(Re)introduction to OpenMx

Sarah Medland

Starting at the beginning

- Opening R
  - Gui – double click
  - Unix/Terminal – type R
- Closing R
  - Gui – click on the x
  - Unix/Terminal – type q() then n
- Yes <- is the same as =
- What directory am I in?
  - getwd()
- Change directory?
  - setwd("H:/tues_morning”)

Starting at the beginning

- Data preparation
  - The algebra style used in Mx expects 1 line per case/family
  - (Almost) limitless number of families and variables
  - Data needs to be read into R before it can be analysed
    - (the commands to read the data can be nested within the R script)
    - Default missing code is now **NA**

Reading in data

- Example data:ozbmi2.txt

fam	agecat	age	zyg	part	wt1	wt2	ht1	ht2	htwt1	htwt2	bmi1	bmi2
115	0	0.21	1	2	58	57	1.7	1.7	20.0692	19.7232	20.9943	20.8726
121	0	0.24	1	2	54	53	1.6299	1.6299	20.3244	19.9481	21.0828	20.9519
158	0	0.21	1	2	55	50	1.6499	1.6799	20.202	17.7154	21.0405	20.121
172	0	0.21	1	2	66	76	1.5698	1.6499	26.7759	27.9155	23.0125	23.3043
182	0	0.19	1	2	50	48	1.6099	1.6299	19.2894	18.0662	20.7169	20.2583
199	0	0.26	1	2	60	60	1.5999	1.5698	23.4375	24.3418	22.0804	22.3454

- data<-read.table("ozbmi2.txt", header=T, na.strings = "NA")
- head(data)

```
> head(data)
  fam agecat  age zyg part wt1 wt2     ht1     ht2   htwt1   htwt2    bmi1    bmi2
1 115      0 0.21   1    2  58  57     1.7     1.7 20.0692 19.7232 20.9943 20.8726
2 121      0 0.24   1    2  54  53  1.6299  1.6299 20.3244 19.9481 21.0828 20.9519
3 158      0 0.21   1    2  55  50  1.6499  1.6799  20.202 17.7154 21.0405  20.121
4 172      0 0.21   1    2  66  76  1.5698  1.6499 26.7759 27.9155 23.0125 23.3043
5 182      0 0.19   1    2  50  48  1.6099  1.6299 19.2894 18.0662 20.7169 20.2583
6 199      0 0.26   1    2  60  60  1.5999  1.5698 23.4375 24.3418 22.0804 22.3454
```

Selecting and sub-setting the data

- Make separate data sets for the MZ and DZ

```
> mzData <- as.data.frame(subset(data, zyg<3, c(bmi1,bmi2)))
> dzData <- as.data.frame(subset(data, zyg>2, c(bmi1,bmi2)))
> head(dzData)
       bmi1    bmi2
843 21.9642      NA
844 21.8791 21.2112
845 22.2321 22.6044
846 19.8491 20.1743
847 20.1743      NA
848 21.7050 21.2905
```

- Check data is numeric and behaves as expected

```
> cov(mzData,use="complete")
          bmi1      bmi2
bmi1 0.8779390 0.6734489
bmi2 0.6734489 0.8987715
> cov(dzData,use="complete")
          bmi1      bmi2
bmi1 0.8908474 0.2872594
bmi2 0.2872594 0.8657751
> colMeans(mzData,na.rm=TRUE)
    bmi1     bmi2
21.75089 21.73471
> colMeans(dzData,na.rm=TRUE)
    bmi1     bmi2
21.68689 21.88095
```

Common error

- Problem: data contains a non numeric value

```
> cov(dzData,use="complete")
          bmi1      bmi2
bmi1 0.8908474 0.2872594
bmi2 0.2872594 0.8657751
Warning message:
In cov(dzData, use = "complete") : NAs introduced by coercion
> colMeans(mzData,na.rm=TRUE)
Error in colMeans(mzData, na.rm = TRUE) : 'x' must be numeric
> colMeans(dzData,na.rm=TRUE)
Error in colMeans(dzData, na.rm = TRUE) : 'x' must be numeric
```

