Intro

Q1.1. Today's practical we will run through a number of ways to construct an ACE model in OpenMx and have a look at power.

Q1.2. As you get started, introduce yourselves and let us know your breakout room:

Q1.3. What are the names of the people in your room?

Q1.4. If you haven't already done so, open the workshop computing environment
https://workshop.colorado.edu

# Open the workshop SSH client
# Create a directory to hold today's work

```
mkdir day2
```

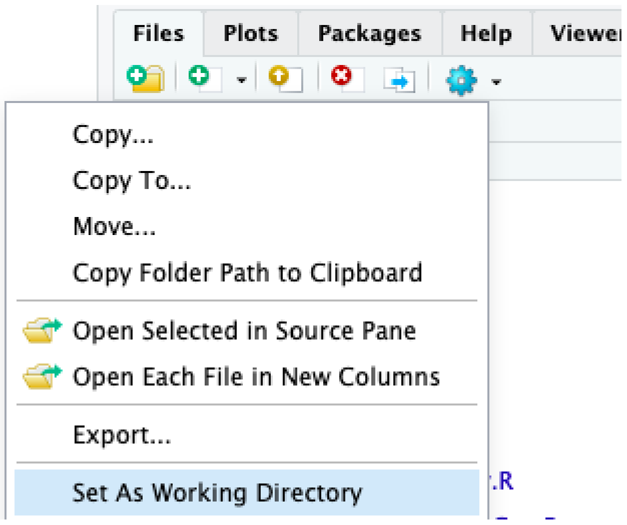
# Change into that directory, and then copy over the exercises.

```
cd day2
cp /faculty/katrina/2022/* .
```

# Open the workshop Rstudio client
# Open the folder day2 (bottom right quadrant of the screen)
# Set this folder as your working directory. If you are in your home directory (which is where you will be first on login in), then you can set the working directory with this command:

```
setwd('day2')
```

# Or by using the gear icon


Files
Plots
Packages
Help
Viewer
Copy...
Copy To...
Move...
Copy Folder Path to Clipboard
Open Selected in Source Pane
Open Each File in New Columns
Export...
Set As Working Directory

Q1.5. We have one set of scripts that are set up to run with a continuous phenotype and another that are set up to run with a binary phenotype.

As a group, choose which you would like to do for this practical.
The continuous section of this practical is more straight-forward.
Remember, you can access both of them later on if you wish to.

- ○ Continuous
- ○ Binary

Continuous

Q2.1. Open **00_ACEvc_contin.R**

This script is a univariate ACE script, like the one that you worked through in the Day 1 Practical. This one incorporates age and sex effects as covariates on the traits means.

For the sex variable in the data, females are coded as 0 and males are coded as 1.

Note. The data is simulated. The phenotype can be whatever you want it to be.

Q2.2. Run the script to the bottom of the section that creates algebra for expected means matrices (~line 58).

Look at these two lines:

```
defSex        <- mxMatrix( type="Full", nrow=1, ncol=nt,
free=FALSE, labels=c("data.sex1","data.sex2"), name="Sex" )
defAge        <- mxMatrix( type="Full", nrow=1, ncol=nt,
free=FALSE, labels=c("data.age1","data.age2"), name="Age" )
```

Putting **data.** in the label tells OpenMx that this is a definition variable and the values will be updated for each case in the data set.

There can be no missing data on a definition variable or the model will not run. If your data set is incomplete (i.e. you have incomplete sets of twin pairs) you might need to recode missing values with a dummy code (i.e. the mean of the variable). Cases that are missing data on the trait are not used fitting the model, so what value you use to recode a missing definition value will not matter. However, if there is trait data for that case, then the recoded data will be treated as a genuine value.

Run the script to the bottom of the section that creates model objects for multiple groups (~line 88).

Here we have created objects that each have a list of other objects:

```
defs       <- list(defAge, defSex)
pars       <- list( intercept, betaS, betaA, covA, covC, covE,
covP)
```

The definition variables have been split out from the rest of the list of objects. This is because we will want to put the objects for definition variables into the MZ and DZ submodels, because definition variables need to go in an mxModel that includes mxData.
We have the second list of objects because it includes objects that may be used in each level of the model.

Run the script to create the final model (~line 100).

We have created an object to extract the unstandardised and standardised variance components.

```
estVC      <- mxAlgebra(
expression=cbind(VA,VC,VE,VA/V,VC/V,VE/V), name="VarC",
dimnames=list(rowVC,colVC) )
```

And can request confidence intervals on the elements in that object. Here we request them on the standardised variance components.

```
ciACE      <- mxCI( "VarC[1,4:6]" )
```

Then put together the final model, which includes the objects for CIs and the constraint on the variance.

```
modelACE  <- mxModel( "ACEvc", modelMZ, modelDZ, multi, pars,
estVC, ciACE )
```

Run the script to fit the model.

Q2.3. Record the model fit, degrees of freedom, and number of parameters:

	Values
Fit -2LL	
df	
parameters	

Q2.4. In plain language, what do the age and sex results mean?
(e.g. for each additional year of age, we would be an XXX SD change in the DV).

Q2.5. Record the estimated standardised A, C, E variance components and their lower and upper 95% confidence intervals:

	lower 95% CI	Estimate	upper 95% CI
A			
C			
E			

Q2.6. Run the section of the script to obtain power.

What power did we have for A and for C?

	Power
A	
C	

Q2.7. Open **01_ACEsib_contin.R**

If you have some prior experience, you might like to try the **challenge_01_ACEsib_contin.R** script. This script has ? noting places that require you to edit the script.

Because we are using many of the same object names across our scripts, at the top of each script there is a line:

```
rm(list=ls())
```

This will clear your workspace and ensure that if there is an error or a problem with the current script when creating an object, then an old object of the same name will not be used in the current model.

Run the model and record the model fit:

	Value
Fit -2LL	
df	
parameters	

Q2.8. Record the estimated variance components:

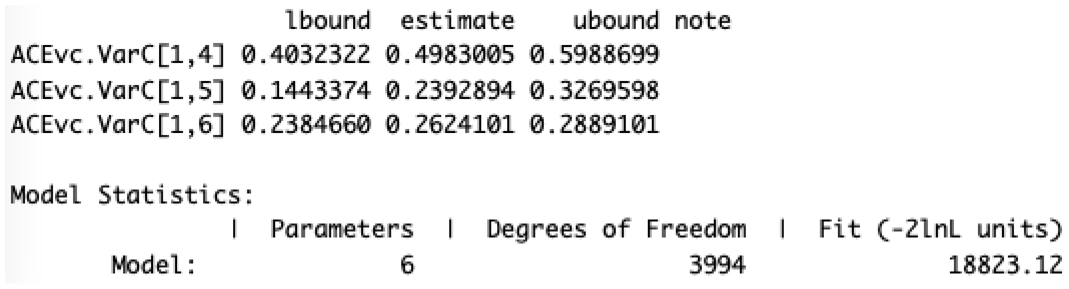
	lower 95% CI	Estimate	upper 95% CI
A			
C			
E			

Q2.9. What power did we have for A and C?

	Power
A	
C	

Q2.10. How do these estimates compare to the twin-only model?

ACE twin pairs



```
                    lbound  estimate    ubound note
ACEvc.VarC[1,4] 0.4032322 0.4983005 0.5988699
ACEvc.VarC[1,5] 0.1443374 0.2392894 0.3269598
ACEvc.VarC[1,6] 0.2384660 0.2624101 0.2889101

Model Statistics:
              |  Parameters  |  Degrees of Freedom  |  Fit (-2lnL units)
      Model:             6                   3994           18823.12
```

Power: A = 1 & C = 0.9988135

Q2.11. So far, to create the variance/covariance matrices we first created the A, C, E components, then used them to create a variance object and a covariance object for MZ and DZ separately, and then put the variance and covariance objects together.

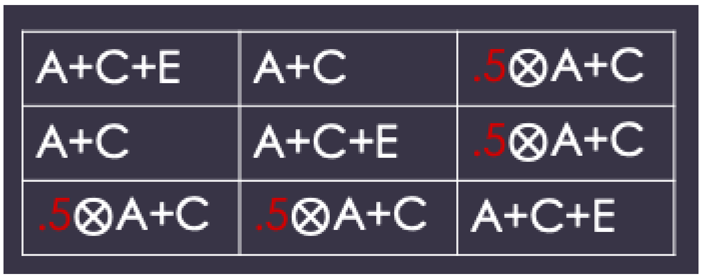
```
# Create Matrices for Variance Components
covA      <- mxMatrix( type="Symm", nrow=nv, ncol=nv, free=TRUE,
values=sVa, label="VA11", name="VA" )
covC      <- mxMatrix( type="Symm", nrow=nv, ncol=nv, free=TRUE,
values=sVc, label="VC11", name="VC" )
covE      <- mxMatrix( type="Symm", nrow=nv, ncol=nv, free=TRUE,
values=sVe, label="VE11", 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, cDZ),
                                           cbind(t(cMZ), V,
cDZ),
                                           cbind(t(cDZ), t(cDZ),
V)), name="expCovMZ" )

expCovDZ  <- mxAlgebra( expression= rbind( cbind(V, cDZ, cDZ),
                                           cbind(t(cDZ), V,
cDZ),
                                           cbind(t(cDZ), t(cDZ),
V)), name="expCovDZ" )
```

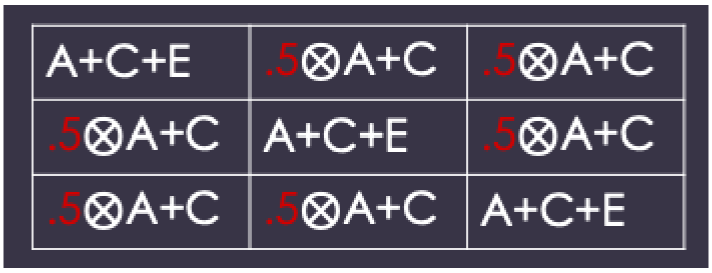
Imaging if you were creating one of these expected variance/covariance matrices to include many siblings. It could become cumbersome. There are other ways that we can build the final expected variance/covariance matrix for our model.

The final expected variance/covariance for an MZ pair with a sibling can be represented:



A+C+E	A+C	.5⊗A+C
A+C	A+C+E	.5⊗A+C
.5⊗A+C	.5⊗A+C	A+C+E

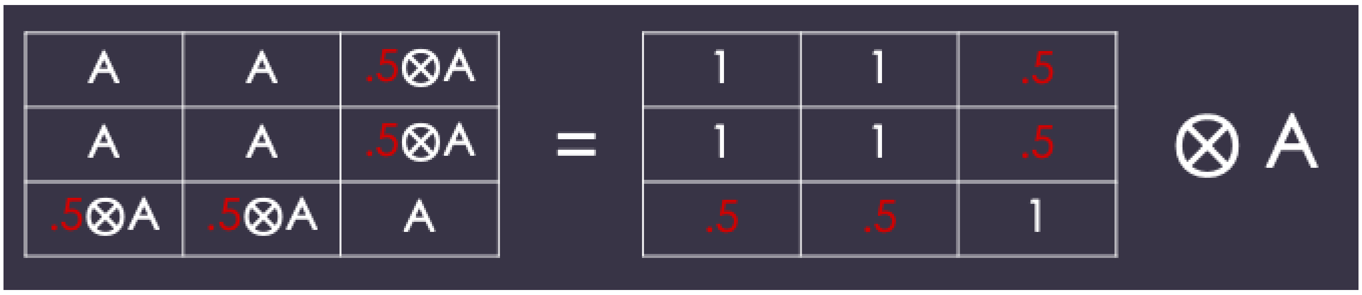
And for a DZ pair and sibling as:



A+C+E	.5⊗A+C	.5⊗A+C
.5⊗A+C	A+C+E	.5⊗A+C
.5⊗A+C	.5⊗A+C	A+C+E

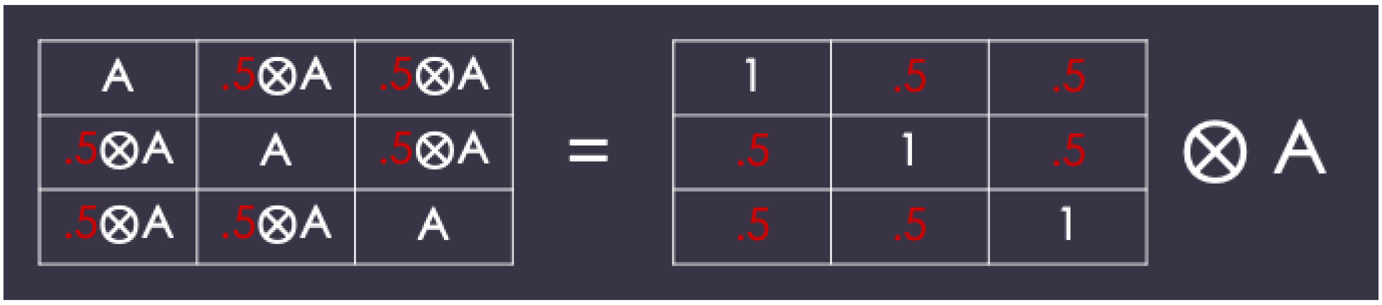
An alternative way to parameterise this is to create a matrix that represents the expected relationships for each A, C, and E component, then use a kronecker product (check 'Matrix Multiplication Sheet' for details on the types of matrix multiplication) to multiply these relationship matrices with each of the A, C, and E components.

