**Twin Research and Human Genetics**

Does Breastfeeding Behavior Run In Families? Evidence from twins, their sisters and their mothers in the Netherlands

P. Merjonen, C. V. Dolan, M. Bartels, D. I. Boomsma

SUPPLEMENTARY MATERIAL: R-SCRIPT FOR MODELING

# -----------------------------------------------------------------------

# Script: breastfeedingHeritability.R

# Author: Paivi Merjonen

# Date: 29 July 2014

#

#

#"R version 3.1.1 (2014-07-10)"

#platform "x86\_64-apple-darwin10.8.0"

# OpenMx 1.4

# Phenotype to be predicted: Breastfeeding (mother-daughter and sister-sister)

# Control variables: cohort

# Idea is to compare breastfeeding behaviors (ever bf vs. never bf) between twin-twin,

# twin-sister and mother-daughter pairs

#

# Type of data: binary data (dichotomous)

# zyg\_mono\_di: 0=MZ 1=DZ

#

#

# Sample: Female Twins and their mothers

# only first pair of twins within a family is included and

# they need to have children

# includes first twin pair, first sister and mother

#

# Aim: Estimate whether breastfeeding runs in families and whether this familiar transmission

# is genetic or environmental

# -----------------------------------------------------------------------

#######################################################

### PART 1 ###

# clear memory of the R concole

rm(list=ls())

# Set your working directory

# This is the directory where your datafile and script are stored

# Note that in R a forward slash is used, no backslash!

getwd()

setwd("~/Desktop/Kaksosdata/NTR/Breastfeeding/breastfeedingHeritability")

#setwd("H:/Documents/NTR/Breastfeeding/breastfeedingHeritability")

# Load the R package OpenMx

install.packages("psych")

library(psych)

source('http://openmx.psyc.virginia.edu/getOpenMx.R')

#install.packages("OpenMx")

library(OpenMx)

# require(foreign) ## Enables reading e.g. SPSS files directly (did not work correctly, so do not use)

# Run the source file GenEpiHelperFunctions.R

# This file contains helper functions that are not included in OpenMx

# And will be used to have a detailed look at the output

# Note that the location of this file is on the internet

#source("http://www.vipbg.vcu.edu/~vipbg/Tc24/GenEpiHelperFunctions.R")

source("http://openmxhelpers.googlecode.com/svn/trunk/GenEpiHelperFunctions.R")

# install graphical packages for to draw figures

install.packages("lattice")

library(lattice)

#install.packages("grid") # this is not available for R version 2.15.2.

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# -----------------------------------------------------------------------

# PREPARE DATA

# TWIN MODELLING

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# ANALYSES DONE WITH THIS DATA

# this is the datafile Conor used

# only cohort is used as definition variable, for other variables bivariate model is used

# missings in definition variables are -1, in phenotype variables -9, use this in OpenMX analysis

DataP2 <- read.table(file="bf\_conor\_26\_7\_2013.dat", header=TRUE, na.strings=c("-9"))

nrow(DataP2) #4570 families, 5581 individuals

summary(DataP2)

head(DataP2)

# set the breastfeeding to be discrete

# note factoring of breastfeeding changes its' values 1-2, although the levels are 0-1

# this probably turns it to opposite direction in twinmodel: estimates the probability of the first factor level, i.e. not breastfeeding

DataP2$everbfFactor1 <- mxFactor(DataP2$everbf1, levels=c(0:1), exclude=NA)

DataP2$everbfFactor2 <- mxFactor(DataP2$everbf2, levels=c(0:1), exclude=NA)

DataP2$everbfFactor10 <- mxFactor(DataP2$everbf10, levels=c(0:1), exclude=NA)

DataP2$everbfFactor41 <- mxFactor(DataP2$everbf41, levels=c(0:1), exclude=NA)

DataP2$churchFactor1 <- mxFactor(DataP2$church1, levels=c(0:2))

DataP2$churchFactor2 <- mxFactor(DataP2$church2, levels=c(0:2))

DataP2$churchFactor10 <- mxFactor(DataP2$church10, levels=c(0:2))

DataP2$churchFactor41 <- mxFactor(DataP2$church41, levels=c(0:2))

DataP2$educ3Factor1 <- mxFactor(DataP2$educ3new1, levels=c(0:2))

DataP2$educ3Factor2 <- mxFactor(DataP2$educ3new2, levels=c(0:2))

DataP2$educ3Factor10 <- mxFactor(DataP2$educ3new10, levels=c(0:2))

DataP2$educ3Factor41 <- mxFactor(DataP2$educ3new41, levels=c(0:2))

