Estimating heritability from twin studies

Katrina Grasby[1*], Karin J.H. Verweij[1*], Miriam A. Mosing[2,3*], Brendan P. Zietsch[4], Sarah E. Medland[1]

1. Psychiatric Genetics, QIMR Berghofer Medical Research Institute, Brisbane, Australia
2. Department of Neuroscience, Karolinska Institute, Stockholm, Sweden.
3. Department of Medical Epidemiology and Biostatistics, Karolinska Institute, Stockholm, Sweden.
4. School of Psychology, University of Queensland, Brisbane, Australia

* These authors contributed equally

Corresponding author:

Sarah Medland

sarah.medland@qimrberghofer.edu.au

Running head: Estimating Heritability from Twin Studies

i. Abstract

This chapter describes how the heritability of a trait can be estimated using data collected from pairs of twins. The principles of the classical twin design are described, followed by the assumptions, and some possible extensions of the design. In the second part of this chapter, two example scripts are presented and the basic steps for estimating heritability using the statistical program OpenMx are explained. OpenMx and the scripts used for this chapter can be downloaded so that readers can adapt and use the scripts for their own purposes.

ii. **Key Words**

heritability, behavior genetics, twin, OpenMx, quantitative genetics, twin modeling, classical twin design, genetics, environment

1. Introduction

Individuals vary considerably on physical, cognitive, and behavioral traits, such as height, intelligence, and personality. These individual differences may arise from variation in genes, environmental experiences, or a combination of both, and it is generally accepted that individual differences in most traits are due both to genetic and environmental factors.

To the extent a trait is influenced by genetic (heritable) effects, phenotypic similarity is correlated with genetic relatedness. In this chapter we describe how this principle can be used to estimate the relative magnitude of genetic and environmental influences on trait variation using identical (monozygotic; MZ) and non-identical (dizygotic; DZ) twin pairs. We first describe the principles of the classical twin design, followed by assumptions and extensions of the design. In the second half of the chapter we provide a practical that explains the basic steps required to estimate heritability using the open-access program OpenMx.

1.1. The classical twin design

In studies with data from twins it is possible to partition the observed variance of a trait into genetic, shared environmental, and residual (also known as unshared or unique environmental) variation. Additive genetic variance (*A*) denotes the variance resulting from the sum of allelic effects across multiple genes (non-additive genetic influences will be discussed later in the chapter). Shared (or common) environmental variance (*C*) denotes the variance resulting from environmental influences shared by family members, such as prenatal environment, home environment, socioeconomic status, and residential area. Residual variance (*E*) denotes the variance resulting from environmental influences that are not shared by family members, such as idiosyncratic experiences, stochastic biological effects (such as

illness and injury), and also captures variance within twin pairs arising from differences in perception or salience of the same event, and measurement error.

These different variance components may be estimated using twin data because identical twins share all their alleles while, on average, non-identical twins share half of their alleles at each locus. Accordingly, if MZ twins resemble each other more than DZ twins on a particular trait, this indicates the trait is partly influenced by genetic effects. If all the variance of a trait were due to genetic variance, we would expect a twin pair correlation of 1.0 for MZ twins and 0.5 for DZ twins. Both MZ and DZ twin pairs share environmental influences. If the shared environment were the only source of variance, we would expect a twin pair correlation of 1.0 for both MZ and DZ pairs. Finally, if nether genetic not shared environmental influences contributed to the variance, we would expect a twin pair correlation of zero for both MZ and DZ twin pairs. In reality, the variance of a trait is generally due to a combination of *A, C,* and *E* influences, and this is reflected in the MZ and DZ twin pair correlations.

The correlation between MZ twins (r_{MZ}) can be summarized as $A + C$ and the correlation between DZ twins (r_{DZ}) as $0.5A + C$. Using the observed MZ and DZ twin pair correlations, it is possible to estimate the proportion of variance accounted for by *A, C,* and *E* **(*1*)**.

Given

$$r_{MZ} = A + C \qquad (1)$$

$$r_{DZ} = 0.5A + C \qquad (2)$$

Equations 1 and 2 can be solved for *A* and *C*

$$A = 2(r_{MZ} - r_{DZ}) \qquad (3)$$

$$C = 2r_{DZ} - r_{MZ} \qquad (4)$$

Given standardized variance equals one

$$A + C + E = 1 \qquad (5)$$

Thus, equations 1 and 5 can be solved for *E*

$$E = 1 - r_{MZ} \qquad (6)$$

Rather than simply calculating the genetic and environmental variance components from twin pair correlations, behavioural geneticists typically employ structural equation modeling to more precisely estimate the combination of *A*, *C*, and *E* influences that best explains the observed data. These models can take into account covariate effects, such as age and sex, can compare the fit of various types of models, and provide confidence intervals for the estimates. The basic model is presented in Fig.1. The boxes represent the observed variables (for twin 1 and twin 2), and the circles represent the latent variables (*A*, *C*, and *E*) that influence the observed variables. The double headed arrows show the correlations between the corresponding latent factors. As explained above, for *A* this correlation is 1.0 for MZ twins and 0.5 for DZ twins, and for both MZ and DZ twins the correlation for *C* is 1 and for *E* is zero.

Structural equation modeling of twin data is most commonly performed in the flexible structural equation modeling package OpenMx (2). OpenMx is a free, open source software for use in the R programming environment (see http://openmx.psyc.virginia.edu/). OpenMx employs maximum likelihood modeling procedures to derive parameter estimates through optimization. Effectively, the optimizer searches the parameter space comparing the observed data with expected variance-covariance matrices under different parameter estimates until it reaches the optimum solution. There are several fit functions that can be requested in OpenMx to compute the difference in observed and expected data, the one discussed in this

chapter is minus twice the log likelihood (there are several abbreviations in OpenMx output for this fit function, including -2lnL, -2LL, and minus2LL). The difference in fit between models can be used to assess if a nested model (a model that is a reduced version of another) acceptably fits the data. That is, whether simplifying the model significantly reduces the fit. This is assessed using a likelihood ratio chi-square test in which the test statistic is calculated as the difference in -2 log likelihood between the nested models (i.e. Δ-2LL or diffLL) with degrees of freedom equal to the difference in degrees of freedom between the nested models (i.e. Δdf or diffdf) being asymptotically distributed as a chi-square distribution. If the difference in model fit results in a *P*-value that is less than the critical alpha (typically $P < .05$), then the fit of the reduced model is considered to be significantly worse than that of the parent model. By testing the difference in model fit we can determine whether dropping model parameters (e.g. reducing an *ACE* model to a *CE* model) or constraining parameters to be equal (e.g. equating the male and female *variance component* estimates) significantly worsens the model fit allowing us to make inferences about the significance of parameters.

