

```

ifa <- function(rr,mm) {
# routine based on image factor analysis; does exploratory factor analysis;
# generates an unrotated common factor coefficients matrix & scree plot.
# In R (v.2.0.1ff), follow w/, say, promax( ) or varimax( ) where
# parentheses contains result$fac if 'ifa' produced object 'result'
# In Splus (v.6.2) follow w/ rotate; e.g. rotate( ), same as above, but
# second argument in rotate could be 'varimax' or 'promax' or, say, 'oblimin'
# rr is taken to be symmetric matrix of correlations or covariances;
# mm is no. of factors. Also, can use fmore.rt (rp function, below).
# NB: this is the routine that appears in my recent (2005, spring) article
# Factor analysis: Exploratory; Wiley Encyclopedia of Behavioral Statistics
# For additional functions or assistance, contact:
# rpruzek@uamail.albany.edu
#rr<-matrix(rr)
rinv <- solve(rr) #takes inverse of rr; so rr must be non-singular
sm2i <- diag(rinv)
smrt <- sqrt(sm2i)# smrt a vector here
dsmrt <- diag(smrt)
rsr <- dsmrt %*% rr %*% dsmrt
reig <- eigen(rsr, sym = T)
vlamd <- reig$va
vlamd<mm <- vlamd[1:mm]
qqm <- as.matrix(reig$ve[, 1:mm])
theta <- mean(vlamd[(mm + 1):nrow(qqm)])
dg <- sqrt(vlamd<mm - theta)
if(mm == 1)fac <- dg[1] * diag(1/smrt) %*% qqm
else fac <- diag(1/smrt) %*% qqm %*% diag(dg)
plot(1:nrow(rr), vlamd, type = "o", ylab='Eigenvalues of DRD')
abline(h = theta, lty = 3)
title("Scree plot for IFA")
fac<-round(fac,2) #sets two decimal digits in output
rownames(fac)<-list(rownames(rr)[1:nrow(rr)])[[1]]
list(vlamd = vlamd, theta = theta, fac = fac) }

fmore.rt<-function(ff){
# ff is generally a factor coefficients matrix, such as fm from csmooth.
# routine appends h2's and row complexities on right of 'orthogonal'

```

```

# factor coefficients matrix, as well as col. avg sqrs and avg h2 at bottom
# of input coef. matrix; also, to make col sums positive by reflecting
# ff is orthogonally rotated w/ varimax, whether or not it had been done.
  ff <- as.matrix(ff)
  mm <- ncol(ff)
  pp <- nrow(ff)
  if(mm == 1) {
    dv <- sign(sum(ff^3))
    fmt <- ff * dv

  }
  if(mm > 1) {
    ff <- rotate(ff)$rmat
    dv <- sign(matrix(1, 1, pp) %*% ff^3)
    dv <- as.vector(dv)
    fmt <- ff %*% diag(dv)
  }
  cpx <- round(complx(fmt), 1)
  mh2 <- apply(fmt^2, 2, mean)
  h2 <- apply(fmt^2, 1, sum)
  cssm <- c(mh2, sum(mh2), mean(cpx))
  mat <- cbind(fmt, h2, cpx)
  mat <- round(rbind(mat, cssm), 2)
  return(mat)
}

complx <-function(aa){
#function used in fmore... generates row complexities for factor coef. matrices
  comp <- (apply(aa^2, 1, sum))^2/apply(aa^4, 1, sum)
  comp }

fmore.rto <-function(ff){
# routine is like .rt version except this is for oblique (rotated) f. solution
# appends h2's and row complexities on right of factor pattern matrix,
# and also prints intercorrelation matrix for correlated factors.
  ff <- as.matrix(ff)
  mm <- ncol(ff)