DataP2$coh4Factor1 <- mxFactor(DataP2$coh4new1, levels=c(0:3))

DataP2$coh4Factor2 <- mxFactor(DataP2$coh4new2, levels=c(0:3))

DataP2$coh4Factor10 <- mxFactor(DataP2$coh4new10, levels=c(0:3))

DataP2$coh4Factor41 <- mxFactor(DataP2$coh4new41, levels=c(0:3))

# Specify numbers

ndef <- 1 # number of definition variables (education, religiosity, cohort (mother), age at childbirth)

nv <- 1 # number of phenotypes (breastfeeding)

nfam <- 4 # number of family members (twin pair 1 + mother + sibling 1)

nvar <- nv\*nfam # number of variables to analyse

nthresh <- 1 # breastfeeding has 2 categories, so only 1 threshold

# Select Variables for analysis

defVars <- c("coh4new")

#phenVars <- c("everbf")

phenVars <- c("everbfFactor") # phenotype variable has to be an ordered factor, otherwise OpenMx does not run the threshold model

allVars <- paste(c(defVars,phenVars),c(rep(1,ndef+nv),rep(2,ndef+nv),rep(10,ndef+nv),rep(41,ndef+nv)),sep="")

selVars <- paste(phenVars,c(rep(1,nv),rep(2,nv),rep(10,nv),rep(41,nv)),sep="")

mz\_DataP <- subset(DataP2, (DataP2$zygosity==0), selVars)

dz\_DataP <- subset(DataP2, (DataP2$zygosity==1), selVars)

mz\_DataD <- subset(DataP2, (DataP2$zygosity==0), allVars)

dz\_DataD <- subset(DataP2, (DataP2$zygosity==1), allVars)

mz\_twins <- subset(DataP2, (DataP2$zygosity==0 & DataP2$everbf1>=0 & DataP2$everbf2>=0), selVars)

mz\_sisters <- subset(DataP2, (DataP2$zygosity==0 & (DataP2$everbf1>=0 | DataP2$everbf2>=0) & DataP2$everbf10>=0), selVars)

mz\_mother\_daughter <- subset(DataP2, (DataP2$zygosity==0 & (DataP2$everbf1>=0 | DataP2$everbf2>=0 | DataP2$everbf10>=0) & DataP2$everbf41>=0), selVars)

dz\_twins <- subset(DataP2, (DataP2$zygosity==1 & DataP2$everbf1>=0 & DataP2$everbf2>=0), selVars)

dz\_sisters <- subset(DataP2, (DataP2$zygosity==1 & (DataP2$everbf1>=0 | DataP2$everbf2>=0) & DataP2$everbf10>=0), selVars)

dz\_mother\_daughter <- subset(DataP2, (DataP2$zygosity==1 & (DataP2$everbf1>=0 | DataP2$everbf2>=0 | DataP2$everbf10>=0) & DataP2$everbf41>=0), selVars)

describe(mz\_twins) # 337 full pairs of MZ twins

describe(mz\_sisters) # 62 full twin-sister pairs in MZ data

describe(mz\_mother\_daughter) # 153 full mother-daughter pairs in MZ data

describe(dz\_twins) # 129 full pairs of DZ twins

describe(dz\_sisters) # 70 full twin-sister pairs in DZ data

describe(dz\_mother\_daughter) # 260 full mother-daughter pairs in DZ data

head(mz\_DataD,20)

head(dz\_DataD,20)

summary(mz\_DataD)

summary(dz\_DataD)

summary(mz\_DataP)

summary(dz\_DataP)

# Check descriptives of MZ data (just to double-check results)

describe(mz\_DataD)

describe(mz\_DataP)

# Check descriptives of DZ data (just to double-check results)

describe(dz\_DataD)

describe(dz\_DataP)

dataMZ <- mxData(observed = mz\_DataD, type = "raw")

dataDZ <- mxData(observed = dz\_DataD, type = "raw")

# threshold = NORMSINV(B1) in excel if B1 is percentage of breastfeeders

# returns the inverse of standard normal cumulative distribution

################################################################################

### PART 3 ###

# Set up the threshold model

# Algebra for expected Mean Matrices

# Algebra for expected Mean and Variance/Covariance Matrices in MZ & DZ twins

# Mean structure, Algebra M to store Expected means

Means <- mxMatrix(type="Zero", nrow=nv, ncol=nvar, name="expMean") # This turns the expected value to be 0 = never breastfeeding

Thresholds <- mxMatrix(type="Full", nrow=nthresh, ncol=nvar, free=TRUE, values=c(-0.82,-0.82,-0.67,-0.42),

 name="expThresh", labels=c("threshold1","threshold1","threshold10","threshold41")) #, dimnames=list('th1',selVars))

