits=3), ctype)
row.names(het) <- c(nam,paste("Correlation Type",nam))
spsspivottable.Display(het,
title="Heterogenous correlations (Max. Likelihood)") }
spsspkg.EndProcedure()
END PROGRAM.
BEGIN PROGRAM R.
# Correlation differences
spsspkg.StartProcedure ("Correlation differences")
if ("n"=="y")  {
if (m1==0)  {cor1 <- cor(mdata, use="complete.obs", method="pearson"); m1 <-1 }
if (m2==0)  { co <- hetcor(da,  ML = FALSE, std.err = FALSE, pd=TRUE,
use="complete.obs", digits=6); m2 <-1
cor2 <- co$correlations }
d12 <- cor1-cor2
spsspivottable.Display(d12,
title="Pearson - Polychoric (Two Step)") }
if ("n"=="y")  {
if (m1==0)  {cor1 <- cor(mdata, use="complete.obs", method="pearson"); m1 <-1 }
if (m3==0)  { co <- hetcor(da,  ML = TRUE, std.err = FALSE, pd=TRUE,
use="complete.obs", digits=6); m3 <-1
cor3 <- co$correlations }
```

```
d13 <- cor1-cor3
spsspivottable.Display(d13,
title="Pearson - Polychoric (Max. Lik.)")  }
if ("n"=="y")  {
if (m2==0)  { co <- hetcor(da,  ML = FALSE, std.err = FALSE, pd=TRUE,
use="complete.obs", digits=6); m2 <-1
cor2 <- co$correlations }
if (m3==0)  { co <- hetcor(da,  ML = TRUE, std.err = FALSE, pd=TRUE,
use="complete.obs", digits=6); m3 <-1
cor3 <- co$correlations }
d23 <- cor2-cor3
spsspivottable.Display(d23,
title="Polychoric (Two Step) - Polychoric (Max. Lik.)")   }
if ("n"=="y")  {
if (m1==0)  { cor1 <- cor(mdata, use="complete.obs", method="pearson"); m1 <-1 }
if (m4==0)  { cor4 <- cor(mdata, use="complete.obs", method="spearman"); m4 <-1 }
d14 <- cor1-cor4
spsspivottable.Display(d14,
title="Pearson - Spearman") }
if ("n"=="y")  {
if (m2==0)  { co <- hetcor(da,  ML = FALSE, std.err = FALSE, pd=TRUE,
use="complete.obs", digits=6); m2 <-1
cor2 <- co$correlations }
if (m4==0)  { cor4 <- cor(mdata, use="complete.obs", method="spearman"); m4 <-1 }
d24 <- cor2-cor4
spsspivottable.Display(d24,
title="Polychoric (Two Step) - Spearman") }
if ("n"=="y")  {
if (m3==0)  { co <- hetcor(da,  ML = TRUE, std.err = FALSE, pd=TRUE,
use="complete.obs", digits=6); m3 <-1
cor3 <- co$correlations }
if (m4==0)  { cor4 <- cor(mdata, use="complete.obs", method="spearman"); m4 <-1 }
d34 <- cor3-cor4
spsspivottable.Display(d34,
title="Polychoric (Max. Lik.) - Spearman") }
spsspkg.EndProcedure()
END PROGRAM.
BEGIN PROGRAM R.
# Principal Components Analysis extracting PCs from the correlation matrix (function
princomp)
if ("no"=="yes")  {
if (""=="a") {spsspkg.StartProcedure ("Principal Component Analysis - Pearson") }
else { if (""=="b") {spsspkg.StartProcedure ("Principal Component Analysis - Polychoric
(Two Step)") }
else { if (""=="c") {spsspkg.StartProcedure ("Principal Component Analysis - Polychoric
(Max. Likelihood)") }
else { if (""=="d") {spsspkg.StartProcedure ("Principal Component Analysis - Spearman") }
else { if (""=="e") {spsspkg.StartProcedure ("Principal Component Analysis - Covariance") }
else { if (""=="f") {spsspkg.StartProcedure ("Principal Component Analysis - Heterogenous
(Two Step)") }
```

```
else { if (""=="g") {spsspkg.StartProcedure ("Principal Component Analysis - Heterogenous
(Max. Likelihood)") }
} } } } } }
if (""=="a")  {
if (m1==0) {cor1 <- cor(mdata, use="complete.obs", method="pearson"); m1 <-1 }
ad <-princomp(cor = FALSE, covmat = cor1)  }
else { if (""=="b")  {
if (m2==0) { co <- hetcor(da,  ML = FALSE, std.err = FALSE, pd=TRUE,
use="complete.obs", digits=6); m2 <-1
cor2 <- co$correlations }
ad <-princomp(cor = FALSE, covmat = cor2)  }
else { if (""=="c")  {
if (m3==0) { co <- hetcor(da,  ML = TRUE, std.err = FALSE, pd=TRUE,
use="complete.obs", digits=6); m3 <-1
cor3 <- co$correlations }
ad <-princomp(cor = FALSE, covmat = cor3)  }
else { if (""=="d") {
if (m4==0) { cor4 <- cor(mdata, use="complete.obs", method="spearman"); m4 <-1 }
ad <-princomp(cor = FALSE, covmat = cor4)  }
else { if (""=="e") { cor <- cov(mdata, use="complete.obs")
ad <-princomp(cor = FALSE, covmat = cor) }
else { if (""=="f")  {
if (m5==0) { co <- hetcor(dah,  ML = FALSE, std.err = FALSE, pd=TRUE,
use="complete.obs", digits=6); m5 <-1
cor5 <- co$correlations }
ad <-princomp(cor = FALSE, covmat = cor5)  }
else { if (""=="g")  {
if (m6==0) { co <- hetcor(dah,  ML = TRUE, std.err = FALSE, pd=TRUE,
use="complete.obs", digits=6); m6 <-1
cor6 <- co$correlations }
ad <-princomp(cor = FALSE, covmat = cor6)  }
 } } } } } }
spsspivottable.Display(ad$loadings[,], title="Eigenvectors")
pro <- ad$sdev^2
loa <- ad$loadings
loa <- loa%*%diag(ad$sdev)
# Sorting or not loadings
if (""=="no") {
spsspivottable.Display(loa[,],
title="Component Loadings",
collabels=paste ("Comp.",1:n, sep="") ) }
else {
# To correct a little problem when dealing with matrices for some versions of ICLUST.sort
loab <- list()
loab$loadings <- loa
loab$pattern <- loa
p <- ICLUST.sort(loab)
p <- p$sorted[,4:(n+3)]
spsspivottable.Display(p,
title="Sorted Component Loadings", collabels=paste ("Comp.",1:n, sep="")) }
sume <- 0
```

```
sum2 <- rep(0,n)
for (i in 1:n) { sume <- sume + pro[i]
sum2[i] = sum2[i] + sume}
junto <- data.frame(ad$sdev,pro,pro/sume*100,sum2/sume*100)
names(junto) <- c("Stdev","Eigenvalues","% of Variance","Cumulative %")
spsspivottable.Display(junto,
title="Variance Explained")
plot(ad,main="Scree Plot",type="lines")
spsspkg.EndProcedure() }
END PROGRAM.
BEGIN PROGRAM R.
#Factor Analysis (functions "principal", "fa", "iterativePrincipalAxis", "PrincipalAxis")
if ("no"=="yes")  {
suprime <-
if ("a"=="a") {
if (m1==0) { cor1 <- cor(mdata, use="complete.obs", method="pearson"); m1 <-1 }
co <- cor1 }
else { if ("a"=="b") {
if (m2==0) { co <- hetcor(da,  ML = FALSE, std.err = FALSE, pd=TRUE,
use="complete.obs", digits=6); m2 <-1
cor2 <- co$correlations }
co <- cor2 }
else { if ("a"=="c") {
if (m3==0) { co <- hetcor(da,  ML = TRUE, std.err = FALSE, pd=TRUE,
use="complete.obs", digits=6); m3 <-1
cor3 <- co$correlations }
co <- cor3 }
else { if ("a"=="d") {
if (m4==0) { cor4 <- cor(mdata, use="complete.obs", method="spearman"); m4 <-1 }
co <- cor4 }
else { if ("a"=="e") {
if (m5==0) { co <- hetcor(dah,  ML = FALSE, std.err = FALSE, pd=TRUE,
use="complete.obs", digits=6); m5 <-1
cor5 <- co$correlations }
co <- cor5 }
else { if ("a"=="f") {
if (m6==0) { co <- hetcor(dah,  ML = TRUE, std.err = FALSE, pd=TRUE,
use="complete.obs", digits=6); m6 <-1
cor6 <- co$correlations }
co <- cor6 }
 } } } } }
spsspkg.StartProcedure ("Factor Analysis")
pr <-
if (""=="pca" && pr > n) { pr <- n }
if (""!="pca" && pr > (n-1)) { pr <- n-1 }
if (""=="pca") {
ad<-principal(co, nfactors = pr, residuals = FALSE, rotate="none", n.obs=ncase,
scores=FALSE) }
else { if (""=="ipa") {
ad<-principal(co, nfactors = pr, rotate="none", n.obs=ncase, scores=FALSE)
```

```
ad3 <- iterativePrincipalAxis(co, nFactors=pr, communalities="", iterations=200,
tolerance=1e-6)
ad$loadings <- ad3$loadings }
else { if (""=="nipa") {
ad<-principal(co, nfactors = pr, rotate="none", n.obs=ncase, scores=FALSE)
ad3 <- principalAxis(co, nFactors=pr, communalities="")
ad$loadings <- ad3$loadings }
else {ad <- fa(co, nfactors=pr, residuals = FALSE, rotate = "none", n.obs=ncase_list,
SMC=TRUE, min.err = 1e-6, digits =8, max.iter=200, symmetric=TRUE, warnings=TRUE,
fm="")
ad$loadings <- ad$loadings[,order(colnames(ad$loadings))]
 } } }
if ("a"=="a") { mat <- "Pearson" }
else { if ("a"=="b") { mat <- "Polychoric (Two Step)" }
else { if ("a"=="c") { mat <- "Polychoric (Max. Lik.)" }
else { if ("a"=="d") { mat <- "Spearman" }
else { if ("a"=="e") { mat <- "Heterogenous (Two Step)" }
else { if ("a"=="f") { mat <- "Heterogenous (Max. Lik.)" }
 } } } } }
if ("complete.obs"=="pairwise.complete.obs") { caso <- "pairwise"}
if ("complete.obs"=="complete.obs") { caso <- "listwise"}
junto <- data.frame(mat, "", caso)
names(junto) <- c("Correlation Matrix", "Factor Extraction", "Missing values")
spsspivottable.Display(junto,
title="Factor Analysis", hiderowdimlabel=TRUE)
ad$loadings <- as.matrix(ad$loadings)
row.names(ad$loadings) <- nam
load <- ad$loadings
ConFac <- load
if (""=="ipa" || ""=="nipa") {
sum2 <- rep(0,n)
for(i in 1:pr) { sum2 <- load[,i]^2 + sum2 }
ad$communality <- sum2
}
extracted <- ad$communality
b <- data.frame(extracted, row.names=nam)
spsspivottable.Display(b, title="Communalities")
# Sorting or not loadings
if (""=="no") {
supre <- load
for (j in 1:pr) {
for (i in 1:nrow(load)) {
if (abs(load[i,j]) < suprime) { supre[i,j] <- NA } } }
spsspivottable.Display(supre[,],
title="Factor Loadings (unrotated)", collabels=paste ("F",1:pr, sep="")) }
else {
ad <- fa.sort(ad)
load2 <- ad$loadings
supre <- load2
for (j in 1:pr) {
for (i in 1:nrow(load2)) {
```