```

```

pp <- nrow(ff)
if(mm == 1) {
  dv <- sign(sum(ff^3))
  fmt <- ff * dv
}
h2 <- apply(ff^2, 1, sum)
if(mm > 1) {
  ffff <- rotate(ff, "promax", orthogonal = F)
  fff <- ffff$rmat
  acor <- ffff$cor
  dv <- sign(apply(fff^3, 2, sum))
  dv <- as.vector(dv)
  aapt <- fff %*% diag(dv)
  aapt <- round(cbind(aapt, h2), 2)
  aacor <- diag(dv) %*% acor %*% diag(dv)
  return(aapt, aacor)
}
dimnames(aapt) <- list(dimnames(ff)[[1]], c(as.character(1:mm)))
acor <- round(aacor, 2)
print("Promax (oblique) pattern & primary factor correlations")
list(aapt=aapt, acor=acor)
}

#Not needed for f.a., but relevant to regression based on factoring principles.
csmooth <- function(cc,m,n,mult=2) {
# Assumes input cc, is cov/correl matrix, of full rank, no reg. done
# m is no. common factors, n is sample size on which cc is based
# small sample corrections a la Muirhead/Pruzek used here
#see last line for outputs returned...
  p <- ncol(cc)
  dnc <- dimnames(cc)[[1]]
  dc <- diag(cc)
  # will be vec-identity if cc == R at outset
  dsd <- sqrt(dc)
  # s.d.'s of init. variables
  rr <- diag(1/dsd) %*% cc %*%
  diag(1/dsd)

```

```

#gives correl. matrix
rinv <- solve(rr)
#assumes rr non-singular
sm2i <- diag(rinv)
a1 <- (n - p - 4)/(n + 1)
a2 <- (a1 * (p - 1))/(n - p - 2)
smc <- 1 - (1/sm2i)
ratt <- smc/(1 - smc)
dsi <- sqrt(a1 * ratt + 1 - a2)
      # ds <- diag(1/dsi)
dsi <- ifelse(dsi < 1, 1, dsi)
rsr <- diag(dsi) %*% rr %*% diag(dsi)
reng <- svd(rsr, nv = 0)
rts <- reng$d
# roots $d = eigenvalues of rescaled correl. matrix
qci <- reng$u[, 1:m]
# vectors $u from 'right' side
vlam <- rts[1:m]
vlapm <- rts[(m + 1):p]
rm <- (sum(vlapm)^2/(sum(vlapm^2)))
gam <- (p * (1 + rm) - 2)/(p - rm)
# NB NO 'm' in denom. here
dgm <- sqrt(vlam - mean(vlapm))
if(m == 1) {
  dgm <- sqrt(vlam - mean(
    vlapm))
  fm <- diag(dsd/dsi) %*%
    as.matrix(qci) *
    dgm
}
if(m > 1) {
  fm <- diag(dsd/dsi) %*%
    qci %*% diag(
    dgm)
}
cestm <- (fm %*% t(fm))
cuniq <- (dc - (diag(cestm)))
# cuniq a vector

```

```

cestm <- cestm + diag(cuniq)
# cestm model-based est. of c
wc <- n/(n + mult * gam)
# scalar for 'smoothed' est. of cov. mat
csm <- wc * cc + (1 - wc) * cestm
# model based 'smoother'
dimnames(fm) <- list(dnc,
  as.character(1:m))
list(fm=fm,csm=csm,dsi=dsi,gam= gam,rts= rts,wc=wc)
}

fmore.rto <- function(ff){
# routine generates promax 'oblique' rotation; also adds h2 as final column
ff <- as.matrix(ff)
mm <- ncol(ff)
pp <- nrow(ff)
if(mm == 1) {
  dv <- sign(sum(ff^3))
  fmt <- ff * dv
}
if(mm==1)h2<-fmt*fmt
h2 <- apply(ff^2, 1, sum)
if(mm > 1) {
  ffff <- rotate(ff, "promax", orthogonal = F)
  fff <- ffff$rmat
  acor <- ffff$cor
  dv <- sign(apply(fff^3, 2, sum))
  dv <- as.vector(dv)
  aapt <- fff %*% diag(dv)
  aapt <- round(cbind(aapt, h2), 2)
  aacor <- diag(dv) %*% acor %*% diag(dv)
  return(aapt, aacor)
}
if(mm > 1)dimnames(aapt) <- list(dimnames(ff)[[1]], c(as.character(1:mm)))
if (mm>1) acor <- round(aacor, 2)
if(mm==1)return(ff=cbind(fmt,h2)
list(aapat=aapat, acor=acor))
}