#Thresholds <- mxMatrix(type="Full", nrow=nthresh, ncol=nvar, free=TRUE, values=c(0.53,0.53,0.53,0.53),

# name="expThresh", labels=c("threshold1","threshold1","threshold1","threshold1")) #, dimnames=list('th1',selVars))

CorrelationMZ <- mxMatrix(type="Symm",nrow=nvar,ncol=nvar, free=c(F,T,T,T,T,F,T,T,T,T,F,T,T,T,T,F), labels=c("var11mz","cov12mz","cov110mz","cov141mz","cov12mz","var11mz","cov110mz","cov141mz","cov110mz","cov110mz","var11mz","cov141mz","cov141mz","cov141mz","cov141mz","var11mz"), values=c(1,0.711,0.180,0.265,0.711,1,0.180,0.265,0.180,0.180,1,0.265,0.265,0.265,0.265,1), name="expCorMZ")

CorrelationDZ <- mxMatrix(type="Symm",nrow=nvar,ncol=nvar, free=c(F,T,T,T,T,F,T,T,T,T,F,T,T,T,T,F), labels=c("var11mz","cov12dz","cov110mz","cov141mz","cov12dz","var11mz","cov110mz","cov141mz","cov110mz","cov110mz","var11mz","cov141mz","cov141mz","cov141mz","cov141mz","var11mz"), values=c(1,0.525,0.180,0.265,0.525,1,0.180,0.265,0.180,0.180,1,0.265,0.265,0.265,0.265,1), name="expCorDZ")

# I now assume that mz and dz betas are the same and different family members have similar beta

#b1mz <- mxMatrix(type="Full", nrow=nv, ncol=nvar, free=TRUE, labels=c("beta11mz","beta11mz","beta11mz","beta11mz"), values=c(-0.017,-0.017,-0.017,-0.017), name="b1mz")

b1mz <- mxMatrix(type="Full", nrow=nv, ncol=nvar, free=TRUE, labels=c("beta11mz","beta11mz","beta11mz","beta11mz"), values=c(0.1), name="b1mz")

#b1mz <- mxMatrix(type="Full", nrow=nv, ncol=nvar, free=TRUE, labels=c("beta11mz","beta11mz","beta110mz","beta141mz"), values=c(-0.2,-0.2,-0.1,-0.06), name="b1mz")

Def1 <- mxMatrix(type="Full", nrow=nv, ncol=nvar, free=FALSE,labels=(rep(c("data.coh4new1","data.coh4new2","data.coh4new10","data.coh4new41"),each=1)), name="Def1")

DefR1mz <- mxAlgebra(expression= b1mz \* Def1, name="DefR1mz")

expThresholdMZ <- mxAlgebra(expression= expThresh + DefR1mz, name="expThreshMZ")

# Algebra for expected Variance/Covariance Matrices in MZ & DZ twins

b1CovMZ <- mxMatrix(type="Symm", nrow=nvar, ncol=nvar, free=FALSE, labels=c("beta1Var11mz","beta1Cov12mz","beta1Cov110mz","beta1Cov141mz","beta1Var11mz","beta1Cov110mz","beta1Cov141mz","beta1Var11mz","beta1Cov141mz","beta1Var4141mz"), values=c(0), name="b1CovMZ")

DefR1CovMZ <- mxAlgebra(expression= b1CovMZ \* rbind(Def1,Def1,Def1,Def1), name="DefR1CovMZ")

expCovMZ <- mxAlgebra(expression= expCorMZ + DefR1CovMZ, name="expCovMZ")

expCovDZ <- mxAlgebra(expression= expCorDZ + DefR1CovMZ, name="expCovDZ")

#Algebra for expected Correlation matrices in MZ & DZ twins

ident <- mxMatrix( type="Iden", nrow=nvar, ncol=nvar, name="I")

# This is if you do zygdef groups separately

isdMZ <- mxAlgebra( solve(sqrt(I\*expCovMZ)), name="iSDmz")

corMZ <- mxAlgebra( iSDmz%\*%expCovMZ%\*%iSDmz, name="expCorMZnew")

isdDZ <- mxAlgebra( solve(sqrt(I\*expCovDZ)), name="iSDdz")

corDZ <- mxAlgebra( iSDdz%\*%expCovDZ%\*%iSDdz, name="expCorDZnew")

# Objective objects for Multiple Groups

# Objectives

MZObjective <- mxFIMLObjective(covariance="expCovMZ", means="expMean", dimnames=selVars,

 thresholds="expThreshMZ")

DZObjective <- mxFIMLObjective(covariance="expCovDZ", means="expMean", dimnames=selVars,

 thresholds="expThreshMZ")