For MZ the A component:



$$\begin{array}{|c|c|c|}\hline A & A & .5\otimes A \\ \hline A & A & .5\otimes A \\ \hline .5\otimes A & .5\otimes A & A \\ \hline \end{array} = \begin{array}{|c|c|c|}\hline 1 & 1 & .5 \\ \hline 1 & 1 & .5 \\ \hline .5 & .5 & 1 \\ \hline \end{array} \otimes A$$

For DZ the A component:



$$\begin{array}{|c|c|c|}\hline A & .5\otimes A & .5\otimes A \\ \hline .5\otimes A & A & .5\otimes A \\ \hline .5\otimes A & .5\otimes A & A \\ \hline \end{array} = \begin{array}{|c|c|c|}\hline 1 & .5 & .5 \\ \hline .5 & 1 & .5 \\ \hline .5 & .5 & 1 \\ \hline \end{array} \otimes A$$

The C component for both MZ and DZ:

$$\begin{bmatrix} C & C & C \\ C & C & C \\ C & C & C \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \otimes C$$

The E component for both MZ and DZ:

$$\begin{bmatrix} E & 0 & 0 \\ 0 & E & 0 \\ 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \otimes E$$

These A, C, E variance-covariance matrices are the same dimensions and can be simply summed together.

In OpenMx the code for the relationship matrices looks like:

```
relMZ      <- mxMatrix( type="Symm", nrow=nt, ncol=nt,
free=FALSE, values=c(1,1,.5,1,.5,1),  name="rAmz" )
relDZ      <- mxMatrix( type="Symm", nrow=nt, ncol=nt,
free=FALSE, values=c(1,.5,.5,1,.5,1), name="rAdz" )
relC       <- mxMatrix( type="Unit", nrow=nt, ncol=nt,
free=FALSE, name="rC" )
relE       <- mxMatrix( type="Iden", nrow=nt, ncol=nt,
free=FALSE, name="rE" )
```

We can multiply these relationship matrices with the A, C, E components and sum them together in a single step:

```
expCovMZ  <- mxAlgebra( expression= rAmz%x%VA + rC%x%VC +
rE%x%VE, name="expCovMZ" )
expCovDZ  <- mxAlgebra( expression= rAdz%x%VA + rC%x%VC +
rE%x%VE, name="expCovDZ" )
```

Q2.12. Open **02_ACEsib_alt_contin.R**
Run the code up to when the model is built (~line 95).

Before you fit the model, have a look in the relMZ and relDZ objects and use mxEval to

look at the expected variance/covariance matrices:
This function allows us to check that our matrices have been set up as we expect prior to running a model.

```
mxEval(expCovMZ, modelMZ, compute=TRUE)
mxEval(expCovDZ, modelDZ, compute=TRUE)
```

The first argument is the name of an object that is created using mxAlgebra. The second is the name of the model that it belongs to. The third asks for the algebra to be calculated.

What are the values in this the expected MZ variance/covariance matrix before the model is run? (NOTE: only the lower diagonal is needed, the matrix is symmetric)

	T1	T2	Sib
T1			
T2			
Sib			

Q2.13. What are the values in the expected DZ variance/covariance matrix before the model is run?

	T1	T2	Sib
T1			
T2			
Sib			

Q2.14. Run the model.

What are the values after estimation for the MZ variance/covariance matrix?

	T1	T2	Sib
T1			
T2			
Sib			

Q2.15. Would you like a hint on how to extract this matrix from the output?

○ Yes

Q2.16. Hint:
fitACE$output$algebras$MZ.expCovMZ
fitACE$output$algebras$DZ.expCovDZ

Q2.17. What are the values after estimation for the DZ variance/covariance matrix?

Notice similarities and differences between the estimated values and the start values.

	T1	T2	Sib
T1			
T2			
Sib			

Q2.18. Record the model fit:

	Values
Fit -2LL	
df	
parameters	

Q2.19. Record the estimated variance components:

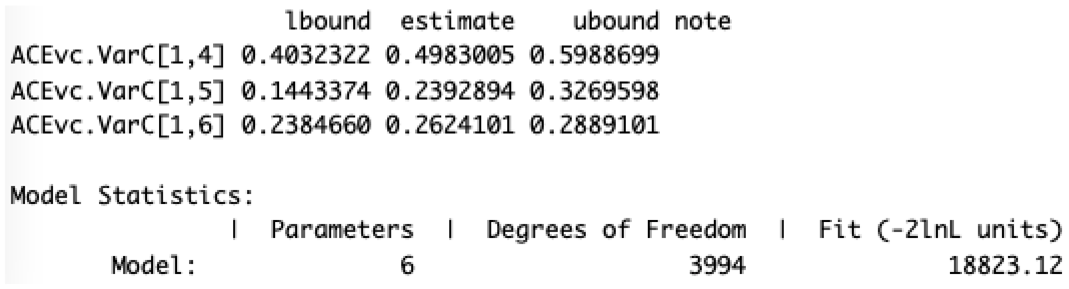
	lower 95% CI	Estimate	upper 95% CI
A			
C			
E			

Q2.20. Record the power:

	Power
A	
C	

Q2.21. How do these estimates compare to the previous models?

ACE twin pairs

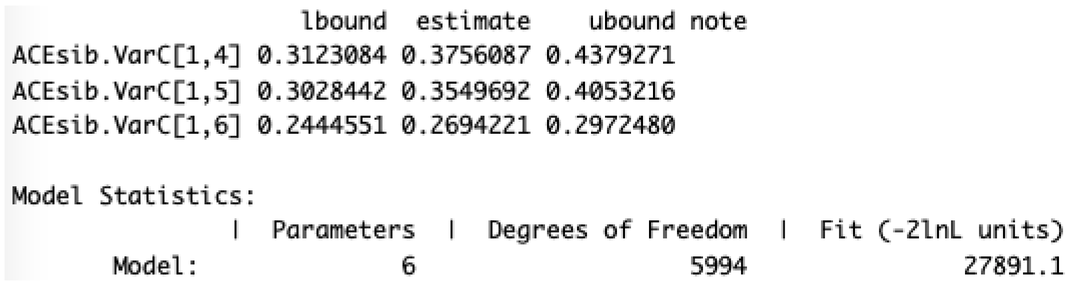


```
                    lbound  estimate    ubound note
ACEvc.VarC[1,4] 0.4032322 0.4983005 0.5988699
ACEvc.VarC[1,5] 0.1443374 0.2392894 0.3269598
ACEvc.VarC[1,6] 0.2384660 0.2624101 0.2889101

Model Statistics:
               |  Parameters  |  Degrees of Freedom  |  Fit (-2lnL units)
      Model:             6                 3994              18823.12
```

Power: A = 1 & C = 0.9988135

Twins and sibs



```
                    lbound  estimate    ubound note
ACEsib.VarC[1,4] 0.3123084 0.3756087 0.4379271
ACEsib.VarC[1,5] 0.3028442 0.3549692 0.4053216
ACEsib.VarC[1,6] 0.2444551 0.2694221 0.2972480

Model Statistics:
               |  Parameters  |  Degrees of Freedom  |  Fit (-2lnL units)
      Model:             6                 5994              27891.1
```

Power: A = 1 & C = 1

Q2.22. Up until now, we have created the final model from two separate models, one for MZ and one for DZ. These models differ only in the coefficient of relatedness that is incorporated into the A part of the expected variance/covariance matrix.

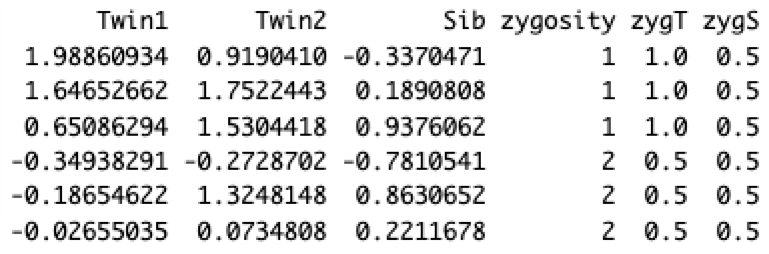
This has been a hard-coded matrix that is different for MZ and DZ:

```
relMZ      <- mxMatrix( type="Symm", nrow=nt, ncol=nt,
free=FALSE, values=c(1,1,.5,1,.5,1),  name="rAmz" )
relDZ      <- mxMatrix( type="Symm", nrow=nt, ncol=nt,
free=FALSE, values=c(1,.5,.5,1,.5,1), name="rAdz" )
```

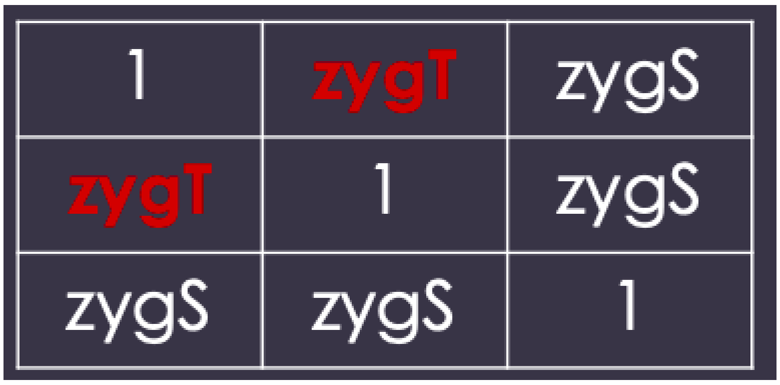
Alternatively, we can use a definition variable to hold the coefficient of relatedness for each pair of individuals and use that data in a relationship matrix that could be used for all twin pairs:

```
relA      <- mxMatrix( type="Stand", nrow=nt, ncol=nt,
free=FALSE, labels=c("data.zygT","data.zygS","data.zygS"),
name="rA" )
```

zygT is the coefficient of relationship between Twin1 and Twin2. For MZ pairs this will equal 1, for DZ pairs this will equal 0.5.
zygS is the coefficient of relationship between Twin1/Twin2 and their Sibling. This will equal 0.5 for all pairs.



```
       Twin1      Twin2        Sib zygosity zygT zygS
  1.98860934  0.9190410 -0.3370471        1  1.0  0.5
  1.64652662  1.7522443  0.1890808        1  1.0  0.5
  0.65086294  1.5304418  0.9376062        1  1.0  0.5
 -0.34938291 -0.2728702 -0.7810541        2  0.5  0.5
 -0.18654622  1.3248148  0.8630652        2  0.5  0.5
 -0.02655035  0.0734808  0.2211678        2  0.5  0.5
```



1	zygT	zygS
zygT	1	zygS
zygS	zygS	1

Q2.23. Open **03_ACEzygdef_contin.R** and run the script up to building the final model (~line 90).

Have a look in the relA matrix and use mxEval to have a look in the expCov matrix.

Do you want a hint for how to use mxEval?

○ Yes

Q2.24. HINT:
`mxEval(expCov,modelACE,compute=T)`

Q2.25. Is the expCov matrix what you would expect for an MZ pair or a DZ pair?

○ MZ

○ DZ

Q2.26. When using mxEval to check matrices, for definition variables it will use the first line of data for values.

Q2.27. Run the model.

When running the model you might have received a warning with a status GREEN and a code 1.

If we had a RED status we would need to investigate, or if our estimates were nonsensical. Some of the ways we can troubleshoot a RED status are covered in the binary part of this tutorial.

But for now, we can keep going.