```
if (abs(load2[i,j]) < suprime) { supre[i,j] <- NA } } }
spsspivottable.Display(supre[,],
title="Sorted Factor Loadings (unrotated)", collabels=paste ("F",1:pr, sep="") ) }
# Initial Eigenvalues
if (""=="pca" || ""=="ipa" || ""=="nipa") {
Eigenvalues <- ad$values }
else { Eigenvalues <- ad$e.values }
# Scree plot
if ("n"=="y") {
plotuScree(Eigenvalues, ylab = "Eigenvalues",
xlab = "Components",
main = "Scree Plot") }
sume  <- 0
sum2 <- rep(0,n)
for (i in 1:n) { sume <- sume  + Eigenvalues[i]
sum2[i] = sum2[i] + sume }
junto <- data.frame(Eigenvalues,Eigenvalues/sume*100,sum2/sume*100)
names(junto) <- c("Eigenvalues","% of Variance","Cumulative %")
spsspivottable.Display(junto,
title=" Variance Explained (initial)")
# Calculation of Sums of Squared Loadings
pro <- rep(0,pr)
for (j in 1:pr) { sum3 <- 0
for (i in 1:n) { sum3<- sum3 + load[i,j]^2 }
pro[j] <- sum3 }
sum3 <- 0
sum2 <- rep(0,pr)
for (i in 1:pr) { sum3<- sum3 + pro[i]
sum2[i] = sum2[i] + sum3 }
junto <- data.frame(pro,pro/sume*100,sum2/sume*100)
names(junto) <- c("Sums of Squared Loadings","% of Variance","Cumulative %")
spsspivottable.Display(junto,
title="Variance Explained (extracted)")
# Factor plot unrotated
if ("n"=="y") {
l1 <-
l2 <-
ll1 <- min(l1,l2); ll2 <- max(l1,l2)
if (ll1 < ll2 && ll2 < (pr+1)) {
x <-matrix(c(load[,ll1],load[,ll2]),ncol=2)
x1 <- paste("Factor", ll1, sep=" ")
y1 <- paste("Factor", ll2, sep=" ")
factor.plot(x, cut = , labels=nam, xlim = c(-1, 1), ylim = c(-1, 1), xlab=x1, ylab=y1, title =
"Factor Plot (initial solution)")  } }
# Factor diagram unrotated
if ("n"=="y") {
fa.diagram(ad, sort=, labels=NULL, cut=, simple=, errors=TRUE, digits=2, e.size=0.05,
rsize=0.15, side=2, main="Factor Diagram (initial solution)", cex=NULL) }
# Rotations
if (""!="none" || ""=="yes") {
```

```
if (""=="varimax") {ad2 <- Varimax(load, Tmat=diag(ncol(load)), normalize=, eps=1e-5,
maxit=500) }
else { if (""=="quartimax") {ad2 <- quartimax(load, Tmat=diag(ncol(load)), normalize=,
eps=1e-5, maxit=500) }
else { if (""=="quartimin") {ad2 <- quartimin(load, Tmat=diag(ncol(load)), normalize=,
eps=1e-5, maxit=500) }
else { if (""=="equamax") {ad2 <- cfT(load, Tmat=diag(ncol(load)),
kappa=pr/(2*nrow(load)), normalize=, eps=1e-5, maxit=500) }
else { if (""=="parsimax") {ad2 <- cfT(load, Tmat=diag(ncol(load)), kappa=(pr-
1)/(nrow(load)+pr-2), normalize=, eps=1e-5, maxit=500) }
else { if (""=="factor.parsimony") {ad2 <- cfT(load, Tmat=diag(ncol(load)), kappa=1,
normalize=, eps=1e-5, maxit=500) }
else { if (""=="biquartimin") { ad2 <- oblimin(load, Tmat=diag(ncol(load)), gam=0.5,
normalize=, eps=1e-5, maxit=500) }
else { if (""=="covarimin") { ad2 <- oblimin(load, Tmat=diag(ncol(load)), gam=1,
normalize=, eps=1e-5, maxit=500) }
else { if (""=="oblimax") {ad2 <- oblimax(load, Tmat=diag(ncol(load)), normalize=, eps=1e-
5, maxit=500) }
else { if (""=="entropy") {
kk <-
if (""=="") { kk <- ncol(load)*nrow(load) - nrow(load) }
ad2 <- entropy(load, Tmat=diag(ncol(load)), normalize=, eps=1e-5, maxit=500) }
else { if (""=="simplimax") {ad2 <- simplimax(load, Tmat=diag(ncol(load)), k=, normalize=,
eps=1e-5, maxit=500) }
else { if (""=="bentlerT") {ad2 <- bentlerT(load, Tmat=diag(ncol(load)), normalize=, eps=1e-
5, maxit=500) }
else { if (""=="bentlerQ") {ad2 <- bentlerQ(load, Tmat=diag(ncol(load)), normalize=,
eps=1e-5, maxit=500) }
else { if (""=="tandemI") {ad2 <- tandemI(load, Tmat=diag(ncol(load)), normalize=, eps=1e-
5, maxit=500) }
else { if (""=="tandemII") {ad2 <- tandemII(load, Tmat=diag(ncol(load)), normalize=,
eps=1e-5, maxit=500) }
else { if (""=="geominT") {ad2 <- geominT(load, Tmat=diag(ncol(load)), delta=0.01,
normalize=, eps=1e-5, maxit=500) }
else { if (""=="geominQ") {ad2 <- geominQ(load, Tmat=diag(ncol(load)), delta=0.01,
normalize=, eps=1e-5, maxit=500) }
else { if (""=="infomaxT") {ad2 <- infomaxT(load, Tmat=diag(ncol(load)), normalize=,
eps=1e-5, maxit=500) }
else { if (""=="infomaxQ") {ad2 <- infomaxQ(load, Tmat=diag(ncol(load)), normalize=,
eps=1e-5, maxit=500) }
else { if (""=="mccammon") {ad2 <- mccammon(load, Tmat=diag(ncol(load)), normalize=,
eps=1e-5, maxit=500) }
} } } } } } } } } } } } } } } } } } }
prom<-0
# Promax rotation (adapted from Promax function from psych package)
if (""!="none") {  promai <- ad2$orthogonal } else { promai <- FALSE }
if ((promai) || (""=="none")) {
if (""=="yes") {
m <-
if (ncol(load) < 2) { return(load)}
dn <- dimnames(load)
```

```
if (""!="none") { load <- ad2$loadings }
Q <- load * abs(load)^(m - 1)
U <- lm.fit(load, Q)$coefficients
d <- diag(solve(t(U) %*% U))
U <- U %*% diag(sqrt(d))
dimnames(U) <- NULL
z <- load %*% U
if (""!="none") {U <- ad2$Th %*% U}
ui <- solve(U)
Phi <- ui %*% t(ui)
dimnames(z) <- dn
prom <-1
ad2<- list(loadings = z, Th = U%*%Phi, Phi = Phi, orthogonal = FALSE, method ="promax",
initial="") } }
load <- ad2$loadings   # if oblique rotation, this is the pattern matrix
 # Sorting or not loadings after rotation
if (""=="no") {
if (ad2$orthogonal == FALSE) {
supre <- load
for (j in 1:pr) {
for (i in 1:nrow(load)) {
if (abs(load[i,j]) < suprime) { supre[i,j] <- NA } } }
spsspivottable.Display(supre[,], title="Pattern Matrix",
collabels=paste ("F",1:pr, sep="") )
p <- as.matrix(load[,])%*%ad2$Phi
supre <- p
for (j in 1:pr) {
for (i in 1:nrow(p)) {
if (abs(p[i,j]) < suprime) { supre[i,j] <- NA } } }
spsspivottable.Display(supre[,],
title="Structure Matrix") }
else {
supre <- load
for (j in 1:pr) {
for (i in 1:nrow(load)) {
if (abs(load[i,j]) < suprime) { supre[i,j] <- NA } } }
spsspivottable.Display(supre[,],
title="Rotated Factor Loadings",
collabels=paste ("F",1:pr, sep="") ) } }
else {
ad$loadings <- ad2$loadings
ad <- fa.sort(ad)
load2 <- ad$loadings
supre <- load2
for (j in 1:pr) {
for (i in 1:nrow(load2)) {
if (abs(load2[i,j]) < suprime) { supre[i,j] <- NA } } }
if ( ! ad2$orthogonal) { spsspivottable.Display(supre[,],
title="Sorted Pattern Matrix",
collabels=paste ("F",1:pr, sep="") )
p <- as.matrix(load2[,])%*%ad2$Phi
```