# Combine Groups

pars <- list(Means, Thresholds,

 b1mz,

 Def1,

 DefR1mz,

 b1CovMZ,

 DefR1CovMZ,

 ident,

 expThresholdMZ,

 CorrelationMZ, CorrelationDZ,

 expCovMZ, expCovDZ,

 isdMZ, corMZ,

 isdDZ, corDZ)

# Models

MZmodel <- mxModel("MZmodel", dataMZ, pars,

 MZObjective)

DZmodel <- mxModel("DZmodel", dataDZ, pars,

 DZObjective)

# Objective

min2sumll <- mxAlgebra( expression = MZmodel.objective + DZmodel.objective, name="min2sumll" )

objective <- mxAlgebraObjective("min2sumll")

# WE ASK FOR 95% CONFIDENCE INTERVALS

ci <- mxCI(c("expThreshMZ",

 "b1mz",

 "expCovMZ","expCovDZ",

 "b1CovMZ"),

 interval = 0.95)

# Saturated model

Saturated\_Model <- mxModel(model="twinSat",

 pars,

 MZmodel, DZmodel,

 min2sumll, objective, dataMZ, dataDZ,

 ci)

## Fitting

# Run the saturated model

Saturated\_Model\_Fit <- mxRun(Saturated\_Model, intervals=T)

#Saturated\_Model\_Fit <- mxRun(Saturated\_Model, intervals=F)

#Saturated\_Model\_Parallel <- omxParallelCI(Saturated\_Model\_Fit)

# Generate Saturated\_Model Output

# -----------------------------------------------------------------------

# Ask for summary of the output

summary(Saturated\_Model\_Fit)

genEpi\_ParameterSpecifications(Saturated\_Model\_Fit) # here you can see that the next line gives values for the last cohort, not for the first, this explains the difference

genEpi\_ExpectedMeansCovariances(Saturated\_Model\_Fit)

genEpi\_TableFitStatistics(Saturated\_Model\_Fit)

#print(Saturated\_Model\_Parallel@output$confidenceIntervals)

MZcor <- Saturated\_Model\_Fit@submodels$MZ@algebras$expCorMZ@result

DZcor <- Saturated\_Model\_Fit@submodels$DZ@algebras$expCorDZ@result

MZcor

DZcor

# Prevalences if thresholds are equal between all the family members

D=-0.1744 # Mx has rounded down, while OpenMx has rounded up

T=-0.4680

cohort=c(0,1,2,3)

thresh=T+D\*cohort # gives thresholds by cohort

prev=pnorm(thresh) # prevalence of never breastfeeding by cohort

thresh

#> thresh

#[1] -0.4680 -0.6424 -0.8168 -0.9912

prev # Why this gives a prevalence of never breastfeeding and not prevalence of breastfeeding?

# this might be related to that the breastfeeding variable had to be changed to factor so

# that it could be used in a threshold model (originally it is 0-1, factorizing changes it to 1-2,

# and the threshold model estimates the probability of 1, thus never breastfeeding)

#> prev

#[1] 0.3198923 0.2603067 0.2070214 0.1607940

1-prev # so the prevalence of breastfeeding is this

#[1] 0.6801077 0.7396933 0.7929786 0.8392060

# prevalence for twins and sisters

D=-0.078

T=-0.799

cohort=c(0,1,2,3)

thresh=T+D\*cohort # gives thresholds by cohort

prev=pnorm(thresh) # prevalence of never breastfeeding by cohort

thresh

prev

1-prev

# prevalence for mothers of twins

D=-0.078

T=-0.422

cohort=c(0,1,2,3)

thresh=T+D\*cohort # gives thresholds by cohort

prev=pnorm(thresh) # prevalence of never breastfeeding by cohort

thresh

prev

1-prev

# If you want to save the results

#save(Saturated\_Model\_Fit, file="Sat\_BF\_cohort.omxs")

# If you want to load the results back

#load("Sat\_BF\_cohort.omxs")

########################

# Fit saturated submodel 1: thresholds equal between twins and sisters:

# YES, we may equate thresholds between twins and sisters

Saturated\_Model\_Sub1 <- Saturated\_Model\_Fit

Saturated\_Model\_Sub1 <- omxSetParameters(Saturated\_Model\_Sub1, label=c("threshold1","threshold10","threshold41"),

 free=TRUE, values=c(-0.79,-0.79,-0.42),

 newlabels=c("threshold1","threshold1","threshold41"))

Saturated\_Model\_Sub1\_Fit <- mxRun(Saturated\_Model\_Sub1, intervals=T)

