Online supplement to "Ordinal Factor Analysis of Graded-Preference Questionnaire Data" by Anna Brown and Alberto Maydeu-Olivares

R Code for Computing Item Information, Standard Errors and Reliability

```
info.matrix.pc <- function (model, pair, point)
{ ##########################################################################
  # computes Fisher information matrix for paired comparisons or graded block
  # designs with intransitivities (second-order models) at a given point
  ####################################
  # model: list containing model parameters in factor loading/threshold
  # parameterization; i.e. parameters given by Mplus with PARAMETERIZATION=THETA
  # the following matrices should be included in the list:
  # A:  p x bn matrix of utility contrasts for each pair
  #     (p - number of pairs; b - number of blocks, n - number of items in each block)
  # LAMBDA: bn x d matrix of utility factor loadings (d - number of factors)
  # TAU: p x c matrix of thresholds (c - number of thresholds; for binary c=1)
  # PSI: bn vector of utility residual variances
  # OMEGA: p vector of intransitivity error variances; these may be all equal
  # PHI: d x d trait correlation matrix (traits must be standardized)
  ####################################
  # pair:  number between 1 and p, refers to a row number in A
  ####################################
  # point: bn vector of utility values
  ##########################################################################

  # create a (d+bn) vector from pair-specific rows of matrices A*LAMBDA and A
  pair.loadings <- cbind(model$A[pair,]%*%model$LAMBDA, t(model$A[pair,]))
  # fill a (d+bn)x(d+bn) Fisher info matrix with multipliers; same for all categories
  info.matrix <- t(pair.loadings)%*%pair.loadings/model$OMEGA[pair]

  # compute category-specific parts of info; the same for all info matrix cells
  info <- 0
  # argument in normal ogive and density functions, contrast between relevant utilities
  z <- crossprod(model$A[pair,],point)
  # normal density and normal ogive for category c=0
    density.c_1 <- 0
    ogive.c_1 <- 1

  # iterate over c thresholds (c=1 for binary case)
   for (c in 1:sum(!is.na(model$TAU[pair,])))
       { z.c <- (z - model$TAU[pair,c])/sqrt(model$OMEGA[pair])
         density.c <- dnorm(z.c)
         ogive.c <- pnorm(z.c)
         # if denominator is not 0 and the category adds info
         if (ogive.c != ogive.c_1)
         { info <- info + (density.c_1-density.c)^2/(ogive.c_1-ogive.c)
         }
         # keep function results for next iteration
         density.c_1 <- density.c
         ogive.c_1 <- ogive.c
       }
  # now the last response category
  density.c <- 0
  ogive.c <- 0
  info <- info + (density.c_1-density.c)^2/(ogive.c_1-ogive.c)

  # now multiply the Fisher matrix entries by the sum of category functions (scalar)
  info.matrix <- as.numeric(info)*info.matrix
  return(info.matrix)
}
```

```
info.matrix.ranking <- function (model, pair, point)
{ ######################################################
  # computing Fisher information matrix for a pair in ranking designs, or graded
  # block designs without intransitivities (first-order model) at a given point
  ####################################
  # model: list containing model parameters in factor loading/threshold
  # parameterization; i.e. parameters given by Mplus with PARAMETERIZATION=THETA
  # the following matrices should be included in the list:
  # A: p x bn matrix of matrix of utility contrasts for each pair
  #    (p - number of pairs; b - number of blocks, n - number of items in each block)
  # LAMBDA: bn x d matrix of utility factor loadings (d - number of factors)
  # TAU: p x c matrix of thresholds (c - number of thresholds; for binary c=1)
  # PSI: bn vector of utility residual variances
  # PHI: d x d trait correlation matrix (traits must be standardized)
  ######################################
  # pair:  number between 1 and p, refers to a row number in A
  ######################################
  # point: d vector of trait values, a point in d-dimensional space
  ###########################################################################

  # create a (d) vector from pair-specific row of matrix A*LAMBDA
  pair.loadings <- model$A[pair,]%*%model$LAMBDA
  # fill a (d)x(d) Fisher info matrix with multipliers; same for all categories
  info.matrix <- t(pair.loadings)%*%pair.loadings/
                 as.numeric(crossprod(model$A[pair,]^2, model$PSI))

  # now compute category-related parts of info; the same for all info matrix cells
  info <- 0
  # arguments in normal ogive and density functions
  z <- crossprod(t(pair.loadings),point)
  # normal density and normal ogive for category c=0
  density.c_1 <- 0
  ogive.c_1 <- 1

  # iterate over c thresholds (c=1 for binary case)
   for (c in 1:sum(!is.na(model$TAU[pair,])))