```
supre <- p
for (j in 1:pr) {
for (i in 1:nrow(p)) {
if (abs(p[i,j]) < suprime) { supre[i,j] <- NA } } }
spsspivottable.Display(supre[,],
title="Sorted Structure Matrix",
collabels=paste ("F",1:pr, sep="") ) }
else {
supre <- load2
for (j in 1:pr) {
for (i in 1:nrow(load2)) {
if (abs(load2[i,j]) < suprime) { supre[i,j] <- NA } } }
spsspivottable.Display(supre[,],
title="Sorted Rotated Factor Loadings",
collabels=paste ("F",1:pr, sep="") ) } }
# Calculation of Sums of Squared Loadings after rotation
if (ad2$orthogonal) {
pro <- rep(0,pr)
for (j in 1:pr) { sum3 <- 0
for (i in 1:n) { sum3<- sum3 + load[i,j]^2 }
pro[j] <- sum3 }
sum3 <- 0
sum2 <- rep(0,pr)
for (i in 1:pr) { sum3<- sum3 + pro[i]
sum2[i] = sum2[i] + sum3 }
junto <- data.frame(pro,pro/sume*100,sum2/sume*100)
names(junto) <- c("Sums of Squared Loadings","% of Variance","Cumulative %")
spsspivottable.Display(junto,
title="Rotated Variance Explained (extracted)") }
else {
pro <- rep(0,pr)
for (j in 1:pr) { sum3 <- 0
for (i in 1:n) { sum3<- sum3 + p[i,j]^2 }
pro[j] <- sum3 }
junto <- data.frame(pro)
names(junto) <- c("Sums of Squared Loadings")
spsspivottable.Display(junto,
title="Rotation Sums of Squared Loadings from Structure Matrix") }
spsspivottable.Display(ad2$Th,
title="Rotation Matrix",
collabels=paste ("F",1:pr, sep=""),
rowlabels=paste ("F",1:pr, sep="") )
spsspivottable.Display(ad2$Phi,
title="Correlation matrix of rotated factors",
collabels=paste ("F",1:pr, sep=""),
rowlabels=paste ("F",1:pr, sep="") )
if (prom==0) {
junto <- data.frame(ad2$method,ad2$orthogonal,ad2$convergence)
names(junto) <- c("method","orthogonal","convergence")
spsspivottable.Display(junto,
title="Rotation", hiderowdimlabel=TRUE) }
```

```
else { junto <- data.frame(ad2$method,ad2$orthogonal, ad2$initial)
names(junto) <- c("method","orthogonal","initial rotation")
spsspivottable.Display(junto,
title="Rotation", hiderowdimlabel=TRUE) }
# Factor plot after rotation
if ("n"=="y") {
l1 <-
l2 <-
ll1 <- min(l1,l2); ll2 <- max(l1,l2)
if (ll1 < ll2 && ll2 < (pr+1)) {
x <-matrix(c(load[,ll1],load[,ll2]), ncol=2)
x1 <- paste("Factor", ll1, sep=" ")
y1 <- paste("Factor", ll2, sep=" ")
factor.plot(x, cut = , labels=nam,  xlim = c(-1, 1), ylim = c(-1, 1), xlab=x1, ylab=y1, title =
"Factor Plot (rotated solution)")  } }
# Factor diagram after rotation
if ("n"=="y") {
ad$loadings <- ad2$loadings
colnames(ad$loadings) <- paste("F", 1:pr, sep = "")
fa.diagram(ad, sort=, labels=NULL, cut=, simple=, errors=TRUE, digits=2, e.size=0.05,
rsize=0.15, side=2, main="Factor Diagram (rotated solution)", cex=NULL) }
 }   #end of rotation
# KMO
if ("n"=="y") {
g <- diag(0,n)
par <- cor2pcor(co)
cco <- co
cco[upper.tri(cco, TRUE)] <- g[upper.tri(cco, TRUE)]
par[upper.tri(par, TRUE)] <- g[upper.tri(par, TRUE)]
cco <- sum(cco^2)
kmo <- cco/(sum(par^2)+cco)
KMO <- c(kmo)
b <- data.frame(KMO)
spsspivottable.Display(b,
title="KMO - Kaiser-Meyer-Olkin measure of sampling adequacy",
hiderowdimlabel=TRUE) }
# MSA
if ("n"=="y") {
par <- cor2pcor(co)
ms <- rep(0,n)
for (i in 1:n) { ms[i] <- ( sum(co[i,]^2)-1 )  / (sum(co[i,]^2)+sum(par[i,]^2)-2) }
MSA <- c(ms)
b <- data.frame(MSA, row.names=nam)
spsspivottable.Display(b,
title="MSA - Measures of sampling adequacy") }
# Residual matrix for extraction using NFactors package
if (""=="ipa" || ""=="nipa") {
res <- diag(0,n)
for (i in 2:n) {
for (j in 1:(i-1)) {
res [i,j] <- co[i,j] - sum(ConFac[i,]*ConFac[j,]) } }
```

```
junto <- res + t(res) + diag(1 - ad$communality)
row.names(junto) <- nam
names(junto) <- nam
ad$residual <- junto
junto <- data.frame(junto) }
# GFI and AGFI
if ("n"=="y") {
s <- ad$residual
som <- sum(diag(s%*%s))
som2 <- sum(diag(co%*%co))
som <- 1-som/som2
b <- data.frame(som)
names(b) <- c("GFI (ULS)")
spsspivottable.Display(b,
title="GFI (Goodness of Fit)", hiderowdimlabel=TRUE) }
if ("n"=="y") {
aus <- matrix(0,n,n)
for (i in 1:(n-1)) {
for (j in (i+1):n) {
aus[i,j] <- sum(ConFac[i,]*ConFac[j,]) } }
aus2 <- t(aus)
diag(aus2) <- 1
aus <- aus + aus2
som <- sum(diag((solve(aus)%*%co-diag(n))%*%(solve(aus)%*%co-diag(n))))
som2 <- sum(diag(solve(aus)%*%co%*%solve(aus)%*%co))
som <- 1-som/som2
b <- data.frame(som)
names(b) <- c("GFI (ML)")
spsspivottable.Display(b,
title="GFI (Goodness of Fit)", hiderowdimlabel=TRUE) }
# RMSR
if ("n"=="y") {
g <- diag(0,n)
s <- ad$residual
s[upper.tri(s, TRUE)] <- g[upper.tri(s, TRUE)]
rmsr <- sqrt(2*sum(s^2)/(n*(n-1)))
RMSR <- c(rmsr)
b <- data.frame(RMSR)
spsspivottable.Display(b,
title="RMSR (Root Mean Square Residual)",
hiderowdimlabel=TRUE) }
# Root Mean Square Partials Correlations Controlling Factors
if ("n"=="y") {
cc <- co-(ConFac%*%t(ConFac))
ca <- sqrt(diag(cc))
d <- diag(1/ca)
ea <- d%*%cc%*%d   #partial correlations controlling factors
rmsp <- sqrt((sum(ea^2)-n)/(n*(n-1)))
RMSP <- c(rmsp)
b <- data.frame(RMSP)
spsspivottable.Display(b,
```

```
title="Root mean square partial correlations controlling factors",
hiderowdimlabel=TRUE) }
# Partial Correlations controlling all other variables
if ("n"=="y") {
par <- cor2pcor(co)
row.names(par) <- nam
par <- data.frame(par)
names(par) <- nam
spsspivottable.Display(par,
title="Partial correlations controlling all other variables") }
# Partial Correlations controlling Factors
if ("n"=="y") {
cc <- co-(ConFac%*%t(ConFac))
ca <- sqrt(diag(cc))
d <- diag(1/ca)
ea <- d%*%cc%*%d   #partial correlations controlling factors
row.names(ea) <- nam
ea <- data.frame(ea)
names(ea) <- nam
spsspivottable.Display(ea,
title="Partial correlations controlling factors") }
# Residual correlations
if (FALSE==TRUE) {
spsspivottable.Display(ad$residual,
title="Residual correlations with uniqueness on the diagonal (computed between observed and
reproduced correlations)")
cont <- 0
s <- ad$residual
for (i in 2:n) { for (j in 1:(i-1)) { if (abs(s[i,j])> 0.05) { cont <- cont+1 } } }
p <- 2*cont/(n*(n-1))*100
table <- spss.BasePivotTable("Fit of the model to the correlation matrix","OMS")
rowdim <- BasePivotTable.Append(table,Dimension.Place.row,"")
coldim <- BasePivotTable.Append(table,Dimension.Place.column,"Residual fit values")
row1 <- spss.CellText.String("Values")
col1 <- spss.CellText.String("Residuals > 0.05")
col2 <- spss.CellText.String("% residuals > 0.05")
BasePivotTable.SetCategories(table,rowdim,list(row1))
BasePivotTable.SetCategories(table,coldim,list(col1,col2))
BasePivotTable.SetCellsByColumn(table,col1,list(spss.CellText.Number(cont,6)))
BasePivotTable.SetCellsByColumn(table,col2,list(spss.CellText.Number(p))) }
# Determinant
if ("n"=="y") {
b <- determinant(co, logarithm = FALSE)
junto <- data.frame(b$modulus)
names(junto) <- c("Determinant")
spsspivottable.Display(junto,
title="Determinant (modulus)",
hiderowdimlabel=TRUE) }
# Tests to correlation matrix
if ("n"=="y") {
bar <- cortest.bartlett(co, n = ncase)
```

```
bar2 <- cortest.normal(co,n1=ncase, fisher = FALSE)
bar3 <- cortest.jennrich(co, diag(rep(1,n)), n1 = ncase, n2=ncase)
table <- spss.BasePivotTable("Tests of whether a correlation matrix is an identity
matrix","OMS")
rowdim <- BasePivotTable.Append(table,Dimension.Place.row,"")
coldim <- BasePivotTable.Append(table,Dimension.Place.column,"Chisquare tests")
row1 <- spss.CellText.String("Bartlett's Test")
row2 <- spss.CellText.String("Steiger Test")
row3 <- spss.CellText.String("Jennrich Test")
col1 <- spss.CellText.String("Chisquare")
col2 <- spss.CellText.String("Degrees of freedom")
col3<- spss.CellText.String("p-value")
BasePivotTable.SetCategories(table,rowdim,list(row1,row2, row3))
BasePivotTable.SetCategories(table,coldim,list(col1,col2,col3))
BasePivotTable.SetCellsByColumn(table,col1,list(spss.CellText.Number(bar$chisq),spss.Cell
Text.Number(bar2$chi2),spss.CellText.Number(bar3$chi2)))
BasePivotTable.SetCellsByColumn(table,col2,list(spss.CellText.Number(bar$df,6),spss.CellT
ext.Number(bar2$df,6),spss.CellText.Number(bar2$df,6)))
BasePivotTable.SetCellsByColumn(table,col3,list(spss.CellText.Number(bar$p.value),spss.C
ellText.Number(bar2$prob),spss.CellText.Number(bar3$prob))) }
# Chisquare test for Maximum Likelihood
if (""=="ml") {
table <- spss.BasePivotTable("Chisquare test for Maximum Likelihood procedure","OMS")
rowdim <- BasePivotTable.Append(table,Dimension.Place.row,"")
coldim <- BasePivotTable.Append(table,Dimension.Place.column,"Chisquare test")
row1 <- spss.CellText.String("Test")
col1 <- spss.CellText.String("Chisquare")
col2 <- spss.CellText.String("Degrees of freedom")
col3<- spss.CellText.String("p-value")
BasePivotTable.SetCategories(table,rowdim,list(row1))
BasePivotTable.SetCategories(table,coldim,list(col1,col2,col3))
BasePivotTable.SetCellsByColumn(table,col1,list(spss.CellText.Number(ad$STATISTIC)))
BasePivotTable.SetCellsByColumn(table,col2,list(spss.CellText.Number(ad$dof,6)))
if (is.numeric(ad$PVAL)) {
BasePivotTable.SetCellsByColumn(table,col3,list(spss.CellText.Number(ad$PVAL))) }
else { BasePivotTable.SetCellsByColumn(table,col3,list(spss.CellText.String(ad$PVAL))) }
}
spsspkg.EndProcedure() }
END PROGRAM.
BEGIN PROGRAM R.
# Number of Factors
if ("no"=="yes" || "y"=="y" || "n"=="y") {
if ("a"=="a") {
if (m1==0) { cor1 <- cor(mdata, use="complete.obs", method="pearson"); m1 <-1 }
co <- cor1 }
if ("a"=="b") {
if (m2==0) { co <- hetcor(da,  ML = FALSE, std.err = FALSE, pd=TRUE,
use="complete.obs", digits=6); m2 <-1
cor2 <- co$correlations }
co <- cor2 }
if ("a"=="c") {
```