Saturated\_Model\_Sub1\_Summ <- summary(Saturated\_Model\_Sub1\_Fit)

Saturated\_Model\_Sub1\_Summ

genEpi\_TableFitStatistics(Saturated\_Model\_Sub1\_Fit)

# Print Comparative Fit Statistics:

genEpi\_TableFitStatistics(Saturated\_Model\_Fit, Saturated\_Model\_Sub1\_Fit)

mxCompare(Saturated\_Model\_Fit, Saturated\_Model\_Sub1\_Fit)

########################

# Fit saturated submodel 2: thresholds equal between family members (twins, sisters, mothers):

# NO, we may not equate thresholds between all the family members

# Twins and sisters yes, but between mothers and daughters no

Saturated\_Model\_Sub2 <- Saturated\_Model\_Fit

Saturated\_Model\_Sub2 <- omxSetParameters(Saturated\_Model\_Sub2, label=c("threshold1","threshold10","threshold41"),

 free=TRUE, values=c(-0.47,-0.47,-0.47),

 newlabels=c("threshold1","threshold1","threshold1"))

Saturated\_Model\_Sub2\_Fit <- mxRun(Saturated\_Model\_Sub2, intervals=T)

Saturated\_Model\_Sub2\_Summ <- summary(Saturated\_Model\_Sub2\_Fit)

Saturated\_Model\_Sub2\_Summ

#parameterSpecifications(Saturated\_Model\_Sub2\_Fit)

genEpi\_TableFitStatistics(Saturated\_Model\_Sub2\_Fit)

# Print Comparative Fit Statistics:

genEpi\_TableFitStatistics(Saturated\_Model\_Fit, Saturated\_Model\_Sub2\_Fit)

genEpi\_TableFitStatistics(Saturated\_Model\_Sub1\_Fit, Saturated\_Model\_Sub2\_Fit)

mxCompare(Saturated\_Model\_Fit, Saturated\_Model\_Sub2\_Fit)

# ---------------------ACE part!------------------------------------

# Set up ACE decomposition, with RawData and Matrices Input

# -----------------------------------------------------------------------

## Modeling

# Matrices declared to store a, c, and e Path Coefficients

# remember label and name cannot be identical

# in bivarite model type="Lower"

pathA <- mxMatrix(type="Symm", nrow=nv, ncol=nv, free=TRUE, values=0.09, label="a1", name="a")

#pathC <- mxMatrix(type="Symm", nrow=nv, ncol=nv, free=TRUE, values=0.01, label="c1", name="c")

pathE <- mxMatrix(type="Symm", nrow=nv, ncol=nv, free=TRUE, values=0.08, label="e1", name="e")

#pathE <- mxMatrix(type="Symm", nrow=nv, ncol=nv, free=FALSE, values=1, label="e1", name="e")

pathC <- mxMatrix(type="Symm", nrow=nv, ncol=nv, free=FALSE, values=0, label="c1", name="c") # shared family environment

pathTw <- mxMatrix(type="Symm", nrow=nv, ncol=nv, free=TRUE, values=0.01, label="tw1", name="tw") # shared twin environment; cannot use just T(program gets confused), thus used Tw

#pathS <- mxMatrix(type="Symm", nrow=nv, ncol=nv, free=TRUE, values=0.01, label="s1", name="s") # shared sibling environment

pathS <- mxMatrix(type="Symm", nrow=nv, ncol=nv, free=FALSE, values=0, label="s1", name="s") # shared sibling environment

#pathTw <- mxMatrix(type="Symm", nrow=nv, ncol=nv, free=FALSE, values=0, label="tw1", name="tw") # shared twin environment; cannot use just T(program gets confused), thus used Tw

# Betas for heritability and environment estimates / Matrices to store the moderated a, c, and e

# now fixed as 0, so doesn't estimate them

# Path coefficients

# Betas for birth cohort effect on heritability

#BetaA1 <- mxMatrix( type="Full", nrow=nv, ncol=1, free=TRUE, values=c(0.08), label=c("ba1"), name="acebA1")

#BetaC1 <- mxMatrix( type="Full", nrow=nv, ncol=1, free=TRUE, values=c(-0.05), label=c("bc1"), name="acebC1")

#BetaE1 <- mxMatrix( type="Full", nrow=nv, ncol=1, free=TRUE, values=c(0.03), label=c("be1"), name="acebE1")

# betas fixed to be zero

BetaA1 <- mxMatrix( type="Full", nrow=nv, ncol=1, free=FALSE, values=c(0), label=c("ba1"), name="acebA1")

BetaC1 <- mxMatrix( type="Full", nrow=nv, ncol=1, free=FALSE, values=c(0), label=c("bc1"), name="acebC1")