Record the model fit:

	Values
Fit -2LL	
df	
parameters	

Q2.28. Record the estimated variance components:

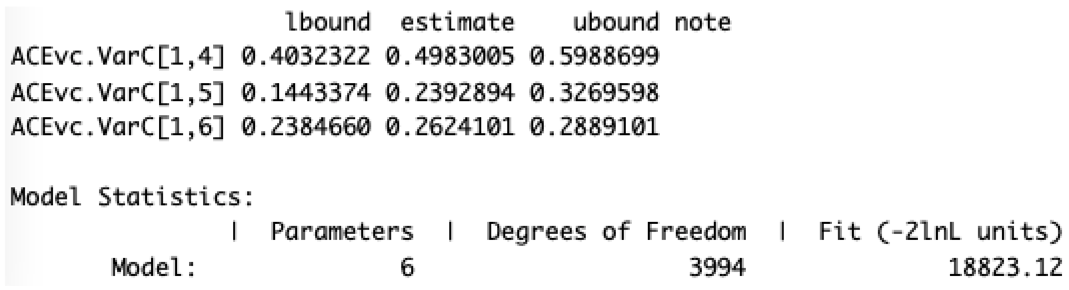
	lower 95% CI	Estimate	upper 95% CI
A			
C			
E			

Q2.29. Record the power:

	Power
A	
C	

Q2.30. How do these estimates compare to the previous models?

ACE twin pairs

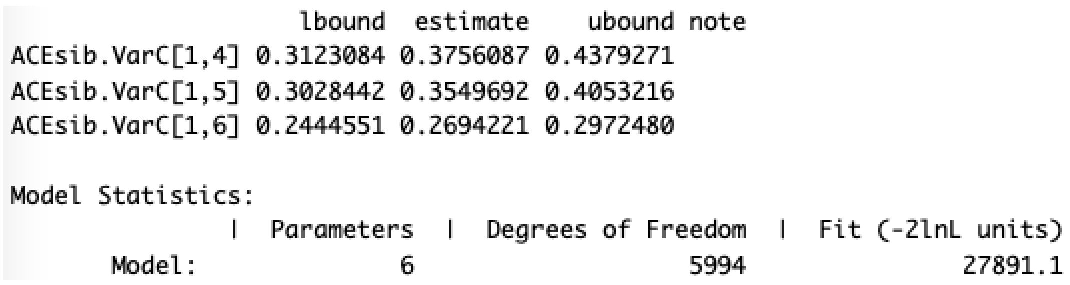


```
                  lbound  estimate    ubound note
ACEvc.VarC[1,4] 0.4032322 0.4983005 0.5988699
ACEvc.VarC[1,5] 0.1443374 0.2392894 0.3269598
ACEvc.VarC[1,6] 0.2384660 0.2624101 0.2889101

Model Statistics:
               |  Parameters  |  Degrees of Freedom  |  Fit (-2lnL units)
       Model:              6                   3994             18823.12
```

Power: A = 1 & C = 0.9988135

Twins and sibs

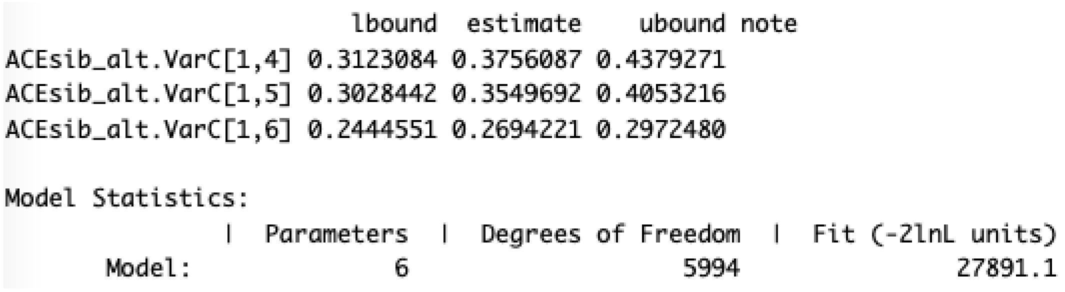


```
                   lbound  estimate    ubound note
ACEsib.VarC[1,4] 0.3123084 0.3756087 0.4379271
ACEsib.VarC[1,5] 0.3028442 0.3549692 0.4053216
ACEsib.VarC[1,6] 0.2444551 0.2694221 0.2972480

Model Statistics:
               |  Parameters  |  Degrees of Freedom  |  Fit (-2lnL units)
      Model:              6                   5994              27891.1
```

Power: A = 1 & C = 1

Twins and sibs alternate parameterisation



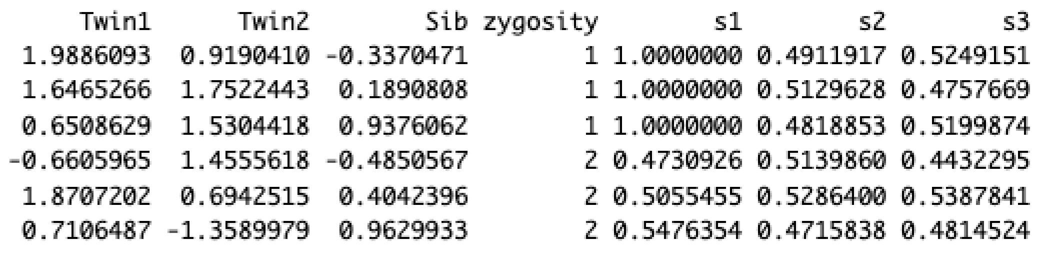
```
                       lbound  estimate    ubound note
ACEsib_alt.VarC[1,4] 0.3123084 0.3756087 0.4379271
ACEsib_alt.VarC[1,5] 0.3028442 0.3549692 0.4053216
ACEsib_alt.VarC[1,6] 0.2444551 0.2694221 0.2972480

Model Statistics:
               |  Parameters  |  Degrees of Freedom  |  Fit (-2lnL units)
      Model:              6                   5994              27891.1
```

Power: A = 1 & C = 1

Q2.31. So far we have used the theoretical coefficient of relatedness based on pedigree information between the individuals in the family. If we have measured genetic relationships between pairs of individuals, then we can use it as a definition variables in these models.

Open **04_ACE_grm_relatedness_contin.R**
Load the data and have a look at the columns.



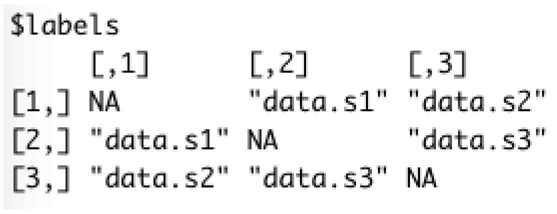
```
     Twin1      Twin2        Sib zygosity        s1        s2        s3
 1.9886093  0.9190410 -0.3370471        1 1.0000000 0.4911917 0.5249151
 1.6465266  1.7522443  0.1890808        1 1.0000000 0.5129628 0.4757669
 0.6508629  1.5304418  0.9376062        1 1.0000000 0.4818853 0.5199874
-0.6605965  1.4555618 -0.4850567        2 0.4730926 0.5139860 0.4432295
 1.8707202  0.6942515  0.4042396        2 0.5055455 0.5286400 0.5387841
 0.7106487 -1.3589979  0.9629933        2 0.5476354 0.4715838 0.4814524
```

s1 = the genetic relatedness coefficient between Twin1 and Twin2
s2 = the genetic relatedness coefficient between Twin1 and Sib
s3 = the genetic relatedness coefficient between Twin2 and Sib

Have a look at the distribution of these relatedness variables.
We can still use a threshold on the relatedness between Twin1 and Twin2 to check out the correlations in our data.

Have a look at the relA matrix, which pulls in the relatedness data as a definition variable:

```
relA      <- mxMatrix( type="Stand", nrow=nt, ncol=nt,
free=FALSE, labels=c("data.s1","data.s2","data.s3"), name="rA"
)
```



```
$labels
     [,1]      [,2]      [,3]
[1,] NA        "data.s1" "data.s2"
[2,] "data.s1" NA        "data.s3"
[3,] "data.s2" "data.s3" NA
```

Run the rest of the script and fit the model.

Record the model fit:

	Values
Fit -2LL	
df	
parameters	

Q2.32. Record the estimated standardised variance components:

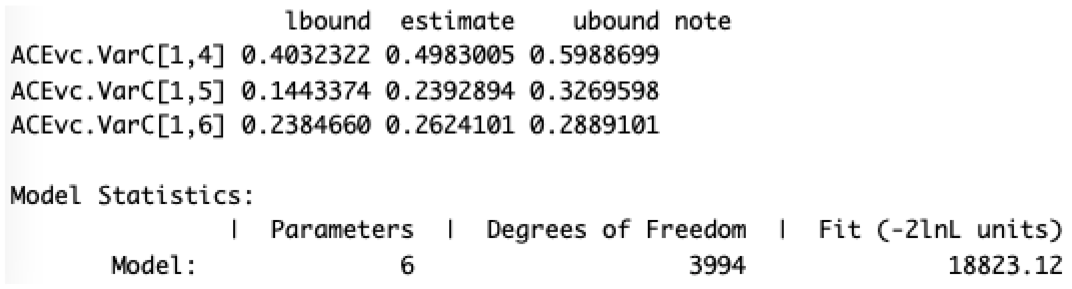
	lower 95% CI	Estimate	upper 95% CI
A			
C			
E			

Q2.33. Record the power:

	Power
A	
C	

Q2.34. How do these estimates compare to the previous models?

ACE twin pairs

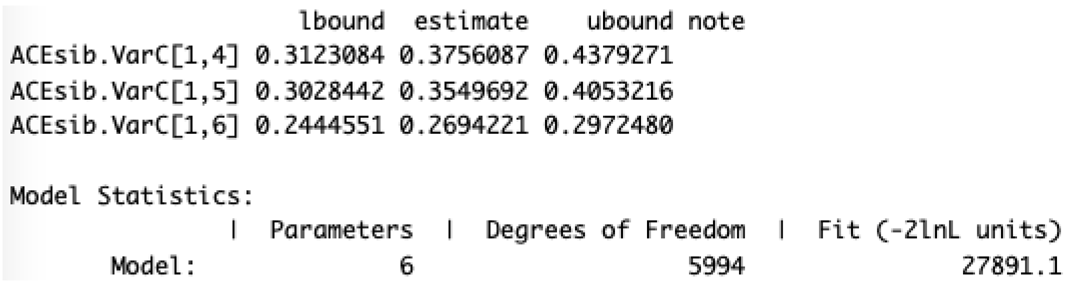


```
                  lbound  estimate    ubound note
ACEvc.VarC[1,4] 0.4032322 0.4983005 0.5988699
ACEvc.VarC[1,5] 0.1443374 0.2392894 0.3269598
ACEvc.VarC[1,6] 0.2384660 0.2624101 0.2889101

Model Statistics:
               |  Parameters  |  Degrees of Freedom  |  Fit (-2lnL units)
      Model:             6                   3994             18823.12
```

Power: A = 1 & C = 0.9988135

Twins and sibs

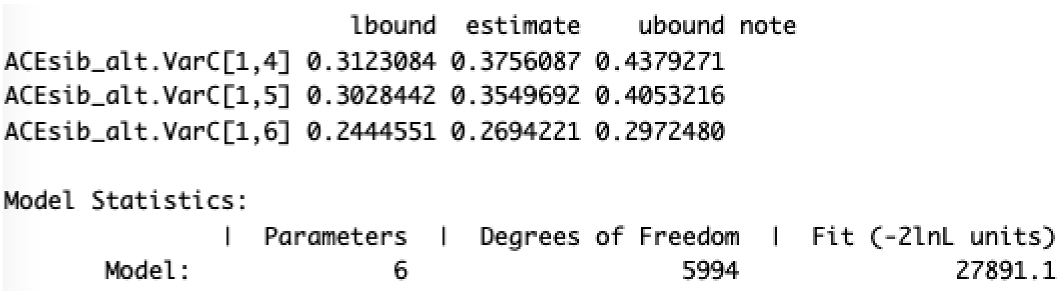


```
                  lbound  estimate    ubound note
ACEsib.VarC[1,4] 0.3123084 0.3756087 0.4379271
ACEsib.VarC[1,5] 0.3028442 0.3549692 0.4053216
ACEsib.VarC[1,6] 0.2444551 0.2694221 0.2972480

Model Statistics:
               |  Parameters  |  Degrees of Freedom  |  Fit (-2lnL units)
      Model:             6                   5994             27891.1
```

Power: A = 1 & C = 1

Twins and sibs alternate parameterisation

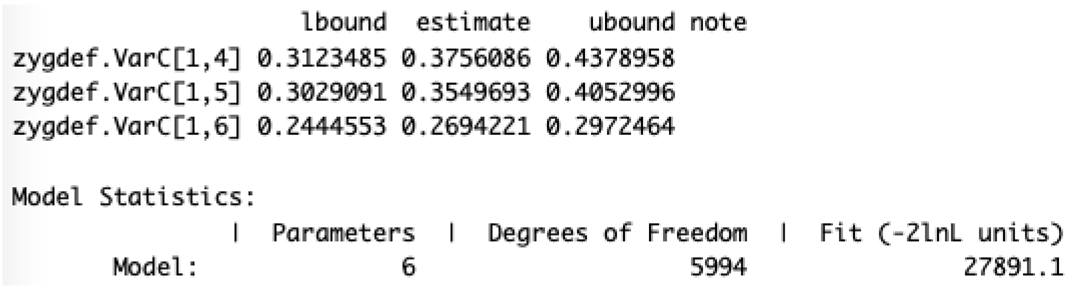


```
                      lbound  estimate    ubound note
ACEsib_alt.VarC[1,4] 0.3123084 0.3756087 0.4379271
ACEsib_alt.VarC[1,5] 0.3028442 0.3549692 0.4053216
ACEsib_alt.VarC[1,6] 0.2444551 0.2694221 0.2972480

Model Statistics:
               |  Parameters  |  Degrees of Freedom  |  Fit (-2lnL units)
      Model:             6                   5994             27891.1
```

Power: A = 1 & C = 1

Twins and sibs with zygosity as a definition variable



```
                  lbound  estimate    ubound note
zygdef.VarC[1,4] 0.3123485 0.3756086 0.4378958
zygdef.VarC[1,5] 0.3029091 0.3549693 0.4052996
zygdef.VarC[1,6] 0.2444553 0.2694221 0.2972464

Model Statistics:
               |  Parameters  |  Degrees of Freedom  |  Fit (-2lnL units)
      Model:             6                   5994             27891.1
```

Power: A = 1 & C = 1

Q2.35. Now that we have a model that uses measured genetic variation, the model can be identified without MZ pairs, but there is a cost!