1.2. Assumptions of the classical twin model

One of the key assumptions of the classical twin design is that trait-relevant environments are similar to the same extent in MZ and DZ twin pairs. If this is not the case then estimates will be biased. For example, if MZ twins are treated more similarly by their parents than DZ twins and this contributes to MZ twin similarity on a trait, then this will falsely inflate the *A* estimate and deflate the *C* estimate for that trait. This *equal environment assumption* can be tested by comparing the similarity of trait-relevant environments between MZ and DZ twin pairs or by comparing the similarity of twin pairs who were mistaken or misinformed about their zygosity. These tests have shown that in general the assumption is valid **(**see, e.g., ref. ***3*)**.

A second assumption is that DZ twins share on average 50% of their alleles. This assumption is only valid if mating occurs randomly in the population and is violated if assortative mating (tendency of individuals to select mates who are similar to themselves) or inbreeding is present. Note that in the context of a complex trait where many loci contribute to trait variation, assortative mating does not mean that mates will share exactly the same alleles across trait-relevant loci, but rather that their overall genetic predisposition will be more similar. Consequently, assortative mating can increase the genetic similarity of DZ twins, which if not taken into account results in a higher *C* estimate and a lower *A* estimate. The effect of assortative mating can be accounted for by adding parents or spouses of twins to the model.

In the standard ACE model, a third assumption is that genetic influences are *additive* and *non-additive* genetic influences are not taken into account; however, the model can be modified to estimate non-additive genetic influences. Non-additive genetic effects include dominance (allelic interactions within loci) and epistasis (interaction between multiple loci). Dominant genetic effects (*D*) predict a twin pair correlation of 1.0 for MZ twins and 0.25 for DZ twins, whereas epistasis predicts a MZ correlation of 1.0 and a DZ correlation between 0 and 0.25 ***(4, 5)***. Dominance and epistatic effects cannot be resolved in a classical twin study, so it is conventional to simply estimate dominance variance *(D*) by specifying an expected MZ correlation of 1.0 and a DZ correlation of 0.25 and accept that this component also includes epistatic variance. When only using data from twins who were reared together, it is not possible to estimate *C* and *D* simultaneously. In the classical twin design, *A, C* and *D* are confounded, they all contribute to MZ and DZ correlations but it is not possible to solve three unknown parameters (*A, C,* and *D*) with only two relevant pieces of information (r_{MZ} and r_{DZ}). The model is under-identified. The choice of an *ACE* or *ADE* model (i.e. a model that includes the components *A, C,* and *E* or *A, D,* and *E*) depends on the pattern of MZ and DZ

correlations; *C* influences increase the DZ correlation relative to the MZ correlation, whereas *D* influences decrease the DZ correlation relative to the MZ correlation. Therefore, *C* is estimated if the DZ twin correlation is more than half the MZ twin correlation, and *D* is estimated if the DZ twin correlation is less than half the MZ correlation. By extending the twin design with parents or children of twins, it is possible to estimate both *C* and *D* in the same model.

A fourth assumption is that there is no correlation or interaction between genes and environment. Gene-environment correlations are present when individuals actively or passively expose themselves to different environments depending on their genotype, such as when individuals' genotypes affect their social interactions or influence the responses they elicit from other individuals. For example, children who are genetically talented at sports are more likely to join a sports club than genetically untalented children, and genetically extroverted children are likely to have different social experiences than genetically introverted children. Environmental influences are then correlated with genetic predisposition. When the environmental influence is not shared between twins, a gene-environment correlation will inflate the *A* estimate in a classical ACE model; whereas when the environmental influence is shared between twins, a gene-environment correlation will inflate *C*. Gene-environment interactions occur when the expression of an individual's genotype depends on the environment. For example, Boomsma et al. ***(6)*** found that a religious upbringing reduced the expression of genetic factors on disinhibition. If gene-environment interaction is present, the relative contribution of genes and environment to the trait variance will differ among individuals as a function of the environment. If not explicitly modelled, gene-environment interaction will inflate the estimate of *E* when the environmental

influence is not shared between twins, or it will inflate the estimate of *C* when the environmental influence is shared between twins.

1.3. Extensions of the classical twin model

Above we described the most basic form of the classical twin design; this model can be extended in various ways. Covariates can be included in the model, which estimate and account for the effects of possible confounders such as sex and age.

Sex differences in genetic and environmental effects can be investigated with the classical twin design if there are data for female and male MZ and DZ twin pairs. Differences between the sexes might be *qualitative*, which result from females and males being exposed to different genetic or environmental factors, or differences might be *quantitative*, which result from the same genetic or environmental factors influencing variation to different magnitudes in females and males. Quantitative sex differences are investigated by estimating separate *A, C,* and *E* parameters for males and females. Estimation of qualitative sex differences, requires data for opposite-sex DZ twins. Qualitative differences in genetic effects between sexes are assessed by estimating the genetic correlation between DZ opposite-sex twins instead of fixing it at 0.5 (which is the genetic correlation of same-sex DZ twin pairs). If completely different genetic factors were influencing males and females, the genetic correlation between opposite-sex twins would be zero. In the same way, sex differences in the source of shared environmental influences can be investigated by estimating the shared environmental correlation for opposite-sex twins instead of fixing it at 1.0. It is not possible to estimate both *A* and *C* opposite-sex correlations in the same model with data only from twins raised together as such a model is under-identified.