BetaE1 <- mxMatrix( type="Full", nrow=nv, ncol=1, free=FALSE, values=c(0), label=c("be1"), name="acebE1")

BetaTw1 <- mxMatrix( type="Full", nrow=nv, ncol=1, free=FALSE, values=c(0), label=c("btw1"), name="acebTw1")

BetaS1 <- mxMatrix( type="Full", nrow=nv, ncol=1, free=FALSE, values=c(0), label=c("bs1"), name="acebS1")

# Algebra for expected Mean and Variance/Covariance Matrices in MZ & DZ twins

# Mean structure, Algebra M to store Expected means

# this is for threshold model

Means <- mxMatrix(type="Zero", nrow=nv, ncol=nv, name="Mean")

expMeans <- mxAlgebra( expression= cbind(Mean,Mean,Mean,Mean), name="expMean" )

#Thresholds <- mxMatrix(type="Full", nrow=nthresh, ncol=nvar, free=TRUE, values=c(-0.9,-0.9,-0.9,-0.9),

# name="Thresh", labels=c("threshold1","threshold1","threshold1","threshold1")) #, dimnames=list('th1',selVars))

#b1 <- mxMatrix(type="Full", nrow=nv, ncol=nv, free=TRUE, labels=("beta1"), values=c(0.1), name="b1")

#Thresholds <- mxMatrix(type="Full", nrow=nthresh, ncol=nvar, free=FALSE, values=c(-0.468,-0.468,-0.468,-0.468),

# name="Thresh", labels=c("threshold1","threshold1","threshold1","threshold1")) #, dimnames=list('th1',selVars))

#b1 <- mxMatrix(type="Full", nrow=nv, ncol=nv, free=FALSE, labels=("beta1"), values=c(-0.1744), name="b1")

#Thresholds <- mxMatrix(type="Full", nrow=nthresh, ncol=nvar, free=FALSE, values=c(-0.875,-0.875,-1.038,-0.362),

# name="Thresh", labels=c("threshold1","threshold1","threshold10","threshold41")) #, dimnames=list('th1',selVars))

#b1 <- mxMatrix(type="Full", nrow=nv, ncol=nvar, free=FALSE, labels=c("beta1","beta1","beta10","beta41"), values=c(-0.04,-0.04,0.13,-0.16), name="b1")

#Thresholds <- mxMatrix(type="Full", nrow=nthresh, ncol=nvar, free=FALSE, values=c(-0.799,-0.799,-0.799,-0.423),

# name="Thresh", labels=c("threshold1","threshold1","threshold1","threshold41")) #, dimnames=list('th1',selVars))

#b1 <- mxMatrix(type="Full", nrow=nv, ncol=nv, free=FALSE, labels=("beta1"), values=c(-0.078), name="b1")

Thresholds <- mxMatrix(type="Full", nrow=nthresh, ncol=nvar, free=TRUE, values=c(-0.799,-0.799,-0.799,-0.422),

 name="Thresh", labels=c("threshold1","threshold1","threshold1","threshold41")) #, dimnames=list('th1',selVars))

b1 <- mxMatrix(type="Full", nrow=nv, ncol=nv, free=TRUE, labels=("beta1"), values=c(-0.078), name="b1")

Def1 <- mxMatrix(type="Full", nrow=nv, ncol=nvar, free=FALSE,

 labels=(rep(c("data.coh4new1","data.coh4new2","data.coh4new10","data.coh4new41"),each=1)),

 name="Def1")

DefR1 <- mxAlgebra(expression= b1 %x% Def1, name="DefR1")

#DefR1 <- mxAlgebra(expression= b1 \* Def1, name="DefR1")

expThreshold <- mxAlgebra(expression= Thresh + DefR1, name="expThresh")

# Matrices to calculate effect on heritability estimates (beta \* definition variable)

covA <- mxAlgebra(expression= (cbind(a,a,a,a) + (acebA1 %x% Def1)) %\*%

 t(cbind(a,a,a,a) + (acebA1 %x% Def1)), name="A")

covC <- mxAlgebra(expression= (cbind(c,c,c,c) + (acebC1 %x% Def1)) %\*%

 t(cbind(c,c,c,c) + (acebC1 %x% Def1)), name="C")

covE <- mxAlgebra(expression= (cbind(e,e,e,e) + (acebE1 %x% Def1)) %\*%

 t(cbind(e,e,e,e) + (acebE1 %x% Def1)), name="E")

covTw <- mxAlgebra(expression= (cbind(tw,tw,tw,tw) + (acebTw1 %x% Def1)) %\*%

 t(cbind(tw,tw,tw,tw) + (acebTw1 %x% Def1)), name="Tw")

covS <- mxAlgebra(expression= (cbind(s,s,s,s) + (acebS1 %x% Def1)) %\*%

 t(cbind(s,s,s,s) + (acebS1 %x% Def1)), name="S")