```

```

fmore <-function(ff){
#routine does no rotation, it merely appends h2's
#and adds row complexities on right of 'orthogonal'
#factor coefficients matrix ff (possibly rotated); also adds
#final row of proportions of variance for cols & mean h2's & complexities
  ff <- as.matrix(ff)
  cpx <- round(complx(ff), 1)
  mh2 <- apply(ff^2, 2, mean)
  h2 <- apply(ff^2, 1, sum)
  cssm <- c(mh2, sum(mh2), mean(cpx))
  mat <- round(cbind(ff, h2), 2)
  mat <- cbind(mat, cpx)
  mat <- round(rbind(mat, cssm), 2)
  mat
}

```

```

covnorm <- function(cov){
  # rescales cov to have ones in diagonal; cov → rc
  cc<-as.matrix(cov) #in case input is a data frame
  dn<-dimnames(cov)
  dg <- diag(cc)
  dg <- sqrt(1/dg)
  rc <- (diag(dg) %*% cc %*% diag(dg))
  dimnames(rc)<-dn
  return(rc=rnd3(rc)) }

```

```

rnd3 <- function(x)round(x,3)

```

```

mnormp <- function(cc, n, rd = 5){
#generates an n x p matrix whose covariance matrix EXACTLY
#reproduces the input correlation matrix cc. Rounded to 5
#decimals, default, but w/ control over this aspect.
  pp <- nrow(cc) #assume cc is correlat. matrix...
#NB: mvpnorm takes off from mvrnorm in V/R [MASS] library
  mvpnorm <- function(Sigma) {
#generates eigen-based factors of Sigma...only

```

```

    p <- nrow(Sigma)
    if(!all.equal(dim(Sigma), c(p, p)))
      stop("incompatible arguments")
    eS <- eigen(Sigma, sym = T)
    if(!all(eS$values >= 0))
      stop("Sigma is not positive definite")
    ef <- eS$vectors %*% diag(sqrt(eS$values))
    ef
  }
  mu <- c(rep(0, pp))
  xnu <- svd(scale(mnorm(diag(pp), n)))$u)
  efc <- mvpnorm(cc)
  xxnp <- t(efc %*% t(xnu))
  return(xxnp) }

```

```

fscrs <- function(xx, m)
{#routine gives regression-based, csmooth-generated factor score #estimates for input data matrix xx [of
order n x p], for m varimax #rotated factors; SO csmooth function must be loaded for this to work
  xs <- scale(xx)
  r <- cor(xx, na.method = "ava")
  csx <- csmooth(r, m, nrow(xx))
  ft <- rotate(csx$f)$rm
  fss <- xs %*% solve(csx$c) %*% ft
  apx <- fss %*% t(ft)
  apx <- round(apx, 2)
  fss <- round(fss, 2)
  ft <- round(ft, 2)
print("fss provides csmooth-based scores for varimax rotated factors")
  return(apx, ft, fss)
}

```

```

allr.wts
function(r){
# r assumed to be correlation matrix of full column rank
# generates all sqrd multiple correlations; see rr2 below.
# Also L.S. regression wts (betas) for predicting each variable from the p-1 others

```

```

# also, all p-2 order partial correlations, cf. matrix rp2 below.
p <- ncol(r)
xri <- solve(r)
s2i <- diag(xri)
# this is diagonal, usually called 'S' ;
ds <- diag(sqrt(1./s2i))
b <- diag(p) - xri %*% diag(1./s2i)
# columns of b are reg. wts for each variable in relation to all others.
# zps <- -zs %*% rp2 %*% ds      # n x p matrix 'Z S^-1 R((p-2)ord partls
rp2 <- diag(p) - ds %*% xri %*% ds
rp2 <- round(rp2, 3)
b.ls <- round(b, 3)
rr2 <- round(diag(round(1. - ds^2., 3.)), 3.)
#print("sq'd mult. correlations, betas [in cols], & p-2 order partials")
list(rr2=rr2, b.ls=b.ls, rp2=rp2) }

```