The classical twin design can be extended by including additional family members (siblings, parents, children, or spouses). Inclusion of extra family members increases the statistical power, can make it possible to estimate more parameters, and relax assumptions regarding mating and cultural transmission. For example, adding data from non-twin siblings makes it possible to test for twin-specific environmental influences. Including parents in the model makes it possible to simultaneously estimate *C* and *D* influences as well as effects from assortative mating, and the correlation between additive genetic effects and family environment ***(8)***. Alternatively, instead of estimating *D*, by including parents of twins it is possible to estimate familial environmental transmission and sibling environment; the limitation on estimating all of these sources of variance from the addition of parents is—again—under-identification. If more family members are included, such as children and spouses of twins, then genetic and environmental effects that contribute to trait variation can be partitioned into more specific sources.

Correlations between MZ and DZ twin pairs are central to the twin design; these statistics assume normally distributed traits, but for ordinal or dichotomous variables a liability threshold model can be used to estimate these correlations ***(7)***. Threshold models assume there is an underlying continuum of liability (e.g. to depression) that is normally distributed in the population, and that our measurement categories (e.g. depressed/not depressed) are due to one or more artificial divisions (thresholds) overlaying this normal distribution. Analyses are effectively performed on the underlying liability to the trait, resulting in estimates of the heritability of the liability.

Until now we have only discussed univariate analyses, where we were interested in disentangling the variance of a trait into that due to genetic and environmental components. By including more than one dependent variable in a model we can additionally partition the covariance between traits into that due to *A*, *C* (or *D*), and *E* in the same way as we do for the variance of a single trait. Multivariate models can be used to test the extent to which the same genetic or environmental factors influence multiple traits. For example, they can test if there is a correlation between novelty seeking and drug use and, if so, to what extent the correlation is due to overlapping genetic and/or environmental influences. Figure 2 shows a bivariate Cholesky decomposition, the base model used for bivariate analyses ***(9)***. From this base model, parameters can be dropped or equated to test specific hypotheses regarding those parameters. Multivariate twin modeling can be used to analyse numerous variables, and to conduct both exploratory and confirmatory factor, longitudinal, and causal analyses.

2. Methods

In this section, two example scripts (preliminary analyses and *ACE* model fitting) are presented and described. To reduce the complexity of the scripts, the practicals below focus on univariate modeling and use height as the example variable. For the practicals, we use an example dataset included with the distribution of OpenMx. The variable *zyg* in the dataset includes five zygosity-by-sex groups that are each divided into younger and older twins. The younger twins are coded MZ females (1), MZ males (2), DZ females (3), DZ males (4), and DZ opposite-sex pairs (5). The older twins are coded 6–10 in the same order. For the practicals we have only used male twin pairs from the young cohort (zygosity 2 and 4). First, we test assumptions regarding the data, which we use for the final *ACE* modeling. The preliminary analyses on assumptions involve testing whether the means, variances, and covariances significantly differ between different subgroups. For example, are the means and variances of MZ and DZ twins equivalent? We are testing this because in the basic *ACE* model these values are typically equated. If the dataset being analyzed contains data from both male and female twins, testing if variances and covariance differ between sexes is particularly useful to assess whether further analyses on qualitative or quantitative sex effects ought to be conducted. Subsequently, in the second script, we estimate the relative magnitude of *A, C,* and *E* components of the variance in height in males by fitting a univariate *ACE* model. The programs used for the following examples (OpenMx and R) can be downloaded from the following pages, http://openmx.psyc.virginia.edu/installing-openmx and http://www.r-project.org/. OpenMx is not a standalone program but rather a package of functions that runs within the R environment. The scripts used in the following examples can be found at http://www.genepi.qimr.edu.au/staff/sarahMe/twinstudies.html. In the following sections we explain the individual steps within the scripts.

2.1. Preliminary analyses

2.1.1. Setting up the R session

Once R is installed, open the program. OpenMx can be easily installed from within R by pasting and running the following code

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

Once OpenMx is installed, the library can be loaded by running the following code

```
require(OpenMx)
```

In this practical we will also be using the psych package to summarize the data. After it is installed, it can be loaded by running

```
require(psych)
```

For each new R session packages will need to be loaded in order to use their functions.

2.1.2. Data preparation

In this example we will be using the *twinData* dataset that is supplied with the OpenMx program; because this dataset is stored within R it is loaded using the data(twinData) command. Typically data is read into R using a function such as read.table() or read.csv(). For details as to how to prepare a new dataset for analysis in OpenMx, see Note 1.

After reading the data into the R session, the describe function (part of the psych package) provides some useful descriptive statistics. In general, it is a good idea to check your data prior to analysis to ensure the data are complete and missing codes have been read correctly by R. We are going to build the scripts using objects. After an object is assigned content the object can be used in subsequent commands or functions to simplify the script; content is assigned to an object using

<-

First, we specify the number of variables/traits (nv) to be analyzed, the number of twins/individuals (nt), and the product of these becomes our total number of variables (ntv). What we call these objects is arbitrary, as long as the name we give them is not the name of something that has specific meaning in a package. Having objects of the number of variables, the number of individuals, and the total number of variables is particularly useful when scripting multivariate models. Thus we specify the number of variables with nv <- 1, and the number of twins with nt <- 2. Next we use these objects to create another that is the total number of variables, ntv <- nv*nt. This object will be used later to create matrices with correct dimensions to hold the means and the variances and covariances. Next we create an object that is a list of the names of the variables to be analyzed, height of twin 1 and height of twin 2, selVars <- c("ht1","ht2"). The quotation marks are important, if they are omitted R will assume they are names of objects. The content of an object is printed to the screen by running the object name within a script or by typing the object name into the Console; this is useful for checking whether the object has been created correctly, in this case *selVars*. After this, we re-scale the data and increase the variance (to improve the optimization) by converting from meters to cm. Using the describe command again, we can check the new mean and find that height is now reported in cm. Finally, two new datasets are created, one including all male MZ twin pairs (mzData; in this dataset zyg==2) and another including all male DZ twin pairs (dzData; zyg==4). Again punctuation is important; the double equal sign indicates that a condition is being set, while a single equal sign is another way to assign a number to an object.

```
# Prepare Data
# ------------------------------------------------------------------------
data(twinData)
describe(twinData)
nv  <- 1
nt  <- 2
ntv <- nv*nt
selVars <- c("ht1","ht2")
selVars
twinData$ht1  <- twinData$ht1*100
twinData$ht2  <- twinData$ht2*100
describe(twinData)
mzData      <- subset(twinData, zyg==2, selVars)
dzData      <- subset(twinData, zyg==4, selVars)
```