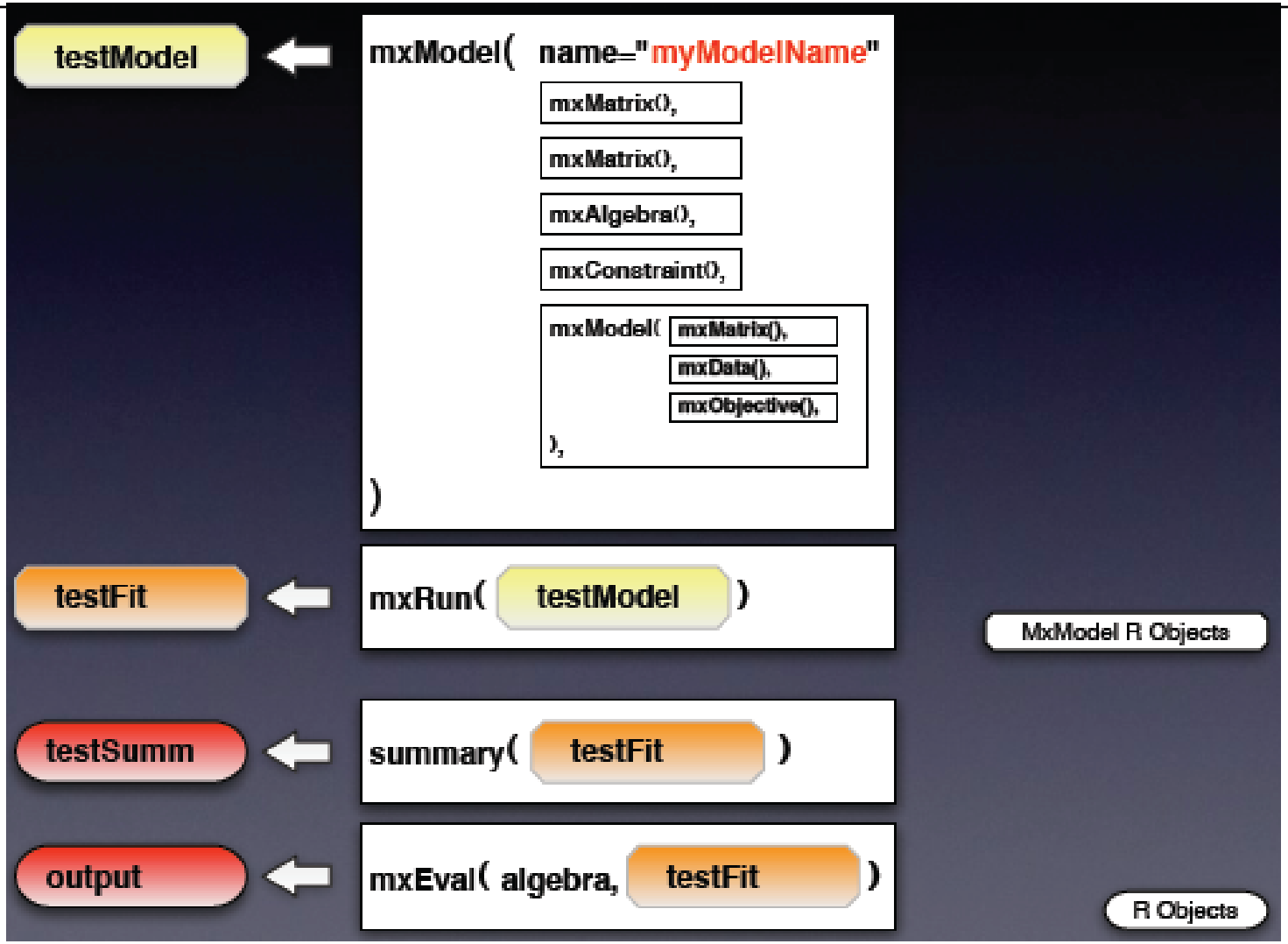
20171	1	0.35	5	2	51	79	1.5999	1.7998	19.9219	24.3827	20.9427	22.3571
20188	1	0.37	5	2	53	65	1.5698	1.73	21.5019	21.7181	21.477	21.547
20204	1	0.53	5	1	58	64	1.6299	NA	21.83	NA	A	NA
20390	1	0.37	5	2	64	73	1.6499	1.8298	23.5078	21.7982	22.1013	21.5728
20398	1	0.52	5	2	60	77	1.6299	1.73	22.5827	25.7276	21.8203	22.7329

- Equivalent Mx Classic error - *Uh-oh... I'm having trouble reading a number in D or E format*

Important structural stuff

- Mx Classic
  - Each job was composed of one or more groups
    - Each group is ‘opened’ with a title
    - Each group is ‘closed’ with an end statement
- openMx
  - Less structured

General Hierarchy



Matrices: the building blocks

```
mxMatrix( type="Lower", nrow=nv, ncol=nv, free=TRUE,
values=.6, label="a11", name="a" ), #X
```

- Many types eg. type="Lower"
- Denoted by names eg. name="a“
- Size eg. nrow=nv, ncol=nv
- All estimated parameters must be placed in a matrix & Mx must be told what type of matrix it is