```
if (m3==0) { co <- hetcor(da,  ML = TRUE, std.err = FALSE, pd=TRUE,
use="complete.obs", digits=6); m3 <-1
cor3 <- co$correlations }
co <- cor3 }
if ("a"=="d") {
if (m4==0) { cor4 <- cor(mdata, use="complete.obs", method="spearman"); m4 <-1 }
co <- cor4 }
if ("a"=="e") {
if (m5==0) { co <- hetcor(dah,  ML = FALSE, std.err = FALSE, pd=TRUE,
use="complete.obs", digits=6); m5 <-1
cor5 <- co$correlations }
co <- cor5 }
if ("a"=="f") {
if (m6==0) { co <- hetcor(dah,  ML = TRUE, std.err = FALSE, pd=TRUE,
use="complete.obs", digits=6); m6 <-1
cor6 <- co$correlations }
co <- cor6 }
spsspkg.StartProcedure ("Number of Components/Factors")
if ("a"=="a") { mat <- "Pearson" }
if ("a"=="b") { mat <- "Polychoric (Two Step)" }
if ("a"=="c") { mat <- "Polychoric (Max. Lik.)" }
if ("a"=="d") { mat <- "Spearman" }
if ("a"=="e") { mat <- "Heterogenous (Two Step)" }
if ("a"=="f") { mat <- "Heterogenous (Max. Lik.)" }
if ("complete.obs"=="pairwise.complete.obs") { caso <- "pairwise"}
if ("complete.obs"=="complete.obs") { caso <- "listwise"}
junto <- data.frame(mat, caso)
names(junto) <- c("Correlation Matrix", "Missing values")
spsspivottable.Display(junto,
title="Number of Components/Factors estimated for the following Correlation Matrix",
hiderowdimlabel=TRUE) }
# Cattel Scree Test
if ("no"=="yes" && n >= 3) {
if (""!="") { set.seed() }
repi <-
evpea <- matrix(NA, ncol = n, nrow = repi)
evpea_b <- matrix(NA, ncol = n, nrow = repi)
# Eigenvalues from components or factors
if (""=="components") {
evt <- eigen(co, only.values = TRUE) }
else { evt <- eigen(co - ginv(diag(diag(ginv(co)))), only.values=TRUE) }
# Simulations
quant <- function(x, sprobs = sprobs) { return(as.vector(quantile(x, probs = c()))) }
# standardized normal
if ("" == "normal") {
co_used <- "Pearson"
for (k in 1:repi) {
y <- mvrnorm(ncase, rep(0, n), diag(1,n), empirical = FALSE)
corY <- cor(y, use="complete.obs", method="pearson")
if (""=="factors") { corY <- corY - ginv(diag(diag(ginv(corY)))) }
ev <- eigen(corY, only.values = TRUE)
```

```
evpea[k, ] <- ev$values }
mevpea <- sapply(as.data.frame(evpea), mean)
qevpea <- sapply(as.data.frame(evpea), quant)
ap <- list(eigen = data.frame(mevpea, qevpea)) }
# permutation
if ("" == "permutation") {
sdata <- matrix (0,ncol = ncol(mdata), nrow =ncase_list)
if (""=="y") {
mdata2 <- as.matrix(mdata)
lcor <- 0
for (i in 1:ncase2) {
ind <- 0
j <- 0
while (j<n && ind ==0) {
j <- j+1
if (is.na(mdata[i,j])) { ind <- 1 } }
if (ind==0) { lcor <- lcor+1; sdata[lcor,] <- mdata2[i,] } } }
else { sdata <- mdata }
for (k in 1:repi) {
perm <- apply(sdata[,2:n], 2, sample, replace = )
perm <- data.frame(sdata[,1],perm)
if (""=="a") { corY <- cor(perm, use="complete.obs", method="pearson"); co_used <-
"Pearson" }
else { if (""=="b") {
co_used <- "Polychoric (Two Step)"
for (i in 1:n) { perm[,i]<-ordered(perm[,i]) }
corY <-  hetcor(perm,  ML = FALSE, std.err = FALSE, pd=TRUE, use="complete.obs",
digits=4)
corY <- corY$correlations }
else { if (""=="c") {
co_used <- "Polychoric (Max. Likelihood)"
for (i in 1:n) { perm[,i]<-ordered(perm[,i]) }
corY <-  hetcor(perm,  ML = TRUE, std.err = FALSE, pd=TRUE, use="complete.obs",
digits=4)
corY <- corY$correlations }
else { if (""=="e") {
co_used <- "Heterogeneous (Two Step)"
for (i in 1:n) {
if  ( scal["varMeasurementLevel",i]=="nominal") { perm[,i]<-factor(perm[,i]) }
else { if  (scal["varMeasurementLevel",i]=="ordinal") { perm[,i]<-ordered(perm[,i]) } } }
corY <-  hetcor(perm,  ML = FALSE, std.err = FALSE, pd=TRUE, use="complete.obs",
digits=4)
corY <- corY$correlations }
else { if (""=="f") {
co_used <- "Heterogeneous (Max. Likelihood)"
for (i in 1:n) {
if  ( scal["varMeasurementLevel",i]=="nominal") { perm[,i]<-factor(perm[,i]) }
else { if  (scal["varMeasurementLevel",i]=="ordinal") { perm[,i]<-ordered(perm[,i]) } } }
corY <-  hetcor(perm,  ML = TRUE, std.err = FALSE, pd=TRUE, use="complete.obs",
digits=4)
corY <- corY$correlations }
```

```
else { corY <- cor(perm, use="complete.obs", method="spearman"); co_used <- "Spearman"
} } } } }
if (""=="factors") { corY <- corY - ginv(diag(diag(ginv(corY)))) }
ev <- eigen(corY, only.values = TRUE)
evpea[k, ] <- ev$values }
mevpea <- sapply(as.data.frame(evpea), mean)
qevpea <- sapply(as.data.frame(evpea), quant)
ap <- list(eigen = data.frame(mevpea, qevpea) ) }
# parallel analysis and scree
nS <- nScree(evt$values, aparallel=)
s <- ""
if ( s=="ap$eigen$mevpea") { ss <- "mean" } else { ss <- "quantile" }
if (""=="factors") { criteria <- mean(evt$values)
nS$Components$nkaiser <- sum(evt$values >= rep(criteria, n))
p.vec <- which(evt$values >= ap$eigen$qevpea)
nS$Components$nparallel <- sum(p.vec == (1:length(p.vec))) }
junto <- data.frame(repi, "", "", ss, mat, co_used)
names(junto) <- c("Number of Samples","Model","Data","Measure",
"Correlation matrix","Matrix for simulations")
spsspivottable.Display(junto,
title="Parameters for Parallel Analysis",
hiderowdimlabel=TRUE, format=6)
junto <- data.frame(paste("",1:n, rep=""), ap$eigen)
names(junto) <- c("Order", "Mean", "Quantile selected")
spsspivottable.Display(junto,
title="Distribution of the eigenvalues computed",
hiderowdimlabel=TRUE)
junto <- data.frame(nS$Components)
names(junto) <- c("Optimal coordinates","Acceleration factor",
"Parallel analysis","Kaiser rule")
spsspivottable.Display(junto,
title="Number of components/factors to retain according to different rules",
hiderowdimlabel=TRUE, format=6)
junto <- data.frame(nS$Analysis)
names(junto) <- c("Eigenvalues","Proportion of variance", "Cumulative", "Parallel analysis",
"Predicted eigenvalues", "OC", "Acceleration factor", "AF")
spsspivottable.Display(junto,
title="Data linked to the different rules")
if (""=="components") {
if (""=="normal") {
plotnScree(nS, ylab = "Eigenvalues", xlab = "Components",
main = "Parallel analysis on random uncorrelated standardized normal") }
if (""=="permutation") {
plotnScree(nS, ylab = "Eigenvalues", xlab = "Components",
main = "Parallel analysis on data permutation") } }
else {
if (""=="normal") {
plotnScree(nS, ylab = "Eigenvalues", xlab = "Factors",
main = "Parallel analysis on random uncorrelated standardized normal") }
if (""=="permutation") {
plotnScree(nS, ylab = "Eigenvalues", xlab = "Factors",
```