2.1.3. Saturated model fitting

After data preparation and specifying starting values for means, variances, and covariances (see Note 2), a univariate saturated model (in which all possible parameters are estimated) is fitted in order to estimate the means, variances, and covariances that best fit the observed data. The goodness-of-fit statistic of this saturated model is later compared to that of the more reduced models in which certain parameters are dropped (i.e. fixed to equal zero) or equated. In this way we can test whether means, variances, and covariances of different sub-samples are significantly different or whether covariates (e.g. age) significantly influence the trait. The model (modelUniTwinSat) used in the present example consists of two submodels, one for the MZ twin data (shown below) and one for the DZ twin data (omitted). To include more zygosity groups (e.g. female MZ, female DZ, opposite-sex DZ) we would increase the number of submodels. Using the mxMatrix function we create a matrix that is free (i.e. parameters will be estimated as opposed to being fixed at a specific value), symmetric (the upper triangle of the matrix is constrained to be equal to the lower triangle, which will be estimated). Each row and column of this matrix is equal to the total number of variables (ntv), thus it is a two by two matrix in this example. This matrix is called *expCovMZ* and contains the expected variances and covariance for twin 1 height and twin 2 height. We also create a free and full matrix (i.e. all elements are estimated) with one row and the number of columns

equal to the total number of variables, thus two columns in this example. This matrix is called *expMeansMZ* and contains the expected means for twin 1 height and twin 2 height. The dataset is specified using the mxData function. The combination of mxExpectationNormal and mxFitFunctionML functions specifies that full-information maximum likelihood is used to estimate the free parameters in the expected means (*expMeansMZ*) and variance-covariance (*expCovMZ*) matrices so that -2 log likelihood is minimized. It is important that the covariance argument and the means argument of the mxExpectationNormal function specify the names of the matrices that hold the estimated parameters and not the object names. The most common mistakes make when writing or editing openMx scripts involve the omission or misplacement of commas and brackets examples of the error codes generated by these mistakes are provided in Note 3.

```
# Fit Univariate Saturated Model
# ------------------------------------------------------------------------
# For MZ twins
covMZ    <- mxMatrix(type="Symm", nrow=ntv, ncol=ntv, free=TRUE, values=StMZcov,
                     labels=c("expVarMZt1","CovMZ","expVarMZt2"), name="expCovMZ")
meanMZ <- mxMatrix(type="Full", nrow=1, ncol=ntv, free=TRUE, values=StMZmean,
                     labels=c("expMeanMZt1","expMeanMZt2"), name="expMeanMZ")
dataMZ   <- mxData(observed=mzData, type="raw" )
expMZ    <- mxExpectationNormal(covariance="expCovMZ", means="expMeanMZ", dimnames=selVars)
fitfun     <- mxFitFunctionML()
```

After the objects of the submodels are created, each submodel is specified by listing all the necessary objects in the mxModel function and supplying each model with a name. Next, an object is created using the mxFitFunctionMultigroup function; by listing each submodel this specifies that the parameters are to be estimated and the fit optimized across all submodels simultaneously. Finally, the whole model is specified, again by using mxModel and listing the objects of the submodels and the mxFitFunctionMultigroup.

```
modelMZ          <- mxModel("MZ", covMZ, meanMZ, dataMZ, expMZ, fitfun )
modelDZ          <- mxModel("DZ", covDZ, meanDZ, dataDZ, expDZ, fitfun )
multi            <- mxFitFunctionMultigroup( c("MZ","DZ"))
modelUniTwinSat <- mxModel("UniTwinSat", modelMZ, modelDZ, multi)
```

2.1.4. Running the models and generating output

After building the model, we run it. By copying the fitted/run model into a fit object we can print to the screen information on the fitted model. Requesting summary statistics is a good starting place.

```
# Run the model
fitUniTwinSat   <- mxRun(modelUniTwinSat)

# Print output
summary(fitUniTwinSat)
modelUniTwinSat$MZ$matrices
modelUniTwinSat$DZ$matrices
fitUniTwinSat$output$matrices
mxCompare(fitUniTwinSat)
```

From the summary we can obtain the fit of the model (e.g. -2lnL), the estimated parameters, and their standard errors. This information can also be extracted with command lines, such as

fitUniTwinSat$output$fit

Similarly, we can extract the expected variance-covariance matrices

fitUniTwinSat$output$matrices

We can also view details of the specified model prior to it having been run, which is useful for checking that elements of matrices have been labeled correctly and allocated appropriate start values

modelUniTwinSatMZmatrices

(see Note 4 for screenshots of output and more detailed explanations). The results show that the expected means for height were about 177 cm and the expected variances were close to 45 for both twins in each of the MZ and DZ groups. The significance of any differences between these means and variances is formally tested in the assumption testing steps.

2.1.5. Assumption testing (mean differences)

The fewer parameters we estimate in a model, the more power we have to estimate them. We expect that our data are representative of the general population and that the trait (height) is normally distributed in the general population. Therefore, the most basic twin model assumes that there are no significant differences in means and variances between the different groups (e.g. twin 1, twin 2, MZ, DZ, males, females, etc.). If there are no significant differences, it means that data from all groups contribute to estimating a single parameter for the mean of height. However, before running the most basic twin model, we need to test whether these assumptions are valid in our data. Violations of these assumptions are not problematic, but need to be accounted for with more complex twin models that allow for separate parameter estimates for different subsamples and/or include covariates (e.g. sex differences). To start with, we test if the means of twin 1 and twin 2 (within each of the MZ and DZ groups) are significantly different. This tests potential birth effects in samples where twin 1 is the first born and twin 2 is the second born. To test this, the means are equated and the model fit is compared to the original saturated model. If the model fit of the restricted model is not significantly worse, the means are not significantly different and we keep them equated in subsequent modeling.