Matrices: the building blocks

- Many types

```
mxMatrix( type="Zero", nrow=2,
ncol=3, name="a" )
```

```
0 0 0
0 0 0
```

```
mxMatrix( type="Unit", nrow=2,
ncol=3, name="a" )
```

```
1 1 1
1 1 1
```

```
mxMatrix( type="Ident", nrow=3,
ncol=3, name="a" )
```

```
1 0 0
0 1 0
0 0 1
```

```
mxMatrix( type="Diag", nrow=3,
ncol=3, free=TRUE, name="a" )
```

```
? 0 0
0 ? 0
0 0 ?
```

```
mxMatrix( type="Sdiag", nrow=3,
ncol=3, free=TRUE, name="a" )
```

```
0 0 0
? 0 0
? ? 0
```

```
mxMatrix( type="Stand", nrow=3,
ncol=3, free=TRUE, name="a" )
```

```
1 ? ?
? 1 ?
? ? 1
```

```
mxMatrix( type="Symm", nrow=3,
ncol=3, free=TRUE, name="a" )
```

```
? ? ?
? ? ?
? ? ?
```

```
mxMatrix( type="Lower", nrow=3,
ncol=3, free=TRUE, name="a" )
```

```
? 0 0
? ? 0
? ? ?
```

```
mxMatrix( type="Full", nrow=2,
ncol=4, free=TRUE, name="a" )
```

```
? ? ? ?
? ? ? ?
```

Short cuts

- Anything after # is read as a comment
- Can predefine frequently used/changed parameters
  - nv <- 1
- Can read in scripts or R functions
  - source("GenEpiHelperFunctions.R")

Setting up the script – overview

```
univACEModel <-
mxModel("univACE",
    mxModel("ACE", ...    ),
    mxModel("MZ", ...    ),
    mxModel("DZ", ... ),
    mxAlgebra( ... ),
    mxAlgebraObjective( ... )
)

univACEFit <- mxRun(univACEModel)
univACESumm <-
summary(univACEFit)
```

Setting up the script – univACE mxModel

- 3 sub mxModels
  - ACE
  - MZ
  - DZ
- mxAlgebra
- mxAlgebraObjective

```
univACEModel <-
mxModel("univACE",
    mxModel("ACE", ...    ),
    mxModel("MZ", ...    ),
    mxModel("DZ", ... ),
    mxAlgebra( ... ),
    mxAlgebraObjective( ... )
)

univACEFit <- mxRun(univACEModel)
univACESumm <-
summary(univACEFit)
```

Setting up the script – ACE mxModel

```
# Fit ACE Model with RawData and Matrices Input
# ------------------------------------------------------------------------------
univACEModel <- mxModel("univACE",
    mxModel("ACE",
    # Matrices a, c, and e to store a, c, and e path coefficients
        mxMatrix( type="Lower", nrow=nv, ncol=nv, free=TRUE, values=.6, label="a11", name="a" ), #X
        mxMatrix( type="Lower", nrow=nv, ncol=nv, free=TRUE, values=.6, label="c11", name="c" ), #Y
        mxMatrix( type="Lower", nrow=nv, ncol=nv, free=TRUE, values=.6, label="e11", name="e" ), #Z
```

Classic Mx translation

```
univACE          !Job Title
ACE              !Group Title
...
a Lower nv nv free
c Lower nv nv free
e Lower nv nv free
...
Start .6 a 1 1 1 c 1 1 1 e 1 1 1
```

Notice we no longer have group types and can declare all the information about a matrix in one place!

Setting up the script – ACE mxModel

```
# Matrices A, C, and E compute variance components
    mxAlgebra( expression=a %*% t(a), name="A" ),
    mxAlgebra( expression=c %*% t(c), name="C" ),
    mxAlgebra( expression=e %*% t(e), name="E" ),
```

Classic Mx translation

A = a*a' ;
C = c*c' ;
E = e*e' ;

Notice we are no longer restricted with matrix names!
Case maters

Setting up the script – ACE mxModel

```
# Algebra to compute total variances and standard deviations (diagonal only)
    mxAlgebra( expression=A+C+E, name="V" ),
    mxMatrix( type="Iden", nrow=nv, ncol=nv, name="I"),
    mxAlgebra( expression=solve(sqrt(I*V)), name="isd"),
```

Classic Mx translation

```
V=A+C+E ;
...
I Iden nv nv ! This is matrix with 1 on the diag and 0 on the off-diag
...
Isd = \sqrt(I.V) ;
```

Notice we no longer need to separate matrices and algebra!

Also the operators are different - very important!

Setting up the script – ACE mxModel