```
main = "Parallel analysis on data permutation") } } }
# Velicer MAP
if ("y"=="y") {
ev <- eigen(co, symmetric=TRUE)
load2 <- ev$vectors%*%sqrt(diag(ev$values))
f <- rep(0,(n-1)); fq <- rep(0,(n-1))
som <- sum(co^2); somq <- sum(co^4)
m <- n*(n-1); f[1] <- (som-n)/m; fq[1] <- (somq-n)/m
for (i in 1:(n-2)) { b <- load2[,1:i]; cc <- co-(b%*%t(b))
d <- diag(1/sqrt(diag(cc))); ea <- d%*%cc%*%d      #partial correlation matrix controlling
components
som <- sum(ea^2); somq <- sum(ea^4)
f[i+1] <- (som-n)/m; fq[i+1] <- (somq-n)/m }
matt <- replace(f, f == "NaN", 2)
matt2 <- replace(fq, fq == "NaN", 2)
fm <- min(matt); fqm <- min(matt2)
for (i in 1:(n-1)) {
if (f[i]==fm) { fma <- i-1; break } }
for (i in 1:(n-1)) {
if (fq[i]==fqm) { fqma <- i-1; break } }
junto <- data.frame(f[1:(n-1)],fq[1:(n-1)],row.names=0:(n-2))
names(junto) <- c("Squared average partial correlations", "4th average partial correlations")
spsspivottable.Display(junto,
title="Velicer's MAP values")
table <- spss.BasePivotTable("Velicer's Minimum Average Partial Test","OMS")
rowdim <- BasePivotTable.Append(table,Dimension.Place.row,"")
coldim <- BasePivotTable.Append(table,Dimension.Place.column,"Velicer's Minimum")
row1 <- spss.CellText.String("Squared MAP")
row2 <- spss.CellText.String("4th power MAP")
col1 <- spss.CellText.String("Minimum")
col2 <- spss.CellText.String("Components to retain")
BasePivotTable.SetCategories(table,rowdim,list(row1, row2))
BasePivotTable.SetCategories(table,coldim,list(col1,col2))
BasePivotTable.SetCellsByRow(table,row1,list(spss.CellText.Number(fm),
spss.CellText.Number(fma,6)))
BasePivotTable.SetCellsByRow(table,row2,list(spss.CellText.Number(fqm),
spss.CellText.Number(fqma,6))) }
# Very Simple Structure
if ("n"=="y") {
v <- VSS(co, n = , rotate = "", diagonal = , fm = "", n.obs=ncase, plot=TRUE)
junto <- data.frame(v$cfit.1, v$cfit.2)
names(junto) <- c("Complexity 1","Complexity 2")
spsspivottable.Display(junto,
title="Very Simple Structure")
nu <-
vss1 <- 0
for (i in 1:nu) {
vss1 <- vss1+1
if ( v$cfit.1[i]==max(v$cfit.1) ) { break }
 }
vss2 <- 0
```

```
for (i in 1:nu) {
vss2 <- vss2+1
if ( v$cfit.2[i]==max(v$cfit.2) ) { break }
 }
table <- spss.BasePivotTable("Very Simple Structure (Number of factors)","OMS")
rowdim <- BasePivotTable.Append(table,Dimension.Place.row,"")
coldim <- BasePivotTable.Append(table,Dimension.Place.column,"Very Simple Structure
(VSS)")
row1 <- spss.CellText.String("Values")
col1 <- spss.CellText.String("Rotation")
col2 <- spss.CellText.String("Factoring method")
col3 <- spss.CellText.String("Max VSS complexity 1")
col4 <- spss.CellText.String("N factors complexity 1")
col5 <- spss.CellText.String("Max VSS complexity 2")
col6 <- spss.CellText.String("N factors complexity 2")
BasePivotTable.SetCategories(table,rowdim,list(row1))
BasePivotTable.SetCategories(table,coldim,list(col1,col2,col3,col4,col5,col6))
BasePivotTable.SetCellsByColumn(table,col1,list(spss.CellText.String("")))
BasePivotTable.SetCellsByColumn(table,col2,list(spss.CellText.String("")))
BasePivotTable.SetCellsByColumn(table,col3,list(spss.CellText.Number(max(v$cfit.1))))
BasePivotTable.SetCellsByColumn(table,col4,list(spss.CellText.Number(vss1,6)))
BasePivotTable.SetCellsByColumn(table,col5,list(spss.CellText.Number(max(v$cfit.2))))
BasePivotTable.SetCellsByColumn(table,col6,list(spss.CellText.Number(vss2,6)))
 }
spsspkg.EndProcedure()
END PROGRAM.
BEGIN PROGRAM R.
# More Number of Factors
if("no"=="yes") {
spsspkg.StartProcedure ("Comparison Data")
N.Pop <-
sdata <- matrix (0,ncol = ncol(mdata), nrow =ncase_list)
if (""=="yes") {
mdata2 <- as.matrix(mdata)
lcor <- 0
for (i in 1:ncase2) {
ind <- 0
j <- 0
while (j<n && ind ==0) {
j <- j+1
if (is.na(mdata[i,j])) { ind <- 1 } }
if (ind==0) { lcor <- lcor+1; sdata[lcor,] <- mdata2[i,] } } }
else { sdata <- mdata }
# Matrix to store each variable score distribution
Distributions <- matrix(0, nrow = N.Pop, ncol = dim(sdata)[2])
# Generate distribution for each variable
if (""!="") { set.seed() }
b2 <- matrix(0, nrow = dim(sdata)[2], ncol = 1)
for (i in 1:dim(sdata)[2]) {
bb <- sort(sample(sdata[,i], size = N.Pop, replace = T) )
b2[i] <- length(bb)
```

```
ac <- N.Pop-b2[i]
if (b2[i] < N.Pop) {
for (j in 1:ac) {
qw <- sample(1: (b2[i] + j),1)
bb <- insert (bb, qw, NA) } }
Distributions[,i] <- as.matrix(bb) }
EFA.Comp.Data <- function(Data, F.Max, N.Samples=500, Alpha = 0.3) {
# Data = N by k data matrix
# F.Max = largest number of factors to consider
# N.Pop = size of finite populations of comparison data
# N.Samples = number of samples drawn from each population
# Alpha = alpha level testing significance of improvement with adding factor
sig2 <- data.frame(1)
N <- dim(Data)[1]
k <- dim(Data)[2]
Data2 <- as.data.frame(Data)
if (""=="a") { cor.Data <- cor(Data,use="complete.obs") }
if (""=="b") {
for (i in 1:k) { Data2[,i]<-ordered(Data[,i])  }
corY <-  hetcor(Data2,  ML = FALSE, std.err = FALSE, pd=TRUE, use="complete.obs",
digits=4)
cor.Data <- corY$correlations }
if (""=="c") {
for (i in 1:k) { Data2[,i]<-ordered(Data[,i])  }
corY <-  hetcor(Data2,  ML = TRUE, std.err = FALSE, pd=TRUE, use="complete.obs",
digits=4)
cor.Data <- corY$correlations }
if (""=="d") { cor.Data <- cor(Data,use="complete.obs",method="spearman") }
if (""=="e") {
for (i in 1:k) {
if  ( scal["varMeasurementLevel",i]=="nominal") { Data2[,i]<-factor(Data[,i]) }
else { if  (scal["varMeasurementLevel",i]=="ordinal") { Data2[,i]<-ordered(Data[,i]) } } }
corY <-  hetcor(Data2,  ML = FALSE, std.err = FALSE, pd=TRUE, use="complete.obs",
digits=4)
cor.Data <- corY$correlations }
if (""=="f") {
for (i in 1:k) {
if  ( scal["varMeasurementLevel",i]=="nominal") { Data2[,i]<-factor(Data[,i]) }
else { if  (scal["varMeasurementLevel",i]=="ordinal") { Data2[,i]<-ordered(Data[,i]) } } }
corY <-  hetcor(Data2,  ML = TRUE, std.err = FALSE, pd=TRUE, use="complete.obs",
digits=4)
cor.Data <- corY$correlations }
Eigs.Data <- eigen(cor.Data)$values
RMSR.Eigs <- matrix(0, nrow = N.Samples, ncol = F.Max)
Sig <- T
F.CD <- 1
nf <- -1
while (F.CD <= F.Max) {
Pop <- GenData(Data, N.Factors = F.CD, N = N.Pop, Target.Corr = cor.Data)
for (j in 1:N.Samples) {
Samp <- Pop[sample(1:N.Pop, size = N, replace = T),]
```

```
Samp2 <- as.data.frame(Samp)
if (""=="a") { cor.Samp <- cor(Samp,use="complete.obs") }
if (""=="b") {
for (i in 1:k) { Samp2[,i]<-ordered(Samp[,i])  }
corY <-  hetcor(Samp2,  ML = FALSE, std.err = FALSE, pd=TRUE, use="complete.obs",
digits=4)
cor.Samp <- corY$correlations }
if (""=="c") {
for (i in 1:k) { Samp2[,i]<-ordered(Samp[,i])  }
corY <-  hetcor(Samp2,  ML = TRUE, std.err = FALSE, pd=TRUE, use="complete.obs",
digits=4)
cor.Samp <- corY$correlations }
if (""=="d") { cor.Samp <- cor(Samp,use="complete.obs",method="spearman") }
if (""=="e") {
for (i in 1:k) {
if  ( scal["varMeasurementLevel",i]=="nominal") { Samp2[,i]<-factor(Samp[,i]) }
else { if  (scal["varMeasurementLevel",i]=="ordinal") { Samp2[,i]<-ordered(Samp[,i]) } } }
corY <-  hetcor(Samp2,  ML = FALSE, std.err = FALSE, pd=TRUE, use="complete.obs",
digits=4)
cor.Samp <- corY$correlations }
if (""=="f") {
for (i in 1:k) {
if  ( scal["varMeasurementLevel",i]=="nominal") { Samp2[,i]<-factor(Samp[,i]) }
else { if  (scal["varMeasurementLevel",i]=="ordinal") { Samp2[,i]<-ordered(Samp[,i]) } } }
corY <-  hetcor(Samp2,  ML = TRUE, std.err = FALSE, pd=TRUE, use="complete.obs",
digits=4)
cor.Samp <- corY$correlations }
Eigs.Samp <- eigen(cor.Samp)$values
RMSR.Eigs[j,F.CD] <- sqrt(sum((Eigs.Samp - Eigs.Data) * (Eigs.Samp - Eigs.Data)) / k) }
if (F.CD > 1) {
sig2[F.CD, ] <- wilcox.test(RMSR.Eigs[,F.CD], RMSR.Eigs[,(F.CD - 1)], "less")$p.value }
else { sig2[1,] <- NA }
if (F.CD > 1)  {
Sig <- (wilcox.test(RMSR.Eigs[,F.CD], RMSR.Eigs[,(F.CD - 1)], "less")$p.value < Alpha) }
F.CD <- F.CD + 1
if (Sig == FALSE && nf == -1) {
nf = F.CD - 2
if (""=="yes") { break }
}
 }
if (nf == -1) { nf = F.Max }
# graph
if (""=="yes") { x.max <- min(nf+1, F.Max) }
else { x.max <- F.Max }
ys <- apply(RMSR.Eigs[,1:x.max], 2, mean)
plot(x = 1:x.max, y = ys, ylim = c(0, max(ys)), xlab = "Factor",
ylab = "RMSR Eigenvalue", type = "b", main = "Fit to Comparison Data")
abline(v = nf, lty = 3)
lis <- list(ys = ys, nf = nf, sig2 = sig2, x.max = x.max)
return(lis)
 }
```