# Algebra to compute total variances and standard deviations (diagonal only)

covV <- mxAlgebra(expression = A+C+S+Tw+E, name = "V")

# Algebra to compute total variances and standard deviations (diagonal only)

matI <- mxMatrix(name= "I", type="Iden", nrow = nv, ncol = nv)

iSD <- mxAlgebra(name ="iSD", expression = solve(sqrt(I\*V)))

corPh <- mxAlgebra(name ="rPh", expression = iSD%\*%V%\*%iSD)

# Constraint on variance of Binary variables

matUnv <- mxMatrix( type="Unit", nrow=nv, ncol=nv, name="Unv1" )

var1 <- mxConstraint( expression=diag2vec(V)==Unv1, name="Var1" )

var6 <- mxConstraint( diag2vec(expCovMZ)==diag2vec(expCovDZ), name="Var6" )

# Algebra to compute standardized path estimates and variance components

stda <- mxAlgebra(expression=iSD%\*%A%\*%iSD, name="sta")

stdc <- mxAlgebra(expression=iSD%\*%C%\*%iSD, name="stc")

stde <- mxAlgebra(expression=iSD%\*%E%\*%iSD, name="ste")

stdtw <- mxAlgebra(expression=iSD%\*%Tw%\*%iSD, name="sttw")

stds <- mxAlgebra(expression=iSD%\*%S%\*%iSD, name="sts")

h2 <- mxAlgebra(expression=A/V, name="h2")

c2 <- mxAlgebra(expression=C/V, name="c2")

e2 <- mxAlgebra(expression=E/V, name="e2")

tw2 <- mxAlgebra(expression=Tw/V, name="tw2")

s2 <- mxAlgebra(expression=S/V, name="s2")

# Algebras generated to hold Parameter Estimates and Derived Variance Components

rowVars <- rep('vars',nv)

colVars <- rep(c('A','C','E','Tw','S','SA','SC','SE','ST','SS'),each=nv)

estVars <- mxAlgebra( expression=cbind(A,C,E,Tw,S,A/V,C/V,E/V,Tw/V,S/V), name="Vars", dimnames=list(rowVars,colVars))

# WE COMPUTE AN EXTRA MATRIX THAT CONTAINS 3 ELEMENTS: THE STANDARDIZED VARIANCE COMPONENTS FOR WHICH WE WANT CI'S

stndvcs <- mxAlgebra(expression=cbind(A/V,C/V,E/V,Tw/V,S/V),name="stndVCs") # 1x5 matrix

# Algebra for expected variance/covariance matrix in MZ

# so there is twin1, twin2, sister and mother

# by changing these you can specify whether you estimate shared environment for the whole family, for the siblings or specific twin environment

# Here is specific shared twin environment

expCovMZ <- mxAlgebra(name = "expCovMZ",

 expression = rbind (cbind(A+C+Tw+S+E, A+C+Tw+S, 0.5%x%A+C+S, 0.5%x%A+C),

 cbind(A+C+Tw+S, A+C+Tw+S+E, 0.5%x%A+C+S, 0.5%x%A+C),

 cbind(0.5%x%A+C+S, 0.5%x%A+C+S, A+C+Tw+S+E, 0.5%x%A+C),

 cbind(0.5%x%A+C, 0.5%x%A+C, 0.5%x%A+C, A+C+Tw+S+E)))

# Algebra for expected variance/covariance matrix in DZ

expCovDZ <- mxAlgebra(name = "expCovDZ",

 expression = rbind (cbind(A+C+Tw+S+E, 0.5%x%A+C+Tw+S, 0.5%x%A+C+S, 0.5%x%A+C),

 cbind(0.5%x%A+C+Tw+S, A+C+Tw+S+E, 0.5%x%A+C+S, 0.5%x%A+C),

 cbind(0.5%x%A+C+S, 0.5%x%A+C+S, A+C+Tw+S+E, 0.5%x%A+C),

 cbind(0.5%x%A+C, 0.5%x%A+C, 0.5%x%A+C, A+C+Tw+S+E)))

# Objectives

MZObjective <- mxFIMLObjective(covariance="expCovMZ", means="expMean", dimnames=selVars,

 thresholds="expThresh")

DZObjective <- mxFIMLObjective(covariance="expCovDZ", means="expMean", dimnames=selVars,

 thresholds="expThresh")