       { z.c <- (z - model$TAU[pair,c])/sqrt(crossprod(model$A[pair,]^2,model$PSI))
         density.c <- dnorm(z.c)
         ogive.c <- pnorm(z.c)
         # if denominator is not 0 and the category adds info
         if (ogive.c != ogive.c_1)
         {  info <- info + (density.c_1-density.c)^2/(ogive.c_1-ogive.c)
         }
         # keep function results for next iteration
         density.c_1 <- density.c
         ogive.c_1 <- ogive.c
       }
  # the last response category
  density.c <- 0
  ogive.c <- 0
  info <- info + (density.c_1-density.c)^2/(ogive.c_1-ogive.c)

  # now multiply the Fisher matrix entries by sum of category functions (scalar)
  info.matrix <- as.numeric(info)*info.matrix
  return(info.matrix)
}
```

```
#####################################################################
########  EMPIRICAL EXAMPLE:  GRADED BLOCKS of size n=3     #########
########           uses function info.matrix.pc()           #########
#####################################################################
#### read matrices of model parameters estimated in Mplus and saved as text files
contrasts <- as.matrix(read.table("GradedBig5A.txt"))
loadings <- as.matrix(read.table("GradedBig5LAMBDA.txt"))
thresholds <- as.matrix(read.table("GradedBig5TAU.txt"))
correlations <- as.matrix(read.table("GradedBig5PHI.txt"))
errors <- as.matrix(read.table("GradedBig5PSI.txt"))
intrans <- as.matrix(read.table("GradedBig5OMEGA.txt"))

#### create a list with all model parameters
ModelGP <- list(A=contrasts, LAMBDA=loadings, TAU=thresholds, PHI=correlations,
                PSI=errors, OMEGA=intrans)

#### read MAP estimated factor scores saved by MPLUS
MAPGP <- as.matrix(read.table("GradedBig5MAPscores.txt",header=TRUE))

#### Compute SE for every MAP score in the sample

# to store squared standard error for each person and each trait
SqErrGP <- matrix(nrow=nrow(MAPGP),ncol=ncol(ModelGP$LAMBDA))
# prepare Fisher test information matrix
test.info.matrix <- matrix(0L,nrow=ncol(ModelGP$LAMBDA)+ncol(ModelGP$A),
               ncol=ncol(ModelGP$LAMBDA)+ncol(ModelGP$A))

for (j in 1:nrow(MAPGP))  ## iterate through all persons in the sample
    { for (i in 1:60) ## iterate through all pairs in the test
        {test.info.matrix <- test.info.matrix +
              info.matrix.pc(ModelGP, pair=i, point=MAPGP[j,2:61])
         }
      # posterior for traits, inverse of the PHI matrix
      test.info.matrix[1:5,1:5] <- test.info.matrix[1:5,1:5] + solve(ModelGP$PHI)
      # posterior for errors, inverse of the PSI matrix
      test.info.matrix[6:65,6:65] <- test.info.matrix[6:65,6:65] +
                                     solve(diag(as.numeric(ModelGP$PSI)))
      SqErrGP[j,] <- diag(solve(test.info.matrix)[1:5,1:5])
      # reset the Fisher info matrix to 0 for the next person
      test.info.matrix[] <- 0L
     }

# Examine average squared errors
colMeans(SqErrGP)

#### Compute empirical reliability
library(matrixStats)
colSds(MAPGP[,62:66])^2/(colMeans(SqErrGP)+ colSds(MAPGP[,62:66])^2)
```

```
#####################################################################
########  EMPIRICAL EXAMPLE:  LIKERT SCALE (block size n=1) #########
########        uses function info.matrix.ranking()         #########
#####################################################################

#### read matrices of model parameters estimated in Mplus and saved as text files
loadings <- as.matrix(read.table("LikertBig5LAMBDA.txt"))
thresholds <- as.matrix(read.table("LikertBig5TAU.txt"))
correlations <- as.matrix(read.table("LikertBig5PHI.txt"))

ModelLikert <- list(A=diag(60), LAMBDA=loadings, TAU=thresholds, PHI=correlations,
PSI=as.matrix(rep(1,60)))

#### read MAP estimated factor scores saved by MPLUS
MAPLikert <- as.matrix(read.table("LikertBig5MAPscores.txt",header=TRUE))

#### Compute SE for every MAP score in the sample
# to store squared standard error for each person and each trait
SqErrLikert <- matrix(nrow=nrow(MAPLikert),ncol=ncol(ModelLikert$LAMBDA))
# prepare Fisher test information matrix
test.info.matrix <- matrix(0L,nrow=nrow(ModelLikert$PHI),ncol=ncol(ModelLikert$PHI))

for (j in 1:nrow(MAPLikert)) ## iterate through all persons in the sample
    { for (i in 1:60) ## iterate through all items in the test
         {test.info.matrix <- test.info.matrix + info.matrix.ranking(ModelLikert,
           pair=i, point=MAPLikert[j,2:6])
          }
      # posterior, add the inverse of the PHI matrix
      test.info.matrix <- test.info.matrix + solve(ModelLikert$PHI)
      SqErrLikert[j,] <- diag(solve(test.info.matrix))
      # reset the Fisher info matrix to 0 for the next person
      test.info.matrix[] <- 0L
    }
#### Examine average squared errors
colMeans(SqErrLikert)

#### Compute empirical reliability
library(matrixStats)
colSds(MAPLikert[,2:6])^2/(colMeans(SqErrLikert)+colSds(MAPLikert[,2:6])^2)
```