Open **05_ACE_GRM_relatednessDZonly_contin.R**

This script is set up to read in another dataset that is much larger but used the same model specifications in the simulation as the previous dataset.

Run the model.

Record the model fit:

	Values
Fit -2LL	
df	
parameters	

Q2.36. Record the estimated standardised variance components:

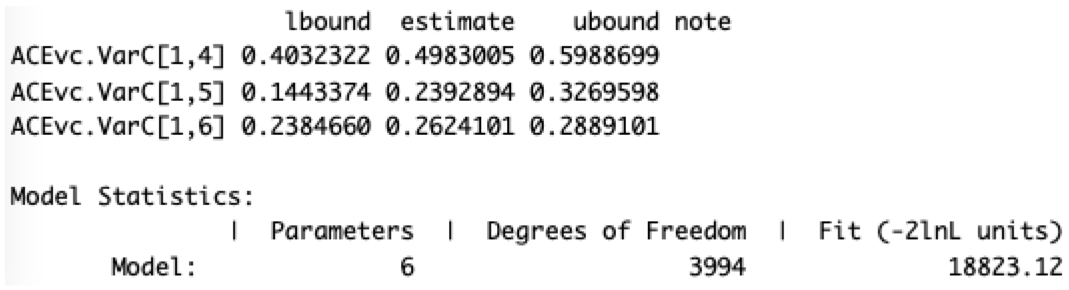
	lower 95% CI	Estimate	upper 95% CI
A			
C			
E			

Q2.37. Record the power:

	Power
A	
C	

Q2.38. How do these estimates compare to the previous models?

ACE twin pairs

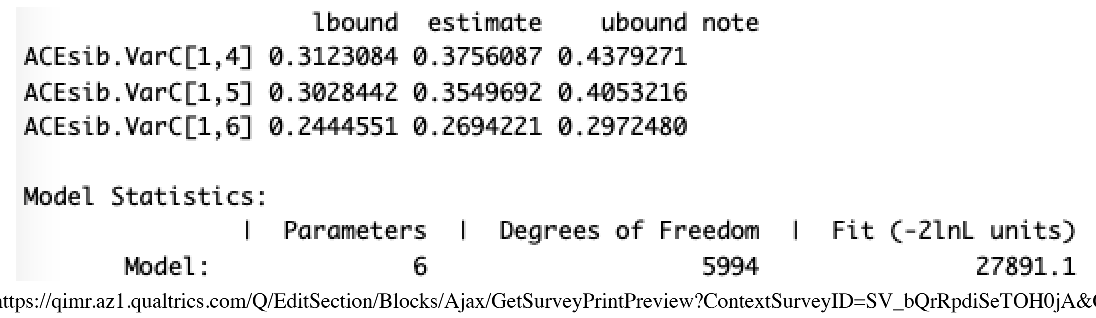


```
                 lbound  estimate    ubound note
ACEvc.VarC[1,4] 0.4032322 0.4983005 0.5988699
ACEvc.VarC[1,5] 0.1443374 0.2392894 0.3269598
ACEvc.VarC[1,6] 0.2384660 0.2624101 0.2889101

Model Statistics:
               |  Parameters  |  Degrees of Freedom  |  Fit (-2lnL units)
       Model:             6                   3994             18823.12
```

Power: A = 1 & C = 0.9988135

Twins and sibs

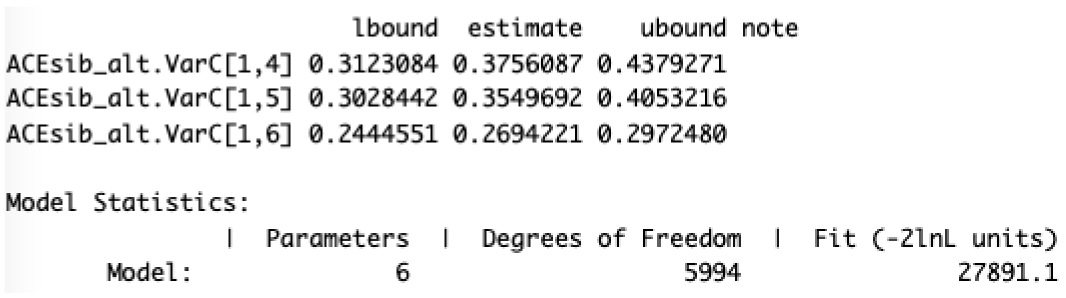


```
                  lbound  estimate    ubound note
ACEsib.VarC[1,4] 0.3123084 0.3756087 0.4379271
ACEsib.VarC[1,5] 0.3028442 0.3549692 0.4053216
ACEsib.VarC[1,6] 0.2444551 0.2694221 0.2972480

Model Statistics:
               |  Parameters  |  Degrees of Freedom  |  Fit (-2lnL units)
       Model:             6                   5994              27891.1
```

Power: A = 1 & C = 1

Twins and sibs alternate parameterisation

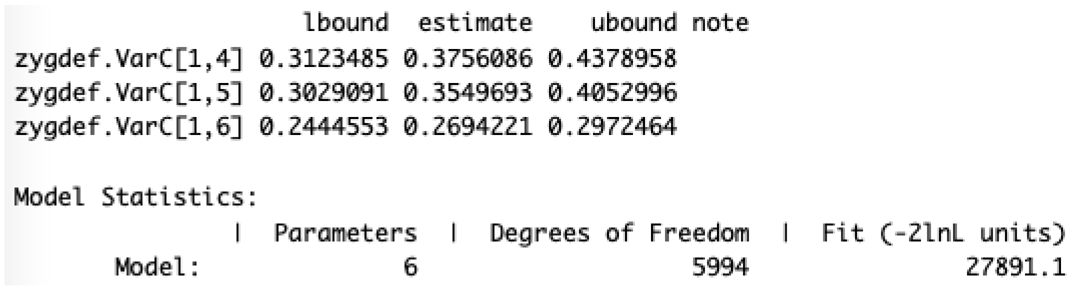


```
                         lbound  estimate    ubound note
ACEsib_alt.VarC[1,4] 0.3123084 0.3756087 0.4379271
ACEsib_alt.VarC[1,5] 0.3028442 0.3549692 0.4053216
ACEsib_alt.VarC[1,6] 0.2444551 0.2694221 0.2972480

Model Statistics:
               |  Parameters  |  Degrees of Freedom  |  Fit (-2lnL units)
      Model:              6                   5994              27891.1
```

Power: A = 1 & C = 1

Twins and sibs with zygosity as a definition variable

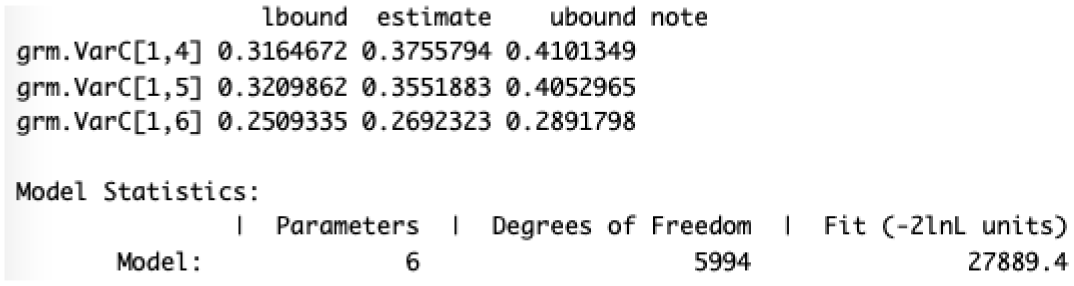


```
                     lbound  estimate    ubound note
 zygdef.VarC[1,4] 0.3123485 0.3756086 0.4378958
 zygdef.VarC[1,5] 0.3029091 0.3549693 0.4052996
 zygdef.VarC[1,6] 0.2444553 0.2694221 0.2972464

 Model Statistics:
               |  Parameters  |  Degrees of Freedom  |  Fit (-2lnL units)
      Model:              6                   5994              27891.1
```

Power: A = 1 & C = 1

Twins and sibs with measured genetic relationship



```
                  lbound  estimate    ubound note
 grm.VarC[1,4] 0.3164672 0.3755794 0.4101349
 grm.VarC[1,5] 0.3209862 0.3551883 0.4052965
 grm.VarC[1,6] 0.2509335 0.2692323 0.2891798

 Model Statistics:
               |  Parameters  |  Degrees of Freedom  |  Fit (-2lnL units)
      Model:              6                   5994              27889.4
```

Power: A = 1 & C = 1

Q2.39. We've now shown you five different ways of fitting a twin model.
00_ACEvc_contin.R & 01_ACEsib_contin.R
02_ACEsib_alt_contin.R
03_ACEzygdef_contin.R
04_ACEgrm_relatedness_contin.R
05_ACEgrm_relatedness_DZonly__contin.R

Under what circumstances might one be a more appropriate choice than another? Discuss amongst your group.

Q2.40. This is the end of today's practical. If you click "next" you will exit the practical.

Q2.41. If you click "next" you will exit the practical.

Binary

Q3.1. Open **00_ACEvc_binary.R**

This script is a univariate ACE script for binary data. It has many similarities to the one that you worked through in the Day 1 Practical, but there are some differences that we will highlight.

For the sex variable in the data, females are coded as 0 and males are coded as 1. Note. The data is simulated. The phenotype can be whatever you want it to be.

Note. The data is simulated.

At the beginning of the script, we convert a continuous variable into a binary one that will be used throughout the script.

Because we no longer have observed variance, we will constrain our model to have a fixed variance of 1 and a mean of zero. Thus mapping onto the liability threshold model.

```
dfBin <- df
dfBin$Twin1 <- ifelse(dfBin$Twin1 > 0, 1, 0)
dfBin$Twin2 <- ifelse(dfBin$Twin2 > 0, 1, 0)
dfBin$Sib   <- ifelse(dfBin$Sib > 0, 1, 0)
```

Check the frequencies:
```
table(dfBin$Twin1)
table(dfBin$Twin2)
table(dfBin$Sib)
```

Once we have created our two groups, we will use mxFactor to ensure that the data are encoded an ordered factor.
```
dfBin$Twin1 <- mxFactor(dfBin$Twin1, levels = 0:1)
dfBin$Twin2 <- mxFactor(dfBin$Twin2, levels = 0:1)
```

```
dfBin$Sib   <- mxFactor(dfBin$Sib, levels = 0:1)
```

Q3.2. Run the script to the bottom of the section that creates algebra for expected means matrices (~line 67).

Look at these two lines:

```
defSex        <- mxMatrix( type="Full", nrow=1, ncol=nt,
free=FALSE, labels=c("data.sex1","data.sex2"), name="Sex" )
defAge        <- mxMatrix( type="Full", nrow=1, ncol=nt,
free=FALSE, labels=c("data.age1","data.age2"), name="Age" )
```

Putting **`data.`** in the label tells OpenMx that this is a definition variable and the values will be updated for each case in the data set.