To equate the means, a new model object is created from the fitted saturated model, *fitUniTwinSat*, and given a new name, *modelEquateMeans1*. The function omxSetParameters is used to modify parameters in the model. To use this function, we first indicate the name of the model to be modified, then use the labels argument to list the labels to be renamed, and the newlabels argument to specify the new label. It is simplest to repeat the use of omxSetParameters for each new label. For example, we have taken the parameters previously estimated as “expMeanMZt1” and “expMeanMZt2” and called both of them “expMeanMZt1t2” in the reduced model. Because we are taking estimated parameters that

probably have different values and creating a model where they will be estimated as a single parameter with the same value, we use the omxAssignFirstParameters function to specify that all occurrences of parameters with the same label will have the same starting value. Then the new reduced model is run and the summary statistics and output are requested.

```
# Assumption Testing - Means - check for mean differences between the different groups
# -----------------------------------------------------------------------
# Check for birth order effects - equate means of twin 1 and twin 2 for MZ and DZ twins
modelEquateMeans1 <- mxModel(fitUniTwinSat, name="equateMeans1")
modelEquateMeans1 <- omxSetParameters( modelEquateMeans1, labels=c("expMeanMZt1","expMeanMZt2"),
                                       newlabel="expMeanMZt1t2" )
modelEquateMeans1 <- omxSetParameters( modelEquateMeans1, labels=c("expMeanDZt1","expMeanDZt2"),
                                       newlabel="expMeanDZt1t2" )
modelEquateMeans1 <- omxAssignFirstParameters(modelEquateMeans1)
fitEquateMeans1       <- mxRun(modelEquateMeans1)
summary(fitEquateMeans1)
modelEquateMeans1$MZ$matrices
modelEquateMeans1$DZ$matrices
fitEquateMeans1$output$matrices
mxCompare(fitUniTwinSat, fitEquateMeans1)
```

The output from the mxCompare function provides fit statistics on the saturated and the reduced model along with the *P*-values of the likelihood ratio test. As explained in Subheading 1.1, the change in fit between two nested models (diffLL) is asymptotically distributed as chi-square with degrees of freedom equal to the difference in degrees of freedom (diffdf) between the two models. Thus, it is possible to test whether constraining the means to be equal significantly worsens the model fit. With assumption testing the most sensitive test is to compare each reduced model with the previous model. For further information regarding output, model fit, and comparing models see Note 4. In the present example, the output shows no significant mean differences between twin 1 and twin 2. Other assumptions of the most basic twin model that are tested (not shown here but in the script online) are whether we can equate:

- the means of the MZ and DZ groups,

- the variances of twin 1 and twin 2 within each of the MZ and DZ groups,

- the variances of the MZ and DZ groups,
- the covariance of the MZ and DZ groups (this is not an assumption, but is a preliminary test for genetic effects).

In a model including data from female MZ and DZ twins, we would set up submodels for each of the four groups in the saturated model, and after testing for mean differences between zygosity groups, we would then equate means across females and males. If there are significant mean differences between the sexes the basic twin model can easily be extended to include sex as a covariate on the means. If variances and covariance are different between the sexes, then the basic twin model should be extended to test for qualitative and quantitative sex effects. To test assumptions regarding opposite-sex DZ twins it is common to arrange the data by sex such that all twin 1 are female and twin 2 are male. To model both the effects of sex and birth order two opposite-sex twin groups are required, one in which twin 1 is male and second in which twin 1 is female.

For the present example, assumption testing showed no mean or variance differences between twin 1 and twin 2 or between the MZ and DZ groups. Therefore, in the *ACE* model below, we only estimate one mean and one variance for the whole sample (i.e. we equate the means and the variances over the different groups). The covariance between MZ twins was significantly higher than for DZ twins, indicating a genetic influence on height variation. This will be formally tested in the *ACE* model below.

2.2. Univariate *ACE* modeling

2.2.1 Saturated model fitting

After the data are prepared and starting values are specified for the standardized *A*, *C*, and *E* estimates (see Note 2), a saturated *ACE* model (*modelAce*) is fitted to estimate the relative contribution of *A, C,* and *E* to variation in height (i.e. the square of the *a, c,* and *e* pathways presented in Fig. 1). Three free lower matrices (i.e. each element in the upper triangle of the matrix is fixed to zero, the other elements are estimated) are created using the mxMatrix function, one for each of the *a*, *c,* and *e* estimates. Note that in a univariate model the lower matrix would be a one by one matrix; however, the scripts are set up so that they can be easily expanded to a bivariate or multivariate model. We compute the variance components for *A, C,* and *E* using matrix multiplication (a Kronecker product, for which the R notation is %*%) and multiplying *a* by the transpose of *a*. This is followed by algebra to compute the total variance by summing the matrices of the three variance components *A*, *C*, and *E* to produce a matrix called *V*. In multivariate models this *V* matrix will hold total trait variances on the diagonal. To standardize the variance components, we first create an identity matrix (*I*), which contains a value of one in all diagonal elements and zeros in all off-diagonal elements. Second, we create a matrix with the standard deviations on the diagonal by taking the square root of the total variance matrix multiplied by the identity matrix. Third, we then take the inverse of the standard deviation, thus matrices that hold path estimates can be multiplied by this matrix (*iSD*) to be standardized.

```
# Fit ACE Model with RawData and Matrices Input
# ------------------------------------------------------------------------------
pathA    <- mxMatrix( type="Lower", nrow=nv, ncol=nv, free=TRUE, values=St_a, labels="a11", name="a" )
pathC    <- mxMatrix( type="Lower", nrow=nv, ncol=nv, free=TRUE, values=St_c, labels="c11", name="c" )
pathE    <- mxMatrix( type="Lower", nrow=nv, ncol=nv, free=TRUE, values=St_e, labels="e11", name="e" )

covA     <- mxAlgebra( expression=a %*% t(a), name="A" )
covC     <- mxAlgebra( expression=c %*% t(c), name="C" )
covE     <- mxAlgebra( expression=e %*% t(e), name="E" )

covP     <- mxAlgebra( expression=A+C+E, name="V" )
matI     <- mxMatrix( type="Iden", nrow=nv, ncol=nv, name="I")
invSD    <- mxAlgebra( expression=solve(sqrt(I*V)), name="iSD")
```