```
GenData <- function(Supplied.Data, N.Factors, N, Max.Trials = 5, Initial.Multiplier = 1,
Target.Corr)
{
# Ruscio, J., & Kaczetow, W. (2008).
# Simulating multivariate nonnormal data using an iterative algorithm.
# Multivariate Behavioral Research, 43(3), 355-381.
k <- dim(Supplied.Data)[2]
Iteration <- 0   # Iteration counter
Best.RMSR <- 1   # Lowest RMSR correlation
Trials.Without.Improvement <- 0   # Trial counter
Data <- matrix(0, nrow = N, ncol = k)   # Matrix to store the simulated data
# Calculate and store a copy of the target correlation matrix
Intermediate.Corr <- Target.Corr
# Generate random normal data, initialize factor loadings
Shared.Comp <- matrix(rnorm(N * N.Factors, 0, 1), nrow = N, ncol = N.Factors)
Unique.Comp <- matrix(rnorm(N.Pop * dim(sdata)[2], 0, 1), nrow = N.Pop, ncol =
dim(sdata)[2])
Shared.Load <- matrix(0, nrow = k, ncol = N.Factors)
Unique.Load <- matrix(0, nrow = k, ncol = 1)
# Begin loop that ends when specified n of iterations pass without improvement in RMSR
correlation
while (Trials.Without.Improvement < Max.Trials)  {
Iteration <- Iteration + 1
# Calculate factor loadings and apply to reproduce desired correlations
Fact.Anal <- Factor.Analysis(Intermediate.Corr, N.Factors = N.Factors)
if (N.Factors == 1) { Shared.Load[,1] <- Fact.Anal$loadings }
else {
for (i in 1:N.Factors) { Shared.Load[,i] <- Fact.Anal$loadings[,i] } }
Shared.Load[Shared.Load > 1] <- 1
Shared.Load[Shared.Load < -1] <- -1
if (Shared.Load[1,1] < 0) { Shared.Load <- Shared.Load * -1 }
for (i in 1:k) {
if (sum(Shared.Load[i,] * Shared.Load[i,]) < 1) {
Unique.Load[i,1] <- (1 - sum(Shared.Load[i,] * Shared.Load[i,])) }
else { Unique.Load[i,1] <- 0 } }
Unique.Load <- sqrt(Unique.Load)
for (i in 1:k) {
Data[,i] <- (Shared.Comp %*% t(Shared.Load))[,i] + Unique.Comp[,i] * Unique.Load[i,1] }
# Replace normal with nonnormal distributions
for (i in 1:k) {
Data <- Data[sort.list(Data[,i]),]
Data[,i] <- Distributions[,i] }
# Calculate RMSR correlation, compare to lowest value, take appropriate action
Data2 <- as.data.frame(Data)
if (""=="a") { Reproduced.Corr <- cor(Data,use="complete.obs") }
if (""=="b") {
for (i in 1:k) { Data2[,i]<-ordered(Data[,i])  }
corY <-  hetcor(Data2,  ML = FALSE, std.err = FALSE, pd=TRUE, use="complete.obs",
digits=4)
Reproduced.Corr <- corY$correlations }
if (""=="c") {
```

```
for (i in 1:k) { Data2[,i]<-ordered(Data[,i])  }
corY <-  hetcor(Data2,  ML = TRUE, std.err = FALSE, pd=TRUE, use="complete.obs",
digits=4)
Reproduced.Corr <- corY$correlations }
if (""=="d") { Reproduced.Corr <- cor(Data,use="complete.obs",method="spearman") }
if (""=="e") {
for (i in 1:k) {
if  ( scal["varMeasurementLevel",i]=="nominal") { Data2[,i]<-factor(Data[,i]) }
else { if  (scal["varMeasurementLevel",i]=="ordinal") { Data2[,i]<-ordered(Data[,i]) } } }
corY <-  hetcor(Data2,  ML = FALSE, std.err = FALSE, pd=TRUE, use="complete.obs",
digits=4)
Reproduced.Corr <- corY$correlations }
if (""=="f") {
for (i in 1:k) {
if  ( scal["varMeasurementLevel",i]=="nominal") { Data2[,i]<-factor(Data[,i]) }
else { if  (scal["varMeasurementLevel",i]=="ordinal") { Data2[,i]<-ordered(Data[,i]) } } }
corY <-  hetcor(Data2,  ML = TRUE, std.err = FALSE, pd=TRUE, use="complete.obs",
digits=4)
Reproduced.Corr <- corY$correlations }
Residual.Corr <- Target.Corr - Reproduced.Corr
RMSR <-
sqrt(sum(Residual.Corr[lower.tri(Residual.Corr)]*Residual.Corr[lower.tri(Residual.Corr)]) /
(.5* (k*k - k)))
if (RMSR < Best.RMSR)  {
Best.RMSR <- RMSR
Best.Corr <- Intermediate.Corr
Best.Res <- Residual.Corr
Intermediate.Corr <- Intermediate.Corr + Initial.Multiplier * Residual.Corr
Trials.Without.Improvement <- 0 }
else  {
Trials.Without.Improvement <- Trials.Without.Improvement + 1
Current.Multiplier <- Initial.Multiplier * .5 ^ Trials.Without.Improvement
Intermediate.Corr <- Best.Corr + Current.Multiplier * Best.Res }  }
# Construct the data set with the lowest RMSR correlation
Fact.Anal <- Factor.Analysis(Best.Corr, N.Factors = N.Factors)
if (N.Factors == 1) { Shared.Load[,1] <- Fact.Anal$loadings }
else {
for (i in 1:N.Factors) {
Shared.Load[,i] <- Fact.Anal$loadings[,i] } }
Shared.Load[Shared.Load > 1] <- 1
Shared.Load[Shared.Load < -1] <- -1
if (Shared.Load[1,1] < 0) { Shared.Load <- Shared.Load * -1 }
for (i in 1:k) {
if (sum(Shared.Load[i,] * Shared.Load[i,]) < 1)  {
Unique.Load[i,1] <- (1 - sum(Shared.Load[i,] * Shared.Load[i,])) }
else { Unique.Load[i,1] <- 0 } }
Unique.Load <- sqrt(Unique.Load)
for (i in 1:k) {
Data[,i] <- (Shared.Comp %*% t(Shared.Load))[,i] + Unique.Comp[,i] * Unique.Load[i,1] }
Data <- apply(Data, 2, scale) # standardizes each variable in the matrix
for (i in 1:k) {
```

```
Data <- Data[sort.list(Data[,i]),]
Data[,i] <- Distributions[,i]  }
# Return the simulated data set
return(Data) }
Factor.Analysis <- function(Data, Max.Iter = 50, N.Factors = 0)
{
Data <- as.matrix(Data)
k <- dim(Data)[2]
if (N.Factors == 0) {
N.Factors <- k
Determine <- T  }
else { Determine <- F }
Cor.Matrix <- Data
Criterion <- .001
Old.H2 <- rep(99, k)
H2 <- rep(0, k)
Change <- 1
Iter <- 0
Factor.Loadings <- matrix(nrow = k, ncol = N.Factors)
while ((Change >= Criterion) & (Iter < Max.Iter)) {
Iter <- Iter + 1
Eig <- eigen(Cor.Matrix)
L <- sqrt(Eig$values[1:N.Factors])
for (i in 1:N.Factors) {
Factor.Loadings[,i] <- Eig$vectors[,i] * L[i] }
for (i in 1:k) {
H2[i] <- sum(Factor.Loadings[i,] * Factor.Loadings[i,]) }
Change <- max(abs(Old.H2 - H2))
Old.H2 <- H2
diag(Cor.Matrix) <- H2
   }
if (Determine) { N.Factors <- sum(Eig$values > 1) }
return(list(loadings = Factor.Loadings[,1:N.Factors], factors = N.Factors))
}
ysa<-EFA.Comp.Data(Data = sdata, F.Max =  ,
N.Samples = , Alpha = )
junto <- data.frame(paste("",1:ysa$x.max, rep = "factor"), ysa$ys, ysa$sig2)
names(junto) <- c("Number of factors","RMSR Eigenvalue","p-value")
spsspivottable.Display(junto,
title="Fit to Comparison Data", hiderowdimlabel=TRUE )
if (""=="a") { mat <- "Pearson" }
else { if (""=="b") { mat <- "Polychoric (Two Step)" }
else { if (""=="c") { mat <- "Polychoric (Max. Lik.)" }
else { if (""=="d") { mat <- "Spearman" }
else { if (""=="e") { mat <- "Heterogenous (Two Step)" }
else { if (""=="f") { mat <- "Heterogenous (Max. Lik.)" }
 } } } } }
junto <- data.frame(mat,ysa$nf)
names(junto) <- c("Correlations","Number of factors to retain")
spsspivottable.Display(junto,
title="Comparison Data", hiderowdimlabel=TRUE, format=6)
```