There can be no missing data on a definition variable or the model will not run. If your data set is incomplete (i.e. you have incomplete sets of twin pairs) you might need to recode any missing values with a dummy code (i.e. the mean of the variable). Cases that are missing data on the trait are not used fitting the model, so what value you use to recode a missing definition value will not matter. However, if there is trait data on that case, then the recoded data will be treated as a genuine value.

The intercept (or mean) is no longer free to be estimated. It is fixed at zero (remember we are mapping the data onto a standard normal distribution):

```
intercept     <- mxMatrix( type="Full", nrow=1, ncol=ntv,
free=FALSE, values=0, labels="interC", name="intercept" )
```

Instead the thresholds are estimated (as this is a binary trait there is only one threshold)

```
expThr        <- mxMatrix(type="Full", nrow=nTH, ncol=ntv, free
= TRUE, values = ThrVals, labels =  paste("th", 1:nTH, sep =
""), name = "expThr")
```

We can either model covariate effects on the mean or on the thresholds. We have modelled the effects on the fixed mean.

```
expMean       <- mxAlgebra( expression = intercept + Sex%x%bS +
Age%x%bA , name="expMean" )
```

Run the next section of script that create the matrices to hold the variance components. (~line 73)

Here we include a matrix that will be used to constrain the total variance to equal 1.

```
cons <- mxConstraint(VA+VC+VE ==1, name = "cons")
```

Run the script to the bottom of the section that creates model objects for multiple groups (~line 98).

Here we have created objects that each have a list of other objects:

```
defs      <- list( defAge, defSex )
pars      <- list( intercept, betaS, betaA, covA, covC, covE, 
covP )
```

The definition variables have been split out from the rest of the list of objects. This is because we will want to put the objects for definition variables into the MZ and DZ submodels, because definition variables need to go in an mxModel that includes mxData. We have the second list of objects because it includes objects that may be used in each level of the model.

Run the script to create the final model (~line 109).

We have created an object to extract the unstandardised and standardised variance components.

```
estVC     <- mxAlgebra( 
expression=cbind(VA,VC,VE,VA/V,VC/V,VE/V), name="VarC", 
dimnames=list(rowVC,colVC) )
```

And can request confidence intervals on the elements in that object. Here we request them on the standardised variance components.

```
ciACE     <- mxCI( "VarC[1,4:6]" )
```

Then put together the final model, which includes the objects for CIs and the constraint on the variance.

```
modelACE  <- mxModel( "ACEvc", modelMZ, modelDZ, multi, pars, 
estVC, ciACE, cons )
```

Run the script to fit the model.

Q3.3. Record the model fit, degrees of freedom, and number of parameters:

Values

	Values
Fit -2LL	
df	
parameters	

Q3.4. In plain language, what do the threshold, age, and sex results mean?
(e.g. for each additional year of age, we would be an XXX SD change in the liability for the DV).

Q3.5. Are males or females more likely to be cases?

- ○ Males
- ○ Females

Q3.6. Are older or younger people more likely to be cases?

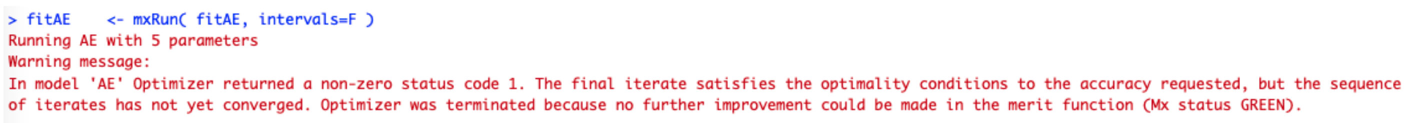
- ○ Older
- ○ Younger

Q3.7. Record the estimated A, C, E variance components and their lower and upper 95% confidence intervals:

	lower 95% CI	Estimate	upper 95% CI
A			
C			
E			

Q3.8. Run the section of the script to obtain power.

When running this section, one of the nested models might give you a GREEN warning:



```
> fitAE     <- mxRun( fitAE, intervals=F )
Running AE with 5 parameters
Warning message:
In model 'AE' Optimizer returned a non-zero status code 1. The final iterate satisfies the optimality conditions to the accuracy requested, but the sequence
of iterates has not yet converged. Optimizer was terminated because no further improvement could be made in the merit function (Mx status GREEN).
```

You can continue with a warning like this. If you were to obtain a code with RED status that would mean you should do some additional troubleshooting. We will cover some of those as this tutorial continues...

What power did we have for A and for C?

	Power
A	
C	

Q3.9. Open **01_ACEsib_binary.R**

If you have some prior experience, you might like to try the **challenge_01_ACEsib_binary.R** script. This script has ? noting places that require you to edit the script.

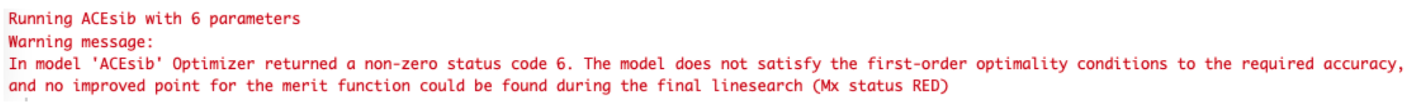
Because we are using many of the same object names across our scripts, at the top of each script there is a line:

```
rm(list=ls())
```

This will ensure that if there is an error or a problem with the current script when creating an object, then an old object of the same name will not be used in the current model.

Run the model.

You might receive a RED warning:



```
Running ACEsib with 6 parameters
Warning message:
In model 'ACEsib' Optimizer returned a non-zero status code 6. The model does not satisfy the first-order optimality conditions to the required accuracy,
and no improved point for the merit function could be found during the final linesearch (Mx status RED)
```

This status code 6 means that the optimiser did not find a solution that was sufficiently precise. This can happen for a lot of reasons. It might be that there's a problem with the model, or maybe it ran out of iterations or that it could not make adjustments that improved the fit. We have to do some troubleshooting!

Some options are:

1. Check the model is identified.
2. Check and adjust start values.

3. Re-run from the last solution. We can do this by using the function mxTryHard() or mxTryHardOrdinal() instead of mxRun(). Both of these make multiple attempts to fit a model and will stop either when a suitable solution is found or when the limit of attempts has been reached (the default is 10 additional attempts).
4. Change the optimiser. There are several optimisers that you can use to fit the model i.e. NPSOL, CSOLNP, SLSQP.

What we try might depend on the type of error or warning that we get. Importantly, sometimes we might not have a warning but we will have negative variances or nonsensical values. These situations are also important to troubleshoot.

For now, let's rerun with CSOLNP as the optimiser.

```
mxOption(NULL,"Default optimizer", "CSOLNP")
fitACE    <- mxRun( modelACE, intervals=T )
```

Record the model fit:

	Value
Fit -2LL	
df	
parameters	

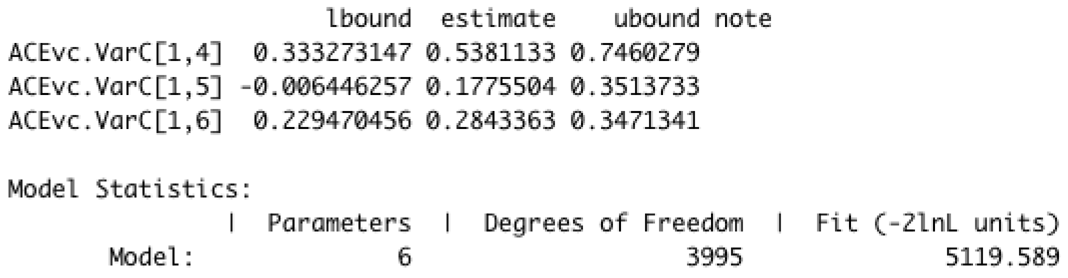
Q3.10. Record the estimated variance components:

	lower 95% CI	Estimate	upper 95% CI
A			
C			
E			

Q3.11. What power did we have for A and C?

	Power
A	
C	

Q3.12. How do these estimates compare to the ACEvc model that had only the twin pairs?



```
                    lbound  estimate    ubound note
ACEvc.VarC[1,4]  0.333273147 0.5381133 0.7460279
ACEvc.VarC[1,5] -0.006446257 0.1775504 0.3513733
ACEvc.VarC[1,6]  0.229470456 0.2843363 0.3471341

Model Statistics:
              |  Parameters  |  Degrees of Freedom  |  Fit (-2lnL units)
       Model:             6                   3995             5119.589
```

Power: A = 0.9997855 & C = 0.5993366

Q3.13. So far, to create the variance/covariance matrices we first created the A, C, E components, then used them to create a variance object and a covariance object for MZ and DZ separately, and then put the variance and covariance objects together.

# Create Matrices for Variance Components

```
covA      <- mxMatrix( type="Symm", nrow=nv, ncol=nv, free=TRUE,
values=sVa, label="VA11", name="VA" )
covC      <- mxMatrix( type="Symm", nrow=nv, ncol=nv, free=TRUE,
values=sVc, label="VC11", name="VC" )
covE      <- mxMatrix( type="Symm", nrow=nv, ncol=nv, free=TRUE,
values=sVe, label="VE11", name="VE" )
```

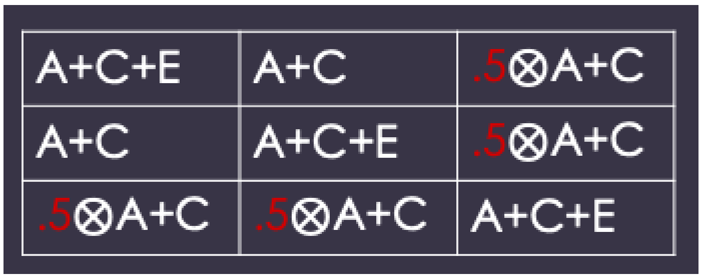
# Create Algebra for expected Variance/Covariance Matrices in MZ & DZ twins

```
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, cDZ),
                                           cbind(t(cMZ), V,
cDZ),
                                           cbind(t(cDZ), t(cDZ),
V)), name="expCovMZ" )

expCovDZ  <- mxAlgebra( expression= rbind( cbind(V, cDZ, cDZ),
                                           cbind(t(cDZ), V,
cDZ),
                                           cbind(t(cDZ), t(cDZ),
V)), name="expCovDZ" )
```

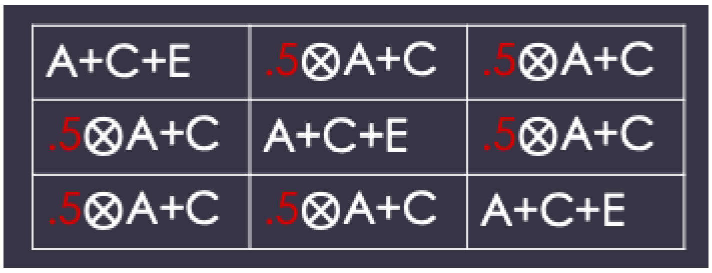
Imagine if you were creating one of these expected variance/covariance matrices to include many siblings. It could become cumbersome. There are other ways that we can build the final expected variance/covariance matrix for our model.

