```r
#~~~~~~~~~~~~~~~#
# INSTRUCTIONS #
#~~~~~~~~~~~~~~~#
# The following code can be used for calculating RMSEA_u and CFI_u
# using approaches (1.1), (1.2), and (2). It can be used for uni- and
# multidimensional CFA models. For a more flexible use, the Ffun-
# function (lines 24 to 31) needs to be adapted.
# An example data is provided with this article, which contains four
# metric variables. Its location needs to be specified in line 37.
# The corresponding parameter estimates (standardized ML loadings)
# are entered in line 54. Estimate vectors of multidimensional CFA
# models first contain all loadings and then the factor correlations.

library(matrixcalc)
library(numDeriv)

#~~~~~~~~~~~~~~~~~#
# basic functions #
#~~~~~~~~~~~~~~~~~#
#matrix trace
tr <- function(x) sum(diag(x))

#model implied correlations based on loadings and factor correlations
Ffun <- function(x) {
  LAM <- diag(q) %x% rep(1,p/q)
  LAM[LAM==1] <- x[(q2+1):(q2+p)]
  PHI <- diag(q)/2
  PHI[lower.tri(PHI)] <- x[1:q2]
  PHI <- PHI + t(PHI)
  (LAM %*% PHI %*% t(LAM))[lower.tri(diag(p))]
}

#~~~~~~#
# data #
#~~~~~~#
#!!! specify data location !!!
X <- read.table('...\\example_data.txt')

#settings
n  <- nrow(X)
p  <- ncol(X)
p2 <- (p*(p-1))/2
q  <- 1
q2 <- (q*(q-1))/2
df <- p2 - p - q2

#sample correlations
S0 <- cor(X)

#~~~~~~~~~~~~~~~~#
# ML CFA-results #
#~~~~~~~~~~~~~~~~#
#std. loading estimates
est <- c(0.3177388, -0.3547450,  0.2909282, -0.4468409)

#model implied correlations
s1 <- Ffun(est)
S. <- diag(p)/2
S.[lower.tri(S.)] <- s1
S1 <- S. + t(S.)

#ML test statistic
T_ML <- n * (log(det(S1)) + tr(S0 %*% solve(S1)) - log(det(S0)) - p)

#ML test statistic (base model)
T_B_ML <- n * (tr(S0 %*% diag(p)) - log(det(S0)) - p)

#~~~~~~~~~~~~~~~~~#
# ULS discrepancy #
#~~~~~~~~~~~~~~~~~#
#correlation residuals
e     <- (S0-S1)[lower.tri(S0)]

```

```r
#model discrepancy
F_uls <- sum(e^2)

#base model discrepancy
F_B_u <- sum(S0[lower.tri(S0)]^2)

#~~~~~~~~~~~~~~~~~~~~#
# calculate matrices #
#~~~~~~~~~~~~~~~~~~~~#
#I
I   <- diag((p*(p-1))/2)

#Gamma
X2 <- X^2
C  <- combn(p,2)
for(i in 1:ncol(C)) X2 <- cbind(X2, X[,C[1,i]]*X[,C[2,i]])
GAM <- var(X2)

#W
Dp  <- duplication.matrix(p)
D.  <- svd.inverse(Dp)
cr  <- which(rowSums(D.^2) != 1)
W.  <- 2 * D. %*% (S0 %x% S0) %*% t(D.)
W   <- solve(W.)[cr,cr]

#Delta
DEL <- jacobian(Ffun, est)

#~~~~~~~~~~~~~~~~#
# approach (1.1) #
#~~~~~~~~~~~~~~~~#
U_1.1  <- I - DEL %*% solve(t(DEL) %*% DEL) %*% t(DEL)
V_1.1  <- (U_1.1 %*% GAM[cr,cr])/n
tr_1.1 <- tr(V_1.1)
s2_1.1 <- 2 * tr(V_1.1^2) + 4 * t(e) %*% V_1.1 %*% e

#corrected RMSEA_u estimate
k_1.1   <- 1 - (s2_1.1) / (4*F_uls^2)
R_u_1.1 <- 1/k_1.1 * sqrt(max(c(F_uls - tr_1.1, 0))/df)

#RMSEA_u standard error
SE_1.1  <- sqrt(s2_1.1/(k_1.1^2*4*F_uls))[1]

#CFI_u
C_u_1.1 <- 1 - (F_uls-tr_1.1) / (F_B_u-tr_1.1)

#output
round(
c(RMSEA_u_1.1 = R_u_1.1,
  low.CI      = max(0, R_u_1.1-qnorm(.95)*SE_1.1),
  upp.CI      = R_u_1.1+qnorm(.95)*SE_1.1,
  CFI_u_1.1   = C_u_1.1), 3)

#~~~~~~~~~~~~~~~~#
# approach (1.2) #
#~~~~~~~~~~~~~~~~#
U_1.2  <- I - DEL %*% solve(t(DEL) %*% W %*% DEL) %*% t(DEL) %*% W
V_1.2  <- 1/n * (U_1.2) %*% GAM[cr,cr] %*% t(U_1.2)
tr_1.2 <- tr(V_1.2)
s2_1.2 <- 2 * tr(V_1.2^2) + 4 * t(e) %*% V_1.2 %*% e

#corrected RMSEA_u estimate
k_1.2   <- 1 - (s2_1.2) / (4*F_uls^2)
R_u_1.2 <- 1/k_1.2 * sqrt(max(c(F_uls - tr_1.2, 0))/df)

#RMSEA_u standard error
SE_1.2  <- sqrt(s2_1.2/(k_1.2^2*4*df*F_uls))[1]

#CFI_u
C_u_1.2 <- 1 - (F_uls-tr_1.2) / (F_B_u-tr_1.2)

#output
round(
```

```r
c(RMSEA_u_1.2 = R_u_1.2,
  low.CI      = max(0, R_u_1.2-qnorm(.95)*SE_1.2),
  upp.CI      = R_u_1.2+qnorm(.95)*SE_1.2,
  CFI_u_1.2   = C_u_1.2), 3)

#~~~~~~~~~~~~~~~#
# approach (2) #
#~~~~~~~~~~~~~~~#
s2_2  <- s2_1.2
#model ADR
ADR   <- (T_ML - df) / T_ML
#base model ADR (for CFI_u)
ADR_B <- (T_B_ML - p2) / T_B_ML

#corrected RMSEA_u estimate
k_2   <- 1 - (s2_2)/(8*F_uls^2)
R_u_2 <- 1/k_2 * sqrt(max(c(ADR * F_uls, 0))/df)

#RMSEA_u standard error
SE_2  <- sqrt((s2_2*ADR)/(k_2^2*4*df*F_uls))[1]

#CFI_u
C_u_2 <- 1 - (ADR * F_uls) / (ADR_B * F_B_u)

#output
round(
c(RMSEA_u_2 = R_u_2,
  low.CI    = max(0, R_u_2-qnorm(.95)*SE_2),
  upp.CI    = R_u_2+qnorm(.95)*SE_2,
  CFI_u_2   = C_u_2),3)

```