Using the mxAlgebra function, three objects are created containing the standardized path coefficients for the *a*, *c*, and *e* effects. This is achieved by multiplying the matrix of the inverse standard deviation by the path coefficients *a*, *c*, and *e*. Note that the order of matrices in star matrix multiplication is important, particularly when expanding to multivariate scripts. Matrices are also created to hold the standardized genetic and environmental variance components. This is achieved by dividing the respective *A*, *C*, and *E* matrices by the total variance matrix (*V*).

```
stpatha   <- mxAlgebra( iSD %*% a, name="sta")
stpathc   <- mxAlgebra( iSD %*% c, name="stc")
stpathe   <- mxAlgebra( iSD %*% e, name="ste")

stcovA   <- mxAlgebra( A/V, name="stA")
stcovC   <- mxAlgebra( C/V, name="stC")
stcovE   <- mxAlgebra( E/V, name="stE")
```

A means vector is created for the expected means of twin 1 and twin 2. Only one mean value is estimated because the assumption testing indicated that the means can be equated across twin 1 and twin 2 and across zygosity. We indicate in the script that it is the same value by using only one label for each element in the means matrix. Next, the algebra for the expected variance/covariance matrix for MZ and DZ twins is specified. The function cbind joins the matrices horizontally while rbind joins the expected variances and covariances together vertically. For the DZ twins, *A* is multiplied by 0.5 using a Kronecker product (for which the R notation is %x%), as only half of the genes are shared between the DZ twins.

```
meanG   <- mxMatrix( type="Full", nrow=1, ncol=ntv, free=TRUE, values=Stmean, label=c("mean","mean"),
                     name="expMean" )

covMZ    <- mxAlgebra( expression= rbind( cbind(A+C+E , A+C),
                                          cbind(A+C   , A+C+E)), name="expCovMZ" )
covDZ    <- mxAlgebra( expression= rbind( cbind(A+C+E     , 0.5%x%A+C),
                                          cbind(0.5%x%A+C , A+C+E)), name="expCovDZ" )
```

After this, the data objects specify the data and the type of data to be used in the MZ and DZ submodels. Then the expectations of each submodel are defined, mxExpectationNormal will calculate these under the assumption of multivariate normality. Because the assumption testing indicated that the means could be equated across zygosity groups, *expMean* is used for both submodels; in contrast the covariances are different for each submodel. The mxFitFunctionML function specifies that the model fit is to be optimized by computing the -2 log likelihood. Each submodel requires data, expectations, and a fit function, and the same type of fit function is used in each submodel.

```
dataMZ   <- mxData( observed=mzData, type="raw" )
dataDZ   <- mxData( observed=dzData, type="raw" )

expMZ    <- mxExpectationNormal( covariance="expCovMZ", means="expMean", dimnames=selVars )
expDZ    <- mxExpectationNormal( covariance="expCovDZ", means="expMean", dimnames=selVars )
fitfun   <- mxFitFunctionML()
```

Each of the MZ and DZ models are specified with the mxModel function by listing all of their objects. This has been abbreviated in this script by creating a list of objects that are in common to each submodel and placing the object of that list in each submodel. The fit of the model as a whole is specified with the mxFitFunctionMultigroup. Finally, 95% confidence intervals are specified on the standardized variance components, and the confidence interval object included in the final model. In order to estimate the confidence intervals the argument "intervals = TRUE" is included when the model is run.

```
pars      <- list( pathA, pathC, pathE, covA, covC, covE, covP, matI, invSD, stcovA, stcovC, stcovE, stpatha, stpathc,
                   stpathe )
modelMZ   <- mxModel( pars, meanG, covMZ, dataMZ, expMZ, fitfun, name="MZ" )
modelDZ   <- mxModel( pars, meanG, covDZ, dataDZ, expDZ, fitfun, name="DZ" )
multi     <- mxFitFunctionMultigroup( c("MZ","DZ"))
CIs       <- mxCI( c('stA','stC','stE'))
modelAce  <- mxModel( "ACE", pars, modelMZ, modelDZ, multi, CIs )
fitAce    <- mxTryHard(modelAce, intervals=TRUE)
```

2.2.2. Fitting submodels

After the general *ACE* model is run, submodels can be fitted to test the significance of parameters specified in the saturated model. To do this we simply drop the parameter of interest, by fixing it to zero, and compare the model fit of the reduced with the full model. Based on the parameter estimates in our example, we dropped the *C* component first (shown below), by setting the *c* estimate to zero, and fitted this *AE* model. The results of the likelihood ratio test are reported using mxCompare; if there is no significant loss of model fit from reducing a model, then this supports a conclusion that the more parsimonious model is an acceptable model of the data. If *A* could be dropped from the model, this would suggest no significant genetic influences on the trait. Note that the *E* parameters can never be dropped as *E* includes measurement error.

```
# Fit AE model
# ------------------------------------------------------------------------------
modelAe   <- mxModel( fitAce, name="AE")
modelAe   <- omxSetParameters( modelAe, labels="c11", free=FALSE, values=0 )
fitAe     <- mxTryHard(modelAe, intervals=FALSE)
summary(fitAe)
mxCompare(fitAce,fitAe)
```

In the assumption testing script we tested each reduced model against the previous model, we can also test each reduced model against the saturated model. This is done by creating a list in the second argument of the mxCompare function. For more information on model output and model fit please see Note 4.

