```r
# ------------------------------------------------------------------------------------------------------------------------
# Program: twoACEvj.R
#  Author: Hermine Maes
#    Date: 10 22 2018
#
# Twin Bivariate ACE model to estimate causes of variation across multiple groups
# Matrix style model - Raw data - Continuous data
# -------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|

# Load Libraries & Options
rm(list=ls())
library(OpenMx)
library(psych); library(polycor)
source("miFunctions.R")

# Create Output
filename    <- "twoACEvj"
sink(paste(filename,".Ro",sep=""), append=FALSE, split=TRUE)

# ------------------------------------------------------------------------------------------------------------------------
# PREPARE DATA

# Load Data
data(twinData)
dim(twinData)
describe(twinData[,1:12], skew=F)
twinData[,'ht1'] <- twinData[,'ht1']*10
twinData[,'ht2'] <- twinData[,'ht2']*10
twinData[,'wt1'] <- twinData[,'wt1']/10
twinData[,'wt2'] <- twinData[,'wt2']/10

# Select Continuous Variables
varsc     <- c('ht')                    # list of continuous variables names
nvc       <- 1                          # number of continuous variables
ntvc      <- nvc*2                      # number of total continuous variables
conVars   <- paste(varsc,c(rep(1,nvc),rep(2,nvc)),sep="")

# Select Ordinal Variables
nth       <- 3                          # number of thresholds
varso     <- c('wt')                    # list of ordinal variables names
nvo       <- 1                          # number of ordinal variables
ntvo      <- nvo*2                      # number of total ordinal variables
ordVars   <- paste(varso,c(rep(1,nvo),rep(2,nvo)),sep="")
ordData   <- twinData
wtquant   <- quantile(ordData[,c('wt1','wt2')],(0:(nth+1))/(nth+1),na.rm=TRUE)
 for (i in c('wt1','wt2')) { ordData[,i] <- cut(ordData[,i], breaks=wtquant, labels=c(0:nth)) }

# Select Variables for Analysis
vars      <- c('ht','wt')               # list of variables names
nv        <- nvc+nvo                    # number of variables
ntv       <- nv*2                       # number of total variables
oc        <- c(0,1)                     # 0 for continuous, 1 for ordinal variables
selVars   <- paste(vars,c(rep(1,nv),rep(2,nv)),sep="")

# Select Data for Analysis
mzData    <- subset(ordData, zyg==1, selVars)
dzData    <- subset(ordData, zyg==3, selVars)
mzDataF   <- cbind(mzData[,conVars],mxFactor( x=mzData[,ordVars], levels=c(0:nth)) )
dzDataF   <- cbind(dzData[,conVars],mxFactor( x=dzData[,ordVars], levels=c(0:nth)) )

# Generate Descriptive Statistics
apply(mzData,2,myMean)
apply(dzData,2,myMean)
sapply(mzData[,ordVars],table)
sapply(dzData[,ordVars],table)
hetcor(mzData)$cor
hetcor(dzData)$cor

# Set Starting Values
frMV      <- c(TRUE,FALSE)              # free status for variables
svMe      <- c(15,0)                    # start value for means
svPa      <- .2                         # start value for path coefficient
svPe      <- .5                         # start value for path coefficient for e
svLTh     <- -.5                        # start value for first threshold
svITh     <- 1                          # start value for increments
svTh      <- matrix(rep(c(svLTh,(rep(svITh,nth-1)))),nrow=nth,ncol=nv)     # start value for thresholds
lbTh      <- matrix(rep(c(-3,(rep(0.001,nth-1))),nv),nrow=nth,ncol=nv)     # lower bounds for thresholds

# ------------------------------------------------------------------------------------------------------------------------
# PREPARE MODEL

# Create Algebra for expected Mean & Threshold Matrices
```

```
meanG     <- mxMatrix( type="Full", nrow=1, ncol=ntv, free=frMV, values=svMe, labels=labVars("mean",vars), name="meanG" )
thinG     <- mxMatrix( type="Full", nrow=nth, ncol=ntvo, free=TRUE, values=svTh, lbound=lbTh, labels=labTh("th",vars,nth),
 name="thinG" )
inc       <- mxMatrix( type="Lower", nrow=nth, ncol=nth, free=FALSE, values=1, name="inc" )
threG     <- mxAlgebra( expression= inc %*% thinG, name="threG" )

# Create Matrices for Variance Components
covA      <- mxMatrix( type="Symm", nrow=nv, ncol=nv, free=TRUE, values=valDiag(svPa,nv), label=labLower("VA",nv), name="VA" )
covC      <- mxMatrix( type="Symm", nrow=nv, ncol=nv, free=TRUE, values=valDiag(svPa,nv), label=labLower("VC",nv), name="VC" )
covE      <- mxMatrix( type="Symm", nrow=nv, ncol=nv, free=TRUE, values=valDiag(svPa,nv), label=labLower("VE",nv), name="VE" )