The final expected variance/covariance for an MZ pair with a sibling can be represented:



A+C+E	A+C	.5⊗A+C
A+C	A+C+E	.5⊗A+C
.5⊗A+C	.5⊗A+C	A+C+E

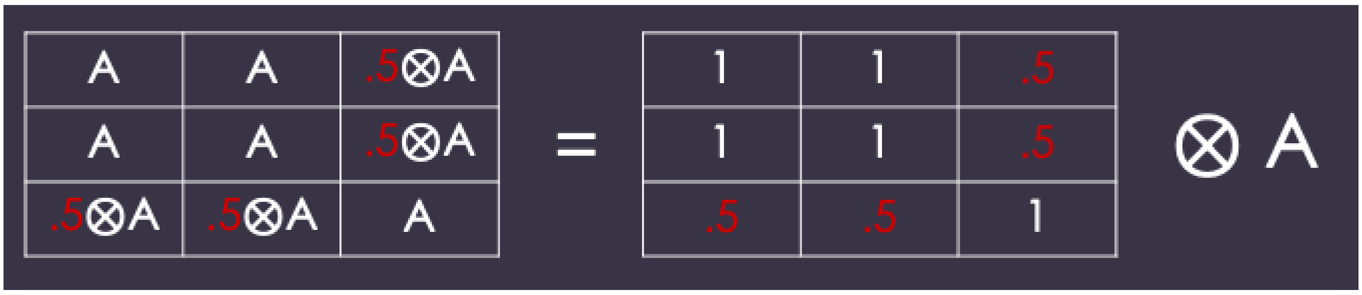
And for a DZ pair and sibling as:



A+C+E	.5⊗A+C	.5⊗A+C
.5⊗A+C	A+C+E	.5⊗A+C
.5⊗A+C	.5⊗A+C	A+C+E

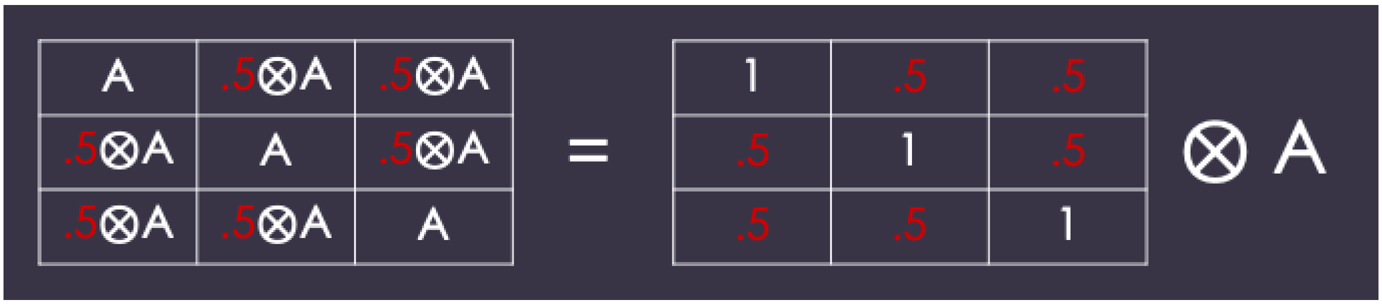
An alternative way to parameterise this is to create a matrix that represents the expected relationships for each A, C, and E component, then use a kronecker product (check 'Matrix Multiplication Sheet' for details on the types of matrix multiplication) to multiply these relationship matrices with each of the A, C, and E components.

For MZ the A component:



$$\begin{bmatrix} A & A & .5\otimes A \\ A & A & .5\otimes A \\ .5\otimes A & .5\otimes A & A \end{bmatrix} = \begin{bmatrix} 1 & 1 & .5 \\ 1 & 1 & .5 \\ .5 & .5 & 1 \end{bmatrix} \otimes A$$

For DZ the A component:



$$\begin{bmatrix} A & .5\otimes A & .5\otimes A \\ .5\otimes A & A & .5\otimes A \\ .5\otimes A & .5\otimes A & A \end{bmatrix} = \begin{bmatrix} 1 & .5 & .5 \\ .5 & 1 & .5 \\ .5 & .5 & 1 \end{bmatrix} \otimes A$$

The C component for both MZ and DZ:

$$\begin{array}{|c|c|c|}\hline C & C & C \\ \hline C & C & C \\ \hline C & C & C \\ \hline\end{array} = \begin{array}{|c|c|c|}\hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline\end{array} \otimes C$$

The E component for both MZ and DZ:

$$\begin{array}{|c|c|c|}\hline E & 0 & 0 \\ \hline 0 & E & 0 \\ \hline 0 & 0 & 0 \\ \hline\end{array} = \begin{array}{|c|c|c|}\hline 1 & 0 & 0 \\ \hline 0 & 1 & 0 \\ \hline 0 & 0 & 1 \\ \hline\end{array} \otimes E$$

These A, C, E variance-covariance matrices are the same dimensions and can be simply summed together.

In OpenMx the code for the relationship matrices looks like:

```
relMZ      <- mxMatrix( type="Symm", nrow=nt, ncol=nt,
free=FALSE, values=c(1,1,.5,1,.5,1),  name="rAmz" )
relDZ      <- mxMatrix( type="Symm", nrow=nt, ncol=nt,
free=FALSE, values=c(1,.5,.5,1,.5,1), name="rAdz" )
relC       <- mxMatrix( type="Unit", nrow=nt, ncol=nt,
free=FALSE, name="rC" )
relE       <- mxMatrix( type="Iden", nrow=nt, ncol=nt,
free=FALSE, name="rE" )
```

We can multiply these relationship matrices with the A, C, E components and sum them together in a single step:

```
expCovMZ  <- mxAlgebra( expression= rAmz%x%VA + rC%x%VC +
rE%x%VE, name="expCovMZ" )
expCovDZ  <- mxAlgebra( expression= rAdz%x%VA + rC%x%VC +
rE%x%VE, name="expCovDZ" )
```

Q3.14. Open **02_ACEsib_alt_binary.R**
Run the code up to when the model is built (~line 104).

Before you fit the model, have a look in the relMZ and relDZ objects and use mxEval to look at the expected variance/covariance matrices:

```
mxEval(expCovMZ, modelMZ, compute=TRUE)
mxEval(expCovDZ, modelDZ, compute=TRUE)
```

The first argument is the name of an object that is created using mxAlgebra. The second is the name of the model that it belongs to. The third asks for the algebra to be calculated.

What are the values in this the expected MZ variance/covariance matrix before the model is run? (NOTE: only the lower diagonal is needed, the matrix is symmetric)

	T1	T2	Sib
T1			
T2			
Sib			

Q3.15. What are the values in the expected DZ variance/covariance matrix before the model is run?

	T1	T2	Sib
T1			
T2			
Sib			

Q3.16. Run the model.

What are the values after estimation for the MZ variance/covariance matrix?

	T1	T2	Sib
T1			
T2			
Sib			

Q3.17. Would you like a hint on how to extract this matrix from the output?

○ Yes

Q3.18. Hint:
fitACE$output$algebras$MZ.expCovMZ

fitACE$output$algebras$DZ.expCovDZ

Q3.19. What are the values after estimation for the DZ variance/covariance matrix?

Notice similarities and differences between the estimated values and the start values.

	T1	T2	Sib
T1			
T2			
Sib			

Q3.20. Record the model fit:

	Values
Fit -2LL	
df	
parameters	

Q3.21. Record the estimated variance components:

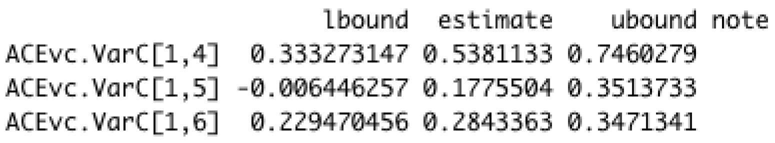
	lower 95% CI	Estimate	upper 95% CI
A			
C			
E			

Q3.22. Record the power:

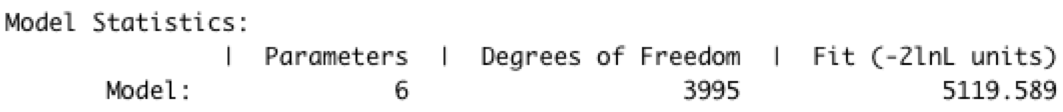
	Power
A	
C	

Q3.23. How do these estimates compare to the previous models?

ACE twins



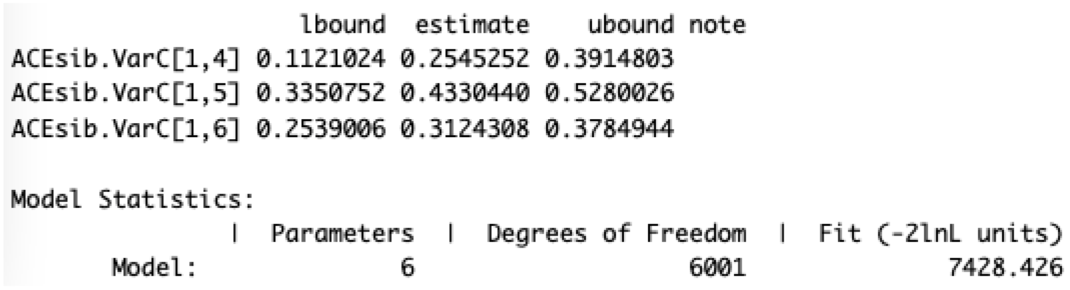
```
                     lbound  estimate    ubound note
ACEvc.VarC[1,4]  0.333273147 0.5381133 0.7460279
ACEvc.VarC[1,5] -0.006446257 0.1775504 0.3513733
ACEvc.VarC[1,6]  0.229470456 0.2843363 0.3471341
```



```
Model Statistics:
               |  Parameters  |  Degrees of Freedom  |  Fit (-2lnL units)
       Model:              6                   3995             5119.589
```

Power: A = 0.9997855 & C = 0.5993366

ACE twins & sib



```
                    lbound  estimate    ubound note
ACEsib.VarC[1,4] 0.1121024 0.2545252 0.3914803
ACEsib.VarC[1,5] 0.3350752 0.4330440 0.5280026
ACEsib.VarC[1,6] 0.2539006 0.3124308 0.3784944

Model Statistics:
               |  Parameters  |  Degrees of Freedom  |  Fit (-2lnL units)
      Model:             6                   6001              7428.426
```

Power: A = 0.9640506 & C = 1

Q3.24. Up until now, we have created the final model from two separate models, one for MZ and one for DZ. These models differ only in the coefficient of relatedness that is incorporated into the A part of the expected variance/covariance matrix.

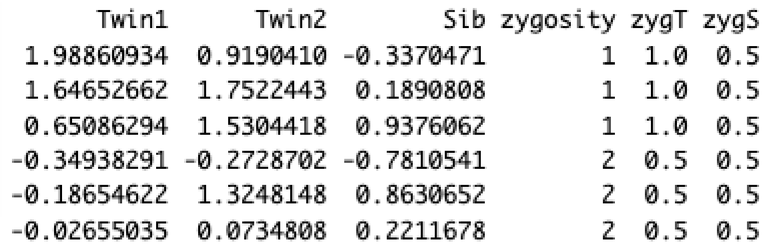
This has been a hard-coded matrix that is different for MZ and DZ:

```
relMZ      <- mxMatrix( type="Symm", nrow=nt, ncol=nt,
free=FALSE, values=c(1,1,.5,1,.5,1),  name="rAmz" )
relDZ      <- mxMatrix( type="Symm", nrow=nt, ncol=nt,
free=FALSE, values=c(1,.5,.5,1,.5,1), name="rAdz" )
```

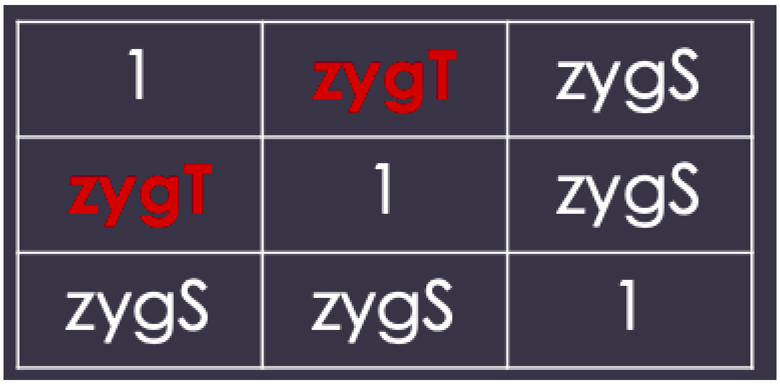
Alternatively, we can use a definition variable to hold the coefficient of relatedness for each pair of individuals and use that data in a relationship matrix that could be used for all twin pairs:

```
relA      <- mxMatrix( type="Stand", nrow=nt, ncol=nt,
free=FALSE, labels=c("data.zygT","data.zygS","data.zygS"),
name="rA" )
```

zygT is the coefficient of relationship between Twin1 and Twin2. For MZ pairs this will equal 1, for DZ pairs this will equal 0.5. zygS is the coefficient of relationship between Twin1/Twin2 and their Sibling. This will equal 0.5 for all pairs.



```
      Twin1      Twin2        Sib zygosity zygT zygS
 1.98860934  0.9190410 -0.3370471        1  1.0  0.5
 1.64652662  1.7522443  0.1890808        1  1.0  0.5
 0.65086294  1.5304418  0.9376062        1  1.0  0.5
-0.34938291 -0.2728702 -0.7810541        2  0.5  0.5
-0.18654622  1.3248148  0.8630652        2  0.5  0.5
-0.02655035  0.0734808  0.2211678        2  0.5  0.5
```



1	zygT	zygS
zygT	1	zygS
zygS	zygS	1

Q3.25. Open **03_ACEzygdef_binary.R** and run the script up to building the final model (~line 94).

Have a look in the relA matrix and use mxEval to have a look in the expCov matrix.

Do you want a hint for how to use mxEval?

- ○ Yes

Q3.26. HINT:
`mxEval(expCov,modelACE,compute=T)`

Q3.27. Is the expCov matrix what you would expect for an MZ pair or a DZ pair?