Notes

1**.** *Data set preparation*

OpenMx uses a very flexible file format. The only real restriction is that data from each unit of analysis need to be on a separate line. Thus, if we were running a multivariate analysis on a sample of unrelated individuals (where the unit of analysis is an individual) each line in the data file would contain the data for a different person, and all the variables for an individual would be one line. In the analysis of twin data, data for each twin pair (or family) are listed on a separate line. The only other restriction on the data is that missing values are correctly identified within the data, either using the default missing code "NA" or by specifying how missing data are coded when reading in the data, e.g. via the argument "na.strings=-99".

Data can be prepared using a standard data management or statistics package (e.g. SPSS or SAS). Alternatively, the data can be prepared within R. Numerous web based tutorials have been developed that describe importing and managing data in R; we recommend the Quick-R pages e.g., http://www.statmethods.net/input/index.html.

2. *Starting Values*

OpenMx optimises a likelihood function under the provided model and tries to find the combination of parameter estimates that best fit the data. It does this via iteration: OpenMx chooses a set of parameters and determines the difference between the observed and the expected variance-covariance matrix, then chooses another set of parameters and refits the model; this process continues until no further improvement in model fit is available. For the first iteration it is necessary to provide a reasonable starting point for each of the parameters to be estimated to speed up the optimisation process. Starting values should ideally be close

to the actual estimates, but starting the optimizer at the solution itself is problematic and the analyses generally fail.

For the expected means and variance-covariance matrix it is possible to use the observed means, variances, and covariances. You could manually enter starting values in the *values* argument of the mxMatrix function. Alternatively, users can take advantage of using objects and the ability to calculate means and variances using inbuilt R functions. For example

```
StMZmean <- vech(mean(mzData,na.rm=T))
```

This code obtains the means of each variable in the mzData dataframe and places them as a vector in the object called StMZmean. This object can then be used in the values argument of the matrix that will hold the MZ means

```
mxMatrix( type="Full", nrow=1, ncol=ntv, free=TRUE, values=StMZmean,
name="expMeanMZ" )
```

Determining starting values for the *a*, *c*, and *e* parameters is less straightforward. Based on the observed data you know what the total variance for a trait is. You can use the observed twin pair correlations and Falconer's formulas ***(7)*** (see equations 3, 4, and 6 in section 1.1) to determine the approximate proportions of *A*, *C*, and *E*. Use these approximations to partition the unstandardized trait variance into *A*, *C*, and *E;* then take the square root of these unstandardized variance components to obtain estimates of the unstandardized *a*, *c*, and *e* paths. These can be then be used as start values. In general, it is a good idea to provide higher starting values for the *e* estimates than for *a* or *c* as this avoids the possibility of non-positive definite matrices (where the estimates for the covariances become larger than the estimates of the variance).

In our script, we set starting values for the *a* path coefficient by using inbuilt R functions to calculate the standard deviation of height in our data and multiply it by a guesstimate of the magnitude of the effect. In this way start values will be place on an appropriate scale for the unstandardized estimates but the values will be inexact. For example

```
St_a <-(vech(sd(twinData$ht1, na.rm=T)))*.3
```

Providing starting values becomes more complicated when the model includes more than one dependent variable, in which case the cross-twin-cross-trait variance-covariance matrix can be used to assist in obtaining appropriate starting values for the pathways.

3**.** *Explanation of errors when commas and brackets are omitted or in excess and how to remove/find them*

There are two main types of errors encountered when running OpenMx, errors relating to the code and errors relating to the optimization. The three most common code errors are typos, missing commas/brackets/quotation marks, and extra commas/brackets/quotation marks. Typos typically result in an error message that looks something like this

```
Error in mxModel("MZ", covMZ, meanMZ, dataMZ, expMZ, fit) :
  object 'fit' not found
```

A missing comma typically results in an error message that looks something like this

```
Error: unexpected symbol in "modelMZ   <- mxModel("MZ", covMZ, meanMZ,
dataMZ, expMZ fitfun"
```

Whereas an extra comma typically results in an error message that looks something like this:

```
Error in mxModel("MZ", covMZ, meanMZ, dataMZ, expMZ, fitfun, ) :
  argument is missing, with no default
```

There are also a number of important optimizer error messages. Any serious errors will be reported in the output. However, it is worth checking for errors on each analysis run. For the *ACE* example these can be viewed by typing:

```
fitUniTwinSat@output$status[1]
```

which yields the following output:

```
> fitUniTwinSat@output$status[1]
$code
[1] 0
```

The status code, which in this case is 0, summarizes the status of the optimizer run, which can take several possible values. A value of 0 means a successful optimization — no error returned. A value of 1 means that it is highly likely that an optimal solution was found but the iterations did not converge. These estimates can generally be considered correct solutions, so this code is labeled (Mx status GREEN). Status codes of -1, 2, 3, 4, 5, 6 reflect critical optimizer failures and results should not be used. Strategies to deal with these errors include, check that the model is correctly specified, check that the starting values are realistic, check that the variance-covariance matrix is positive definite. It is possible that the optimizer required more time to reach a solution, using the function mxTryHard instead of mxRun will take the best fitting parameter estimates from one run and use them as starting values for a subsequent run. The default maximum number of runs for mxTryHard is 11.

4. *Output*

If we request the matrices from our specified model (i.e. the model before it is run, not the fitted model), we can see which parameters we decided to estimate.

```
> modelUniTwinSat$MZ$matrices
$expCovMZ
SymmMatrix 'expCovMZ'
$labels
     [,1]       [,2]
[1,] "expVarMZt1" "CovMZ"          check labels are in
[2,] "CovMZ"      "expVarMZt2"     the correct position
$values
       [,1]     [,2]
[1,] 45.64889 39.49632             check start values
[2,] 39.49632 43.86638             are sensible
$free
     [,1] [,2]
[1,] TRUE TRUE                     TRUE elements
[2,] TRUE TRUE                     are estimated
$lbound: No lower bounds assigned.
$ubound: No upper bounds assigned.
```