```
spsspkg.EndProcedure()
 }
END PROGRAM.
BEGIN PROGRAM R.
# Items, Scales and Realibility
ind <- 0
spsspkg.StartProcedure ("Items, Scales and Realibility")
if ("n"=="y" || "n"=="y") {
if ("no"=="yes" && "a"=="a") { w <- factor2cluster(load, cut = 0);nu <- ncol(w); ind <- 1; w
<- as.data.frame(w); names(w) <- paste ("S",1:nu, rep="") }
if ("a"=="m") { ke <- list(c(1,3,-4),c(6,7))
w <- make.keys(n, ke, key.labels = NULL, item.labels = nam);nu <- ncol(w); ind <- 1; w <-
as.data.frame(w); names(w) <- paste ("S",1:nu, rep="") }
if ("a"=="all") { S1 <- rep(1,n); w <- data.frame(S1, row.names=nam); ind <- 1 }
if (ind==1) {
nu <- ncol(w)
spsspivottable.Display(w,
title="Assigned items to clusters (scores are adjusted for reverse scored items)", format=6)
if ("n"=="y") {
see <- scoreItems(w, mdata, totals = , ilabels = nam, missing = , impute="", delete=FALSE,
digits=6) }
else { see <- scoreItems(w, mdata, totals = TRUE, ilabels = nam, missing = TRUE,
impute="median", delete=FALSE, digits=6) }
junto <- data.frame(see$n.items)
names(junto) <- c("N Items")
spsspivottable.Display(junto,
title="Number of items for each scale",
format=6)
junto <- data.frame(see$av.r)
names(junto) <- paste ("S",1:nu, rep="")
spsspivottable.Display(junto, hiderowdimlabel=TRUE,
title="Average correlation")
spsspivottable.Display(see$cor, title="Intercorrelation of scales")
see$corrected[lower.tri(see$corrected)] <- t(see$corrected)[lower.tri(t(see$corrected))]
spsspivottable.Display(see$corrected,
title="Unattenuated correlations of scales (alpha on the diagonal)")
spsspivottable.Display(see$item.cor,
title="Correlation of items with scales (not corrected)")
spsspivottable.Display(see$item.corrected,
title=" Correlation of items with scales (corrected for item overlap)")
# Beggining Cronbach alpha
if (""=="y") {
cc <- cov(mdata, use="complete.obs", method="pearson")
alpha.u <- rep(0,nu)
alpha.s <- rep(0,nu)
if (m1==0) { co <- cor(mdata, use="complete.obs", method="pearson"); cor1 <- co; m1 <- 1 }
else { co <- cor1 }
for (i in 1:nu) {
ke <- diag(w[,i]) %*% co %*% diag(w[,i])
ke2 <- diag(w[,i]) %*% cc %*% diag(w[,i])
s <- sum(abs(w[,i]))
```

```
alpha.s[i] <- (1 - tr(ke)/sum(ke)) * (s/(s - 1))
alpha.u[i] <- (1 - tr(ke2)/sum(ke2)) * (s/(s - 1)) }
junto <- data.frame(t(alpha.u))
names(junto) <- paste ("S",1:nu, rep="")
spsspivottable.Display(junto, hiderowdimlabel=TRUE,
title="Raw Cronbach alpha")
junto <- data.frame(t(alpha.s))
names(junto) <- paste ("S",1:nu, rep="")
spsspivottable.Display(junto, hiderowdimlabel=TRUE,
title="Standardized Cronbach alpha")
w2 <- w
alp.u <- matrix (nrow=n,ncol=nu)
alp.s <- matrix (nrow=n,ncol=nu)
for (i in 1:n) {
for (j in 1:nu) {
if (w[i,j] != 0) { w2[i,j] = 0
ke <- diag(w2[,j]) %*% co %*% diag(w2[,j])
ke2 <- diag(w2[,j]) %*% cc %*% diag(w2[,j])
s <- sum(abs(w2[,j]))
alp.s[i,j] <- (1 - tr(ke)/sum(ke)) * (s/(s - 1))
alp.u[i,j] <- (1 - tr(ke2)/sum(ke2)) * (s/(s - 1))
w2 <- w } } }
junto <- data.frame(alp.u, row.names=nam)
names(junto) <- paste ("S",1:nu, rep="")
spsspivottable.Display(junto,
title="Raw Cronbach alpha if item deleted")
junto <- data.frame(alp.s, row.names=nam)
names(junto) <- paste ("S",1:nu, rep="")
spsspivottable.Display(junto,
title="Standardized Cronbach alpha if item deleted") } # end Cronbach alpha
# Beginning ordinal Cronbach alpha
if (""=="y") {
if (""=="a") {
if (m2==0) { co <- hetcor(da,  ML = FALSE, std.err = FALSE, pd=TRUE,
use="complete.obs", digits=6)
co <- co$correlations; cor2 <- co; m2 <- 1 }
else { co <- cor2 } }
if (""=="b") {
if (m3==0) { co <- hetcor(da,  ML = TRUE, std.err = FALSE, pd=TRUE,
use="complete.obs", digits=6)
co <- co$correlations; cor3<- co; m3 <- 1 }
else { co <- cor3 } }
or.al <- rep(0,nu)
for (i in 1:nu) {
ke <- diag(w[,i]) %*% co %*% diag(w[,i])
s <- sum(abs(w[,i]))
or.al[i] <- (1 - tr(ke)/sum(ke)) * (s/(s - 1)) }
junto <- data.frame(t(or.al))
names(junto) <- paste ("S",1:nu, rep="")
spsspivottable.Display(junto, hiderowdimlabel=TRUE,
title="Ordinal coefficient alpha")
```

```
w2 <- w
or.al <- matrix (nrow=n,ncol=nu)
for (i in 1:n) {
for (j in 1:nu) {
if (w[i,j] != 0) { w2[i,j] = 0
ke <- diag(w2[,j]) %*% co %*% diag(w2[,j])
s <- sum(abs(w2[,j]))
or.al[i,j] <- (1 - tr(ke)/sum(ke)) * (s/(s - 1))
w2 <- w } } }
junto <- data.frame(or.al, row.names=nam)
names(junto) <- paste ("S",1:nu, rep="")
spsspivottable.Display(junto,
title="Ordinal coefficient alpha if item deleted") } # end ordinal Cronbach alpha
# Beginning Armor's reliability theta
if (""=="y")  {
armo <- rep(0,nu)
co3 <- list()
if (m1==0) { co <- cor(mdata, use="complete.obs", method="pearson") }
else { co <- cor1 }
for (j in 1:nu) {
con <- 0
co2 <- co
for (i in 1:n) {
if (w[i,j] == 0) { co2 <- co2[-(i-con),-(i-con)]; con <- con+1 }
else { if (w[i,j] == -1) { co2[i-con,] <- (-1)*co2[i-con,]; co2[,i-con] <- (-1)*co2[,i-con] } } }
co3[[j]] <- co2
ei <- eigen(co2, symmetric=TRUE, only.values=TRUE)
ei <- ei$values
s <- dim(as.matrix(co2))[1]
armo[j] <- ((ei[1]-1)/ei[1]) * (s/(s - 1)) }
junto <- data.frame(t(armo))
names(junto) <- paste ("S",1:nu, rep="")
spsspivottable.Display(junto, hiderowdimlabel=TRUE,
title="Armor's reliability theta")
armo <- matrix (nrow=n,ncol=nu)
indi <- list()
for (j in 1:nu) {
con <- 1
for (i in 1:n) { if (w[i,j] != 0) {indi[[con]] <- i; con <- con+1} }
s <- dim(as.matrix(co3[[j]]))[1]
for (i in 1:s) {
co2 <- as.matrix(co3[[j]])
if (dim(co2)[1] > 1) {
co2 <- co2[-i,-i]
ei <- eigen(co2, symmetric=TRUE, only.values=TRUE)
ei <- ei$values
armo[indi[[i]],j] <- ((ei[1]-1)/ei[1]) * ((s-1)/(s - 2)) } } }
junto <- data.frame(armo, row.names=nam)
names(junto) <- paste ("S",1:nu, rep="")
spsspivottable.Display(junto,
title="Armor's reliability theta if item deleted") } # end Armor's reliability theta
```

```
# Beginning ordinal Armor's reliability theta
if (""=="y")  {
armo <- rep(0,nu)
co3 <- list()
if (""=="a") {
if (m2==0) {
co <- hetcor(da,  ML = FALSE, std.err = FALSE, pd=TRUE, use="complete.obs", digits=6)
co <- co$correlations; cor2 <- co; m2 <- 1 }
else { co <- cor2 } }
if (""=="b") {
if (m3==0) {
co <- hetcor(da,  ML = TRUE, std.err = FALSE, pd=TRUE, use="complete.obs", digits=6)
co <- co$correlations; cor3 <- co; m3 <-1 }
else { co <- cor3 } }
for (j in 1:nu) {
con <- 0
co2 <- co
for (i in 1:n) {
if (w[i,j] == 0) { co2 <- co2[-(i-con),-(i-con)]; con <- con+1 }
else { if (w[i,j] == -1) { co2[i-con,] <- (-1)*co2[i-con,]; co2[,i-con] <- (-1)*co2[,i-con] } } }
co3[[j]] <- co2
ei <- eigen(co2, symmetric=TRUE, only.values=TRUE)
ei <- ei$values
s <- dim(as.matrix(co2))[1]
armo[j] <- ((ei[1]-1)/ei[1]) * (s/(s - 1)) }
junto <- data.frame(t(armo))
names(junto) <- paste ("S",1:nu, rep="")
spsspivottable.Display(junto, hiderowdimlabel=TRUE,
title="Ordinal coefficient theta")
armo <- matrix (nrow=n,ncol=nu)
indi <- list()
for (j in 1:nu) {
con <- 1
for (i in 1:n) { if (w[i,j] != 0) {indi[[con]] <- i; con <- con+1} }
s <- dim(as.matrix(co3[[j]]))[1]
for (i in 1:s) {
co2 <- as.matrix(co3[[j]])
if (dim(co2)[1]>1) {
co2 <- co2[-i,-i]
ei <- eigen(co2, symmetric=TRUE, only.values=TRUE)
ei <- ei$values
armo[indi[[i]],j] <- ((ei[1]-1)/ei[1]) * ((s-1)/(s - 2)) } } }
junto <- data.frame(armo, row.names=nam)
names(junto) <- paste ("S",1:nu, rep="")
spsspivottable.Display(junto,
title="Ordinal coefficient theta if item deleted") } # end ordinal Armor's reliability theta
 } #end (ind=1)
 }
spsspkg.EndProcedure()
# Saving scores to SPSS
if ("n"=="y") {
```

```
if (""=="y" && ind==1) {
rm(mdata)
dict1 <- spssdictionary.GetDictionaryFromSPSS()
mdata <- spssdata.GetDataFromSPSS()
g <- list()
com <- ncol(w)
for (i in 1:com) { h <- as.character(i)
h <- paste("factor", h, sep="")
g[[i]] <- c(h,"",0,"F8.4","scale") }
dict <- spssdictionary.CreateSPSSDictionary(g)
dict <- data.frame(dict1,dict)
spssdictionary.SetDictionaryToSPSS("",dict)
fa=nrow(mdata)
fb=row.names(as.data.frame(see$scores))
ult=fb[nrow(see$scores)]
dife=fa-as.numeric(ult)
line=rep(NA,ncol(see$scores))
if (com==1) {
if (fb[1] != 1)  {
see$scores=rbind(as.matrix(line),as.matrix(see$scores[1:nrow(see$scores),]))
fb=row.names(as.data.frame(see$scores)) }
for (i in 2:(fa-dife-1)) {
if  (fb[i] != i) {
see$scores=rbind(as.matrix(see$scores[1:(i-
1),]),as.matrix(line),as.matrix(see$scores[i:nrow(see$scores),]))
fb=row.names(as.data.frame(see$scores)) } }
if (dife>0) {
for (i in 1:dife) {
see$scores=rbind(as.matrix(see$scores),as.matrix(line))  } } }
else{
if (fb[1] != 1)  {
see$scores=rbind(line,as.data.frame(see$scores[1:nrow(see$scores),]))
fb=row.names(as.data.frame(see$scores)) }
for (i in 2:(fa-dife-1)) {
if  (fb[i] != i) {
see$scores=rbind(as.data.frame(see$scores[1:(i-
1),]),line,as.data.frame(see$scores[i:nrow(see$scores),]))
fb=row.names(as.data.frame(see$scores)) } }
if (dife>0) {
for (i in 1:dife) {
see$scores=rbind(as.data.frame(see$scores),line)  } } }
see$scores=data.frame(see$scores,row.names=1:fa)
casedata <- data.frame(mdata,see$scores)
spssdata.SetDataToSPSS("",casedata)
spssdictionary.EndDataStep() } }
END PROGRAM.
set printback on.
```

Step 4: Final factor analysis with the 22 items of the SSAW scale.