- ○ MZ
- ○ DZ

Q3.28. When using mxEval to check matrices, for definition variables it will use the first line of data. In our case that is an MZ twin pair.

Q3.29. Run the model.

Record the model fit:

	Values
Fit -2LL	
df	
parameters	

Q3.30. Record the estimated variance components:

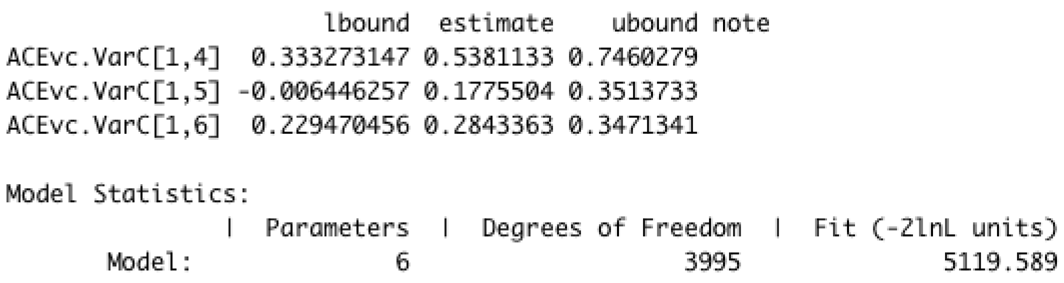
	lower 95% CI	Estimate	upper 95% CI
A			
C			
E			

Q3.31. Record the power:

	Power
A	
C	

Q3.32. How do these estimates compare to the previous models?

ACE twins

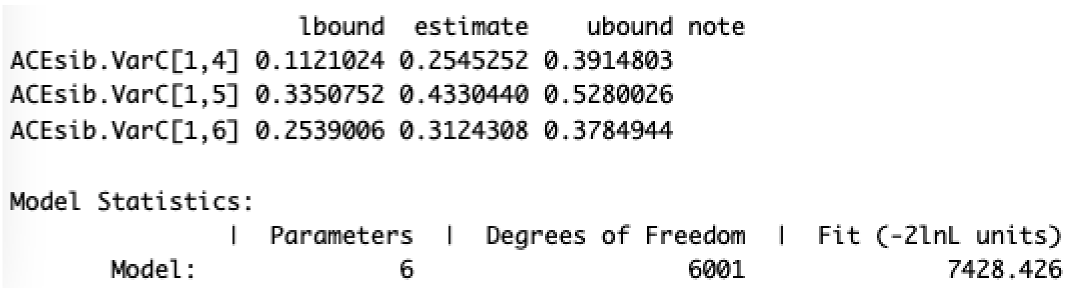


```
                      lbound  estimate    ubound note
ACEvc.VarC[1,4]  0.333273147 0.5381133 0.7460279
ACEvc.VarC[1,5] -0.006446257 0.1775504 0.3513733
ACEvc.VarC[1,6]  0.229470456 0.2843363 0.3471341

Model Statistics:
               |  Parameters  |  Degrees of Freedom  |  Fit (-2lnL units)
       Model:              6                   3995             5119.589
```

Power: A = 0.9997855 & C = 0.5993366

ACE twins & sib

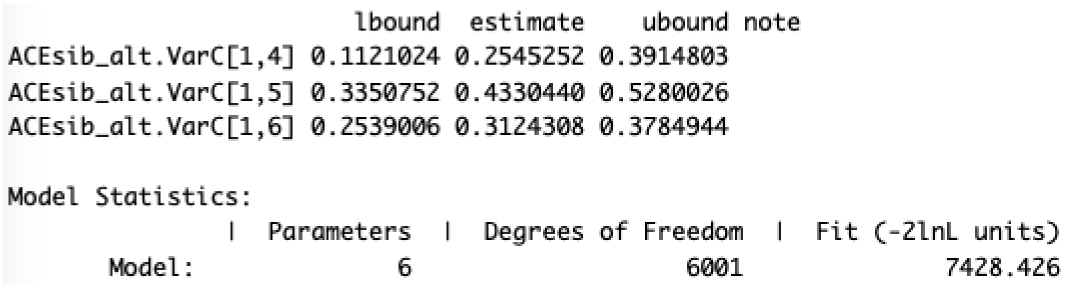


```
                    lbound  estimate    ubound note
ACEsib.VarC[1,4] 0.1121024 0.2545252 0.3914803
ACEsib.VarC[1,5] 0.3350752 0.4330440 0.5280026
ACEsib.VarC[1,6] 0.2539006 0.3124308 0.3784944

Model Statistics:
               |  Parameters  |  Degrees of Freedom  |  Fit (-2lnL units)
       Model:              6                   6001             7428.426
```

Power: A = 0.9640506 & C = 1

ACE twins & sib alternate parameterisation



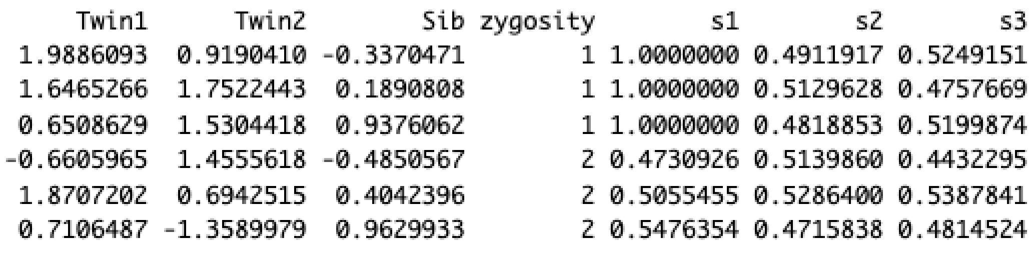
```
                        lbound  estimate    ubound note
ACEsib_alt.VarC[1,4] 0.1121024 0.2545252 0.3914803
ACEsib_alt.VarC[1,5] 0.3350752 0.4330440 0.5280026
ACEsib_alt.VarC[1,6] 0.2539006 0.3124308 0.3784944

Model Statistics:
               |  Parameters  |  Degrees of Freedom  |  Fit (-2lnL units)
       Model:              6                   6001             7428.426
```

Power: A = 0.9640506 & C = 1

Q3.33. So far we have used the theoretical coefficient of relatedness based on pedigree information between the individuals in the family. If we have measured genetic relationships between pairs of individuals, then we can use it as a definition variables in these models.

Open **04_ACEgrm_relatedness_binary.R**
Load the data and have a look at the columns.



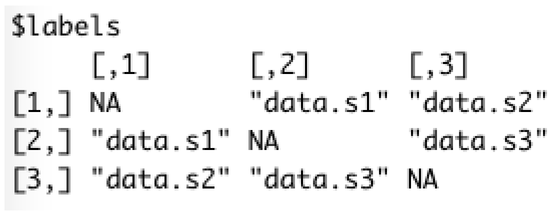
```
     Twin1      Twin2        Sib zygosity        s1        s2        s3
 1.9886093  0.9190410 -0.3370471        1 1.0000000 0.4911917 0.5249151
 1.6465266  1.7522443  0.1890808        1 1.0000000 0.5129628 0.4757669
 0.6508629  1.5304418  0.9376062        1 1.0000000 0.4818853 0.5199874
-0.6605965  1.4555618 -0.4850567        2 0.4730926 0.5139860 0.4432295
 1.8707202  0.6942515  0.4042396        2 0.5055455 0.5286400 0.5387841
 0.7106487 -1.3589979  0.9629933        2 0.5476354 0.4715838 0.4814524
```

s1 = the genetic relatedness coefficient between Twin1 and Twin 2
s2 = the genetic relatedness coefficient between Twin1 and Sib
s3 = the genetic relatedness coefficient between Twin2 and Sib

Have a look at the distribution of these relatedness variables.
We can still use a threshold on the relatedness between Twin1 and Twin2 to check out the correlations in our data.

Have a look at the relA matrix, which pulls in the relatedness data as a definition variable:

```
relA      <- mxMatrix( type="Stand", nrow=nt, ncol=nt,
free=FALSE, labels=c("data.s1","data.s2","data.s3"), name="rA"
)
```



```
$labels
     [,1]      [,2]      [,3]
[1,] NA        "data.s1" "data.s2"
[2,] "data.s1" NA        "data.s3"
[3,] "data.s2" "data.s3" NA
```

Run the rest of the script fit the model.

Record the model fit:

	Values
Fit -2LL	
df	
parameters	

Q3.34. Record the estimated variance components:

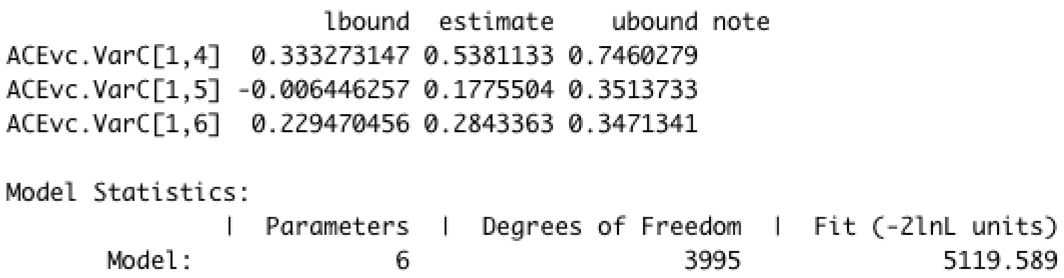
	lower 95% CI	Estimate	upper 95% CI
A			
C			
E			

Q3.35. Record the power:

	Power
A	
C	

Q3.36. How do these estimates compare to the previous models?

ACE twins

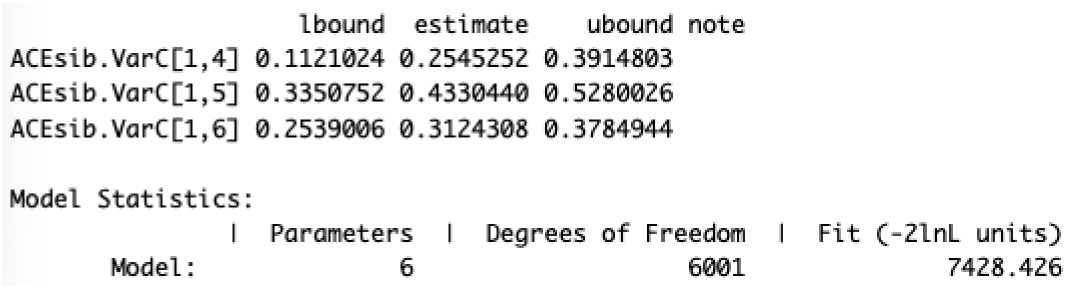


```
                     lbound  estimate    ubound note
ACEvc.VarC[1,4]  0.333273147 0.5381133 0.7460279
ACEvc.VarC[1,5] -0.006446257 0.1775504 0.3513733
ACEvc.VarC[1,6]  0.229470456 0.2843363 0.3471341

Model Statistics:
               |  Parameters  |  Degrees of Freedom  |  Fit (-2lnL units)
       Model:              6                   3995             5119.589
```

Power: A = 0.9997855 & C = 0.5993366

ACE twins & sib

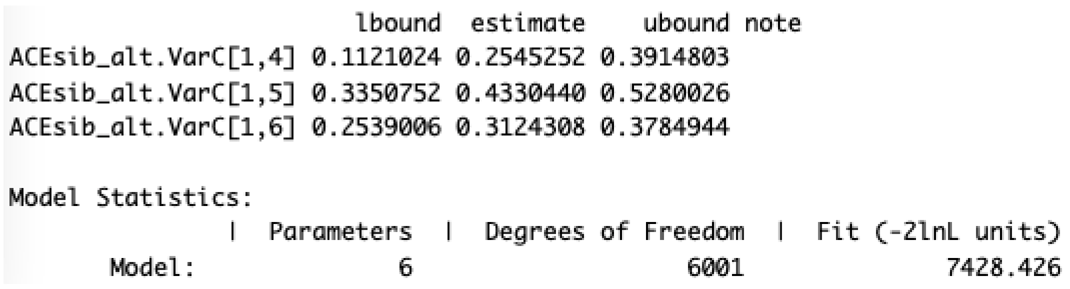


```
                   lbound  estimate    ubound note
ACEsib.VarC[1,4] 0.1121024 0.2545252 0.3914803
ACEsib.VarC[1,5] 0.3350752 0.4330440 0.5280026
ACEsib.VarC[1,6] 0.2539006 0.3124308 0.3784944

Model Statistics:
               |  Parameters  |  Degrees of Freedom  |  Fit (-2lnL units)
       Model:              6                   6001             7428.426
```