If we request matrices from the fitted model, we can display the expected variance-covariance matrix as well as the expected means for MZ and DZ twins. Here we can see that the differences in means and variances between zygosity groups and twin 1 and twin 2 are small, although the significance of any differences was assessed in the assumption testing.

```
> fitUniTwinSat$output$matrices
$MZ.expCovMZ
       [,1]     [,2]
[1,] 44.50668 38.40409
[2,] 38.40409 42.76752
$MZ.expMeanMZ
       [,1]     [,2]
[1,] 177.7721 177.6414
$DZ.expCovDZ
       [,1]     [,2]
[1,] 48.34238 15.66656
[2,] 15.66656 43.19497
$DZ.expMeanDZ
       [,1]     [,2]
[1,] 177.7643 177.5032
```

After equating the means of twin 1 and twin 2 within each zygosity group, we can see those means matching when we request to see the matrices from the fitted reduced model

```
> fitEquateMeans1$output$matrices
$MZ.expCovMZ
       [,1]    [,2]
[1,] 44.52930 38.40438
[2,] 38.40438 42.76180
$MZ.expMeanMZ
       [,1]    [,2]
[1,] 177.6962 177.6962
$DZ.expCovDZ
       [,1]    [,2]
[1,] 48.36900 15.66329
[2,] 15.66329 43.21206
$DZ.expMeanDZ
       [,1]    [,2]
[1,] 177.6215 177.6215
```

The mxCompare output reports the fit of each model in the column called minus2LL; the diffLL reports the difference in model fit, and diffdf reports the difference in degrees of freedom between the two models. The reduced model should have more degrees of freedom and a poorer fit (larger minus2LL) than the full model. We can see from these results that there is no significant loss of fit when the means were equated across twin 1 and twin 2 within the zygosity groups.

```
> mxCompare(fitUniTwinSat,   fitEquateMeans1)
   base          comparison     ep  minus2LL  df      AIC          diffLL          diffdf   p
1 UniTwinSat   <NA>            10  5740.330   917     3906.330     NA              NA       NA
2 UniTwinSat   equateMeans1    8   5740.974   919     3902.974     0.6443898       2     0.724557
```

The summary of the *ACE* model is shown below. First the parameter estimates for *a*, *c*, and *e* are shown, each with its standard error. Subsequently, standardized variance components of *A* (heritability), *C*, and *E* are presented with confidence intervals. The exclamation marks against the standardized *C* variance component indicate that the 95% confidence intervals were not suitably estimated, in this case it is probably the result of the near zero estimate for *C*. Next, information about the model fit, including -2 log likelihood (-2lnL) and Aikaike's A Information Criterion (AIC). A smaller AIC indicates a better model fit. Reduced models

always have a greater -2 log likelihood, indicating a worse fit, but as long as this fit is not significantly worse, the more reduced submodel is the more parsimonious one.

```
> summary(fitAce)
Summary of ACE

free parameters:
  name    matrix row col    Estimate Std.Error A
1  a11        a   1   1 6.374162e+00 0.2005139
2  c11        c   1   1 2.462840e-07 1.1262497
3  e11        e   1   1 2.312030e+00 0.1028260
4 mean MZ.expMean   1 ht1 1.776703e+02 0.2846936

confidence intervals:
                lbound    estimate    ubound note
ACE.stA[1,1] 7.975177e-01 8.837317e-01 0.90502701
ACE.stC[1,1] 1.319309e-15 1.319309e-15 0.08473914  !!!
ACE.stE[1,1] 9.497295e-02 1.162683e-01 0.14311327

observed statistics:  927
estimated parameters:  4
degrees of freedom:  923
fit value ( -2lnL units ):  5745.863
number of observations:  479
Information Criteria:
    |  df Penalty  |  Parameters Penalty  |  Sample-Size Adjusted
AIC:    3899.86258           5753.863                NA
BIC:     49.38292           5770.549           5757.854
```

The likelihood ratio test from the mxCompare function tests if the reduced model fit is significantly worse. We can either compare the reduced model to the previous one, or to the full model, in this example all nested models are compared to the saturated model. The results indicate that C can be dropped from the model without a significantly poorer model fit, but A cannot. Therefore, genetic effects significantly influence variation in height in males.

```
>mxCompare(fitUniTwinSat,list(fitAce,fitAe,fitCe,fitE))
     base       comparison  ep  minus2LL  df       AIC         diffLL      diffdf  p
1   UniTwinSat     <NA>    10  5740.330  917     3906.330    NA          NA      NA
2   UniTwinSat     ACE     4   5745.863  923     3899.863    5.53248     6       4.775396e-01
3   UniTwinSat     AE      3   5745.863  924     3897.863    5.53248     7       5.952689e-01
4   UniTwinSat     CE      3   5911.061  924     4063.061    170.730     7       1.760800e-33
5   UniTwinSat     E       2   6156.235  925     4306.235    415.905     8       7.402437e-85
```

4. References

1. Holzinger KJ (1929) The relative effect of nature and nurture influences on twin differences. J Educ Psychol **20**: 241-248
2. Neale MC, et al. (2016). OpenMx 2.0: Extended structural equation and statistical modeling. Psychometrika **81**: 535-5549
3. Kendler KS, et al (1993) A test of the equal-environment assumption in twin studies of psychiatric illness. Behav Genet **23**: 21-27
4. Keller MC, Coventry WL (2005) Quantifying and addressing parameter indeterminacy in the classical twin design. Twin Res Hum Genet **8**: 201-213
5. Mather K (1974) Non-allelic interaction in continuous variation of randomly breeding populations. Heredity **32**: 414-419
6. Boomsma DI, et al (1999) A religious upbringing reduces the influence of genetic factors on disinhibition: Evidence for interaction between genotype and environment on personality. Twin Res **2**: 115-125
7. Falconer DS (1989) Introduction to quantitative genetics, Longman Scientific and Technical, Harlow, Essex, UK
8. Keller MC, et al (2009) Modeling Extended Twin Family Data I: Description of the Cascade Model. Twin Res Hum Genet **12**: 8-18
9. Neale MC, Cardon LR (1992) Methodology for Genetic Studies of Twins and Families. Kluwer, Dordrecht