```
FACTOR
  /VARIABLES SWSRS_prä_aa_2 SWSRS_prä_aa_6 SWSRS_prä_sm_1 SWSRS_prä_sm_3
SWSRS_prä_sm_7 SWSRS_prä_sm_8 SWSRS_akt_sk_1 SWSRS_akt_sk_7
SWSRS_akt_sk_15 SWSRS_akt_sk_17 SWSRS_akt_sk_19 SWSRS_akt_sb_2
SWSRS_akt_sb_4 SWSRS_akt_sb_7 SWSRS_akt_sb_9 SWSRS_post_sb_1
SWSRS_post_sb_3 SWSRS_post_sb_4 SWSRS_post_sr_2 SWSRS_post_sr_4
SWSRS_post_sr_6 SWSRS_post_sr_7
  /MISSING LISTWISE
  /ANALYSIS SWSRS_prä_aa_2 SWSRS_prä_aa_6 SWSRS_prä_sm_1 SWSRS_prä_sm_3
SWSRS_prä_sm_7 SWSRS_prä_sm_8 SWSRS_akt_sk_1 SWSRS_akt_sk_7
SWSRS_akt_sk_15 SWSRS_akt_sk_17 SWSRS_akt_sk_19 SWSRS_akt_sb_2
SWSRS_akt_sb_4 SWSRS_akt_sb_7 SWSRS_akt_sb_9 SWSRS_post_sb_1
SWSRS_post_sb_3 SWSRS_post_sb_4 SWSRS_post_sr_2 SWSRS_post_sr_4
SWSRS_post_sr_6 SWSRS_post_sr_7
  /PRINT UNIVARIATE INITIAL CORRELATION KMO REPR AIC EXTRACTION
ROTATION
  /PLOT EIGEN
  /CRITERIA FACTORS(1) ITERATE(100)
  /EXTRACTION PAF
  /CRITERIA ITERATE(100)
  /ROTATION PROMAX(4)
  /METHOD=CORRELATION.
```

SPSS-Syntax for Item Analysis and Correlations Between the SSAW scale, General Self-Efficacy, and Self-Regulation Scale

```
*Skalenbildung SWSRS*

COMPUTE SWSRS_22=MEAN(SWSRS_prä_aa_2, SWSRS_prä_aa_6, SWSRS_prä_sm_1,
SWSRS_prä_sm_3, SWSRS_prä_sm_7, SWSRS_prä_sm_8, SWSRS_akt_sk_1,
SWSRS_akt_sk_7, SWSRS_akt_sk_15, SWSRS_akt_sk_17, SWSRS_akt_sk_19,
SWSRS_akt_sb_2, SWSRS_akt_sb_4, SWSRS_akt_sb_7, SWSRS_akt_sb_9,
SWSRS_post_sb_1, SWSRS_post_sb_3, SWSRS_post_sb_4, SWSRS_post_sr_2,
SWSRS_post_sr_4, SWSRS_post_sr_6, SWSRS_post_sr_7).
EXECUTE.

*Reliabilität SWSRS*

RELIABILITY
  /VARIABLES=SWSRS_prä_aa_2, SWSRS_prä_aa_6, SWSRS_prä_sm_1,
SWSRS_prä_sm_3, SWSRS_prä_sm_7, SWSRS_prä_sm_8, SWSRS_akt_sk_1,
SWSRS_akt_sk_7, SWSRS_akt_sk_15, SWSRS_akt_sk_17, SWSRS_akt_sk_19,
SWSRS_akt_sb_2, SWSRS_akt_sb_4, SWSRS_akt_sb_7, SWSRS_akt_sb_9,
SWSRS_post_sb_1, SWSRS_post_sb_3, SWSRS_post_sb_4, SWSRS_post_sr_2,
SWSRS_post_sr_4, SWSRS_post_sr_6, SWSRS_post_sr_7
  /SCALE('SWSRS_22') ALL
  /MODEL=ALPHA
  /STATISTICS=DESCRIPTIVE SCALE
  /SUMMARY=TOTAL.
```

```
*Skalenbildung SWSRS_PRÄ*

COMPUTE SWSRS_PRÄ_22=MEAN(SWSRS_prä_aa_2, SWSRS_prä_aa_6,
SWSRS_prä_sm_1, SWSRS_prä_sm_3, SWSRS_prä_sm_7, SWSRS_prä_sm_8).
EXECUTE.

*Reliabilität SWSRS_PRÄ*

RELIABILITY
  /VARIABLES=SWSRS_prä_aa_2, SWSRS_prä_aa_6, SWSRS_prä_sm_1,
SWSRS_prä_sm_3, SWSRS_prä_sm_7, SWSRS_prä_sm_8
  /SCALE('SWSRS_PRÄ_22') ALL
  /MODEL=ALPHA
  /STATISTICS=DESCRIPTIVE SCALE
  /SUMMARY=TOTAL.

*Skalenbildung SWSRS_AKT*

COMPUTE SWSRS_AKT_22=MEAN(SWSRS_akt_sk_1, SWSRS_akt_sk_7,
SWSRS_akt_sk_15, SWSRS_akt_sk_17, SWSRS_akt_sk_19, SWSRS_akt_sb_2,
SWSRS_akt_sb_4, SWSRS_akt_sb_7, SWSRS_akt_sb_9).
EXECUTE.

*Reliabilität SWSRS_AKT*

RELIABILITY
  /VARIABLES=SWSRS_akt_sk_1, SWSRS_akt_sk_7, SWSRS_akt_sk_15,
SWSRS_akt_sk_17, SWSRS_akt_sk_19, SWSRS_akt_sb_2, SWSRS_akt_sb_4,
SWSRS_akt_sb_7, SWSRS_akt_sb_9
  /SCALE('SWSRS_AKT_22') ALL
  /MODEL=ALPHA
  /STATISTICS=DESCRIPTIVE SCALE
  /SUMMARY=TOTAL.

*Skalenbildung SWSRS_POST*

COMPUTE SWSRS_POST_22=MEAN(SWSRS_post_sb_1, SWSRS_post_sb_3,
SWSRS_post_sb_4, SWSRS_post_sr_2, SWSRS_post_sr_4, SWSRS_post_sr_6,
SWSRS_post_sr_7).
EXECUTE.

*Reliabilität SWSRS_POST*

RELIABILITY
  /VARIABLES=SWSRS_post_sb_1, SWSRS_post_sb_3, SWSRS_post_sb_4,
SWSRS_post_sr_2, SWSRS_post_sr_4, SWSRS_post_sr_6, SWSRS_post_sr_7
  /SCALE('SWSRS_POST_22') ALL
  /MODEL=ALPHA
  /STATISTICS=DESCRIPTIVE SCALE
  /SUMMARY=TOTAL.
```

```
*Skalenbildung ASW*

COMPUTE ASW=MEAN(ASW_1, ASW_2, ASW_3, ASW_4, ASW_5, ASW_6, ASW_7,
ASW_8, ASW_9, ASW_10).
EXECUTE.

*Reliabilität ASW*

RELIABILITY
  /VARIABLES=ASW_1 ASW_2 ASW_3 ASW_4 ASW_5 ASW_6 ASW_7 ASW_8
ASW_9 ASW_10
  /SCALE('ASW') ALL
  /MODEL=ALPHA
  /STATISTICS=DESCRIPTIVE SCALE
  /SUMMARY=TOTAL.

*Skalenbildung REG*
*umkodierte Variablen*

RECODE REG_5 (1=4) (2=3) (3=2) (4=1) INTO REG_5_i.
VARIABLE LABELS  REG_5_i 'REG_5_i'.
EXECUTE.

RECODE REG_7 (1=4) (2=3) (3=2) (4=1) INTO REG_7_i.
VARIABLE LABELS  REG_7_i 'REG_7_i'.
EXECUTE.

RECODE REG_9 (1=4) (2=3) (3=2) (4=1) INTO REG_9_i.
VARIABLE LABELS  REG_9_i 'REG_9_i'.
EXECUTE.

COMPUTE REG=MEAN(REG_1, REG_2, REG_3, REG_4, REG_5_i, REG_6, REG_7_i,
REG_8, REG_9_i, REG_10).
EXECUTE.

*Reliabilität REG*

RELIABILITY
  /VARIABLES=REG_1 REG_2 REG_3 REG_4 REG_5_i REG_6 REG_7_i REG_8
REG_9_i REG_10
  /SCALE('REG') ALL
  /MODEL=ALPHA
  /STATISTICS=DESCRIPTIVE SCALE
  /SUMMARY=TOTAL.

*Korrelationen SWSRS_22 und ASW*

CORRELATIONS
  /VARIABLES=SWSRS_22 SWSRS_PRÄ_22 SWSRS_AKT_22 SWSRS_POST_22 ASW
  /PRINT=TWOTAIL NOSIG
  /MISSING=PAIRWISE.
```

Korrelationen SWSRS_22 und REG

```
CORRELATIONS
  /VARIABLES=SWSRS_22 SWSRS_PRÄ_22 SWSRS_AKT_22 SWSRS_POST_22 REG
  /PRINT=TWOTAIL NOSIG
  /MISSING=PAIRWISE.
```