Power: A = 0.9640506 & C = 1

ACE twins & sib alternate parameterisation

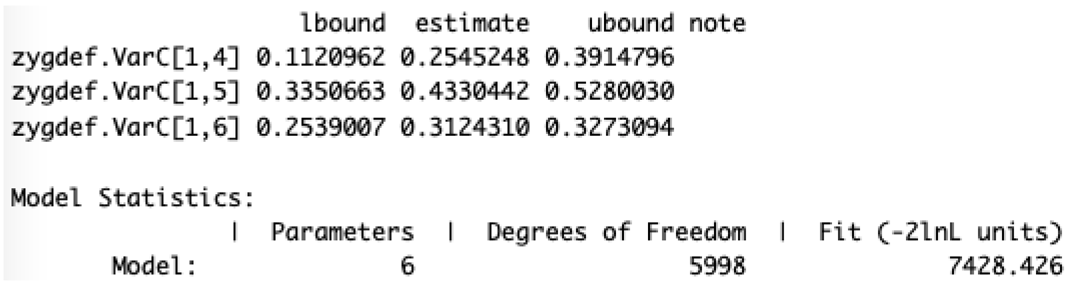


```
                        lbound  estimate    ubound note
ACEsib_alt.VarC[1,4] 0.1121024 0.2545252 0.3914803
ACEsib_alt.VarC[1,5] 0.3350752 0.4330440 0.5280026
ACEsib_alt.VarC[1,6] 0.2539006 0.3124308 0.3784944

Model Statistics:
               |  Parameters  |  Degrees of Freedom  |  Fit (-2lnL units)
      Model:              6                   6001              7428.426
```

Power: A = 0.9640506 & C = 1

ACE twins and sib with zygosity as a definition variable



```
                    lbound  estimate    ubound note
zygdef.VarC[1,4] 0.1120962 0.2545248 0.3914796
zygdef.VarC[1,5] 0.3350663 0.4330442 0.5280030
zygdef.VarC[1,6] 0.2539007 0.3124310 0.3273094

Model Statistics:
               |  Parameters  |  Degrees of Freedom  |  Fit (-2lnL units)
      Model:              6                   5998              7428.426
```

Power: A = 0.9640506 & C = 1

Q3.37. Now that we have a model that uses measured genetic variation, the model can be identified without MZ pairs, but there is a cost!

Open **05_ACEgrm_relatednessDZonly_binary.R**

This script is set up to read in a different dataset is much larger but used the same model specifications in the simulation as the previous dataset.

Run the model.

Q3.38. Record the model fit:

	Values
Fit -2LL	
df	
parameters	

Q3.39. Record the estimated variance components:

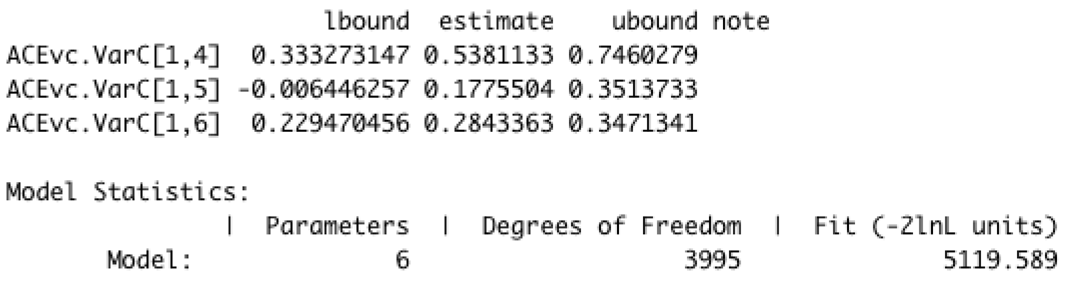
	lower 95% CI	Estimate	upper 95% CI
A			
C			
E			

Q3.40. Record the power:

	Power
A	
C	

Q3.41. How do these estimates compare to the previous models?

ACE twins

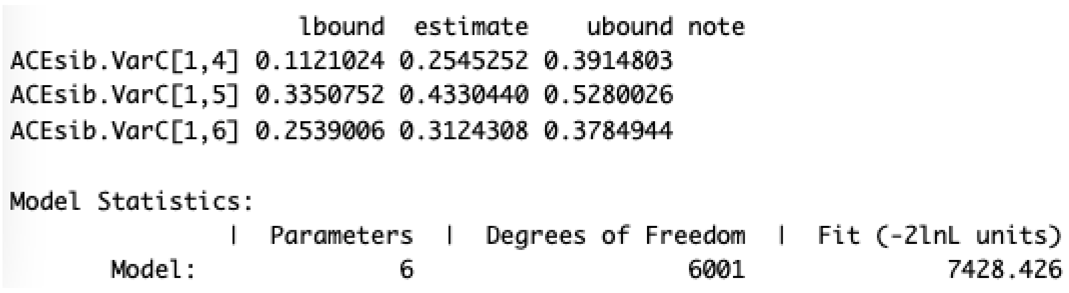


```
                      lbound  estimate    ubound note
ACEvc.VarC[1,4]  0.333273147 0.5381133 0.7460279
ACEvc.VarC[1,5] -0.006446257 0.1775504 0.3513733
ACEvc.VarC[1,6]  0.229470456 0.2843363 0.3471341

Model Statistics:
               |  Parameters  |  Degrees of Freedom  |  Fit (-2lnL units)
       Model:              6                   3995             5119.589
```

Power: A = 0.9997855 & C = 0.5993366

ACE twins & sib

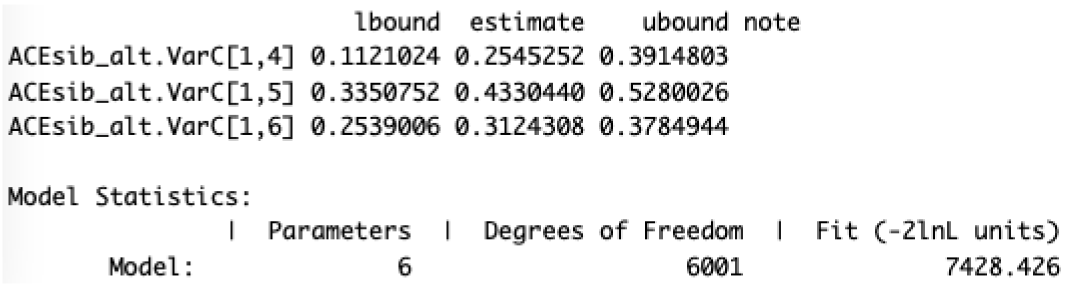


```
                    lbound  estimate    ubound note
ACEsib.VarC[1,4] 0.1121024 0.2545252 0.3914803
ACEsib.VarC[1,5] 0.3350752 0.4330440 0.5280026
ACEsib.VarC[1,6] 0.2539006 0.3124308 0.3784944

Model Statistics:
               |  Parameters  |  Degrees of Freedom  |  Fit (-2lnL units)
       Model:              6                   6001             7428.426
```

Power: A = 0.9640506 & C = 1

ACE twins & sib alternate parameterisation

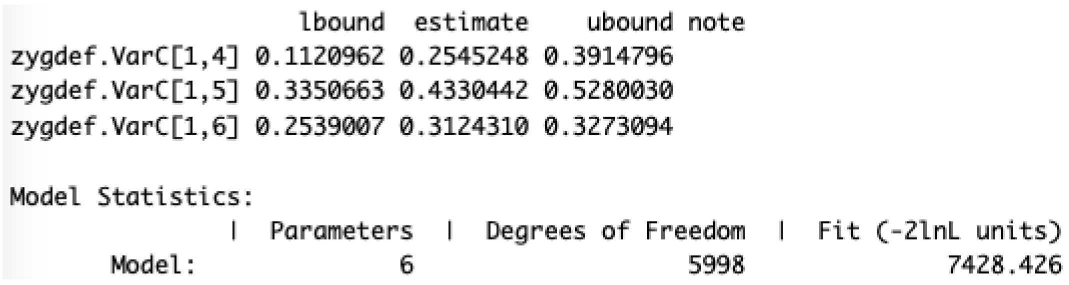


```
                        lbound  estimate    ubound note
ACEsib_alt.VarC[1,4] 0.1121024 0.2545252 0.3914803
ACEsib_alt.VarC[1,5] 0.3350752 0.4330440 0.5280026
ACEsib_alt.VarC[1,6] 0.2539006 0.3124308 0.3784944

Model Statistics:
               |  Parameters  |  Degrees of Freedom  |  Fit (-2lnL units)
       Model:              6                   6001             7428.426
```

Power: A = 0.9640506 & C = 1

ACE twins and sib with zygosity as a definition variable

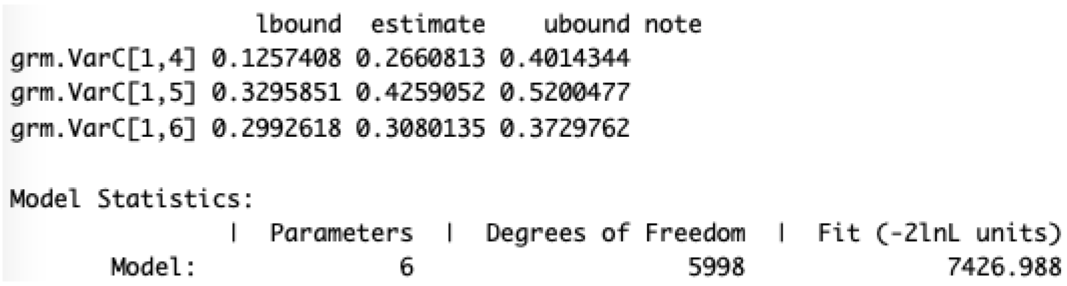


```
                    lbound  estimate    ubound note
zygdef.VarC[1,4] 0.1120962 0.2545248 0.3914796
zygdef.VarC[1,5] 0.3350663 0.4330442 0.5280030
zygdef.VarC[1,6] 0.2539007 0.3124310 0.3273094

Model Statistics:
               |  Parameters  |  Degrees of Freedom  |  Fit (-2lnL units)
       Model:              6                   5998             7428.426
```

Power: A = 0.9640506 & C = 1

ACE twins and sib with genetic relatedness measured

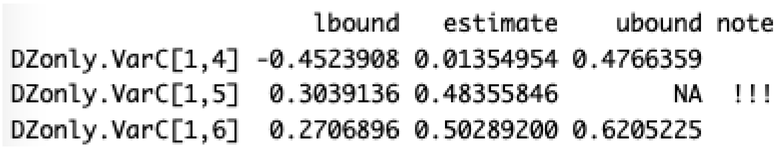


```
                lbound  estimate    ubound note
grm.VarC[1,4] 0.1257408 0.2660813 0.4014344
grm.VarC[1,5] 0.3295851 0.4259052 0.5200477
grm.VarC[1,6] 0.2992618 0.3080135 0.3729762

Model Statistics:
              |  Parameters  |  Degrees of Freedom  |  Fit (-2lnL units)
      Model:             6                  5998             7426.988
```

Power: A = 0.9773919 & C = 1

Q3.42. You might have noticed that not all the confidence intervals were estimated with this final model.



```
                    lbound   estimate    ubound note
DZonly.VarC[1,4] -0.4523908 0.01354954 0.4766359
DZonly.VarC[1,5]  0.3039136 0.48355846        NA  !!!
DZonly.VarC[1,6]  0.2706896 0.50289200 0.6205225
```

Again, we can try a few things to fit these confidence intervals. Like before, some places to start are:

1. Check and maybe change start values
2. Use mxTryHard() or mxTryHardOrdinal() instead of mxRun()
3. Try a different optimiser: NPSOL or CSOLNP or SLSQP e.g. mxOption(NULL,"Default optimizer", "CSOLNP")

In this case, try fitting the model with mxTryHardOrdinal()
This function has different default options that guide optimisation and like the other "TryHard" function, it will iterate through several attempts at running the model (default is 10 extra attempts) to try and obtain an acceptable fit.

To try fitting the model with this run:
```
fitACE    <- mxTryHardOrdinal( modelACE, intervals=T )
```

Q3.43. We've now shown you five different ways of fitting a twin model.
00_ACEvc_contin.R & 01_ACEsib_contin.R
02_ACEsib_alt_contin.R
03_ACEzygdef_contin.R
04_ACEgrm_relatedness_contin.R
05_ACEgrm_relatedness_DZonly__contin.R

Under what circumstances might one be a more appropriate choice than another? Discuss amongst your group.

Q3.44. This is the end of today's practical.

Q3.45. If you click "next" you will exit the practical.

Powered by Qualtrics