# Combine Groups

pars <- list(pathA,pathC,

 pathE, pathTw, pathS,

 covA,

 covC,

 covE,

 covTw,

 covS,

 covV,

 matI, iSD,

 matUnv, var1,

 var6,

 corPh,

 stndvcs,

 Means, expMeans, Thresholds,

 b1,

 Def1,

 DefR1,

 expThreshold,

 estVars,

 BetaA1,

 BetaC1,

 BetaE1,

 BetaTw1,

 BetaS1,

 stda, stdc, stde,stdtw,stds,

 h2, c2, e2, tw2, s2,

 expCovMZ, expCovDZ)

# Models

MZmodel <- mxModel("MZmodel", dataMZ, pars,covA,covC,

 covE, covTw, covS,

 covV,

 expCovMZ, MZObjective)

DZmodel <- mxModel("DZmodel", dataDZ, pars,covA,covC,

 covE, covTw, covS,

 covV,

 expCovDZ, DZObjective)

# Objective

min2sumll <- mxAlgebra( expression = MZmodel.objective + DZmodel.objective, name="min2sumll" )

objective <- mxAlgebraObjective("min2sumll")

# WE ASK FOR 95% CONFIDENCE INTERVALS AROUND THE STANDARDIZED VARIANCE COMPONENTS

ci <- mxCI(c("a","c",

 "e", "tw", "s" ,

 "Thresh","b1"

 ), interval = 0.95)

# Cholesky ACE model

CholACEModel <- mxModel("Chol", pars, MZmodel, DZmodel,

 min2sumll, objective, dataMZ, dataDZ, ci)

## Fitting

CholACEFit <- mxRun(CholACEModel, intervals=F)

#CholACEFit <- mxRun(CholACEModel, intervals=TRUE)

 # Generate ACE Output

# -----------------------------------------------------------------------

summary(CholACEFit)

genEpi\_ParameterSpecifications(CholACEFit)

genEpi\_ExpectedMeansCovariances(CholACEFit) # don't look the thresholds from here, gives them separately for MZ and DZ, although in the model only one threshold is estimated

genEpi\_TableFitStatistics(CholACEFit)

AceFitSummFixed <- summary(CholACEFit)

AceFitSummFixed

round(CholACEFit@output$estimate,4) # gives unconstraint estimates (path loadings)

round(CholACEFit$Vars@result,4) # gives constrainted estimates, standardised

# variance components and proportions of variance explained by A, C, and E

########################

# Fit submodel 1c: AE, no T, shared twin environment is not significant:

ACE\_Model\_Sub1c <- CholACEFit

ACE\_Model\_Sub1c <- omxSetParameters(ACE\_Model\_Sub1c, label="tw1", free=FALSE, value=0)

ACE\_Model\_Sub1c\_Fit <- mxRun(ACE\_Model\_Sub1c, intervals=F)

ACE\_Model\_Sub1c\_Summ <- summary(ACE\_Model\_Sub1c\_Fit)

ACE\_Model\_Sub1c\_Summ

#parameterSpecifications(ACE\_Model\_Sub1c\_Fit)

genEpi\_TableFitStatistics(ACE\_Model\_Sub1c\_Fit)

round(ACE\_Model\_Sub1c\_Fit@output$estimate,4)

round(ACE\_Model\_Sub1c\_Fit$Vars@result,4)

# Print Comparative Fit Statistics:

genEpi\_TableFitStatistics(CholACEFit, ACE\_Model\_Sub1c\_Fit)

mxCompare(CholACEFit, ACE\_Model\_Sub1c\_Fit)

########################

# Fit submodel 1d: TwE, no A, heritability is not significant:

# Heritability is significant, so you cannot leave it out

ACE\_Model\_Sub1d <- CholACEFit

ACE\_Model\_Sub1d <- omxSetParameters(ACE\_Model\_Sub1d, label="a1", free=FALSE, value=0)

ACE\_Model\_Sub1d\_Fit <- mxRun(ACE\_Model\_Sub1d, intervals=F)

ACE\_Model\_Sub1d\_Summ <- summary(ACE\_Model\_Sub1d\_Fit)

ACE\_Model\_Sub1d\_Summ

#parameterSpecifications(ACE\_Model\_Sub1d\_Fit)

genEpi\_TableFitStatistics(ACE\_Model\_Sub1d\_Fit)

round(ACE\_Model\_Sub1d\_Fit@output$estimate,4)

round(ACE\_Model\_Sub1d\_Fit$Vars@result,4)

# Print Comparative Fit Statistics:

genEpi\_TableFitStatistics(CholACEFit, ACE\_Model\_Sub1d\_Fit)