# Create Algebra for expected Variance/Covariance Matrices in MZ & DZ twins
covP      <- mxAlgebra( expression= VA+VC+VE, name="V" )
covMZ     <- mxAlgebra( expression= VA+VC, name="cMZ" )
covDZ     <- mxAlgebra( expression= 0.5%x%VA+ VC, name="cDZ" )
expCovMZ  <- mxAlgebra( expression= rbind( cbind(V, cMZ), cbind(t(cMZ), V)), name="expCovMZ" )
expCovDZ  <- mxAlgebra( expression= rbind( cbind(V, cDZ), cbind(t(cDZ), V)), name="expCovDZ" )

# Constrain Variance of Binary Variables
matUnv    <- mxMatrix( type="Unit", nrow=nvo, ncol=1, name="Unv1" )
matOc     <- mxMatrix( type="Full", nrow=1, ncol=nv, free=FALSE, values=oc, name="Oc" )
var1      <- mxConstraint( expression=diag2vec(Oc%&%V)==Unv1, name="Var1" )

# Create Data Objects for Multiple Groups
dataMZ    <- mxData( observed=mzDataF, type="raw" )
dataDZ    <- mxData( observed=dzDataF, type="raw" )

# Create Expectation Objects for Multiple Groups
expMZ     <- mxExpectationNormal( covariance="expCovMZ", means="meanG", dimnames=selVars, thresholds="threG",
 threshnames=ordVars )
expDZ     <- mxExpectationNormal( covariance="expCovDZ", means="meanG", dimnames=selVars, thresholds="threG",
 threshnames=ordVars )
funML     <- mxFitFunctionML()

# Create Model Objects for Multiple Groups
pars      <- list(meanG, thinG, inc, threG, matUnv, matOc, covA, covC, covE, covP )
modelMZ   <- mxModel( pars, covMZ, expCovMZ, dataMZ, expMZ, funML, name="MZ" )
modelDZ   <- mxModel( pars, covDZ, expCovDZ, dataDZ, expDZ, funML, name="DZ" )
multi     <- mxFitFunctionMultigroup( c("MZ","DZ") )

# Create Algebra for Standardization
matI      <- mxMatrix( type="Iden", nrow=nv, ncol=nv, name="I" )
invSD     <- mxAlgebra( expression=solve(sqrt(I*V)), name="iSD" )

# Calculate genetic and environmental correlations
corA      <- mxAlgebra( expression=solve(sqrt(I*VA))%&%VA, name ="rA" ) #cov2cor()
corC      <- mxAlgebra( expression=solve(sqrt(I*VC))%&%VC, name ="rC" )
corE      <- mxAlgebra( expression=solve(sqrt(I*VE))%&%VE, name ="rE" )

# Create Algebra for Unstandardized and Standardized Variance Components
rowUS     <- rep('US',nv)
colUS     <- rep(c('VA','VC','VE','SA','SC','SE'),each=nv)
estUS     <- mxAlgebra( expression=cbind(VA,VC,VE,VA/V,VC/V,VE/V), name="US", dimnames=list(rowUS,colUS) )

# Create Confidence Interval Objects
odd       <- seq(1+3*nv,2*3*nv,nv)
even      <- seq(2+3*nv,2*3*nv,nv)
ciACE     <- mxCI( c("US[1,odd]","US[2,odd]","US[2,even]") )

# Build Model with Confidence Intervals
calc      <- list( matI, invSD, corA, corC, corE, estUS, ciACE )
modelACE  <- mxModel( "twoACEvj", pars, var1, modelMZ, modelDZ, multi, calc )

# ------------------------------------------------------------------------------------------------------------------------
# RUN MODEL

# Run ACE Model
fitACE    <- mxRun( modelACE, intervals=T )
sumACE    <- summary( fitACE )

# Compare with Saturated Model
#mxCompare( fitSAT, fitACE )

# Print Goodness-of-fit Statistics & Parameter Estimates
fitGofs(fitACE)
fitEstCis(fitACE)

# ------------------------------------------------------------------------------------------------------------------------
# RUN SUBMODELS

# Run AE model
modelAE   <- mxModel( fitACE, name="twoAEvj" )
```

```
modelAE   <- omxSetParameters( modelAE, labels=labLower("VC",nv), free=FALSE, values=0 )
modelAE   <- omxSetParameters( modelAE, labels=labLower("VA",nv), free=TRUE, values=.6 )
fitAE     <- mxRun( modelAE, intervals=T )
fitGofs(fitAE); fitEstCis(fitAE)

# Run CE model
modelCE   <- mxModel( fitACE, name="twoCEvj" )
modelCE   <- omxSetParameters( modelCE, labels=labLower("VA",nv), free=FALSE, values=0 )
modelCE   <- omxSetParameters( modelCE, labels=labLower("VC",nv), free=TRUE, values=.6 )
fitCE     <- mxRun( modelCE, intervals=T )
fitGofs(fitCE); fitEstCis(fitCE)

# Run E model
modelE    <- mxModel( fitAE, name="twoEvj" )
modelE    <- omxSetParameters( modelE, labels=labLower("VA",nv), free=FALSE, values=0 )
fitE      <- mxRun( modelE, intervals=T )
fitGofs(fitE); fitEstCis(fitE)

# Print Comparative Fit Statistics
mxCompare( fitACE, nested <- list(fitAE, fitCE, fitE) )
#round(rbind(fitACE$US$result,fitAE$US$result,fitCE$US$result,fitE$US$result),4)

# ----------------------------------------------------------------------------------------------------------------------
sink()
save.image(paste(filename,".Ri",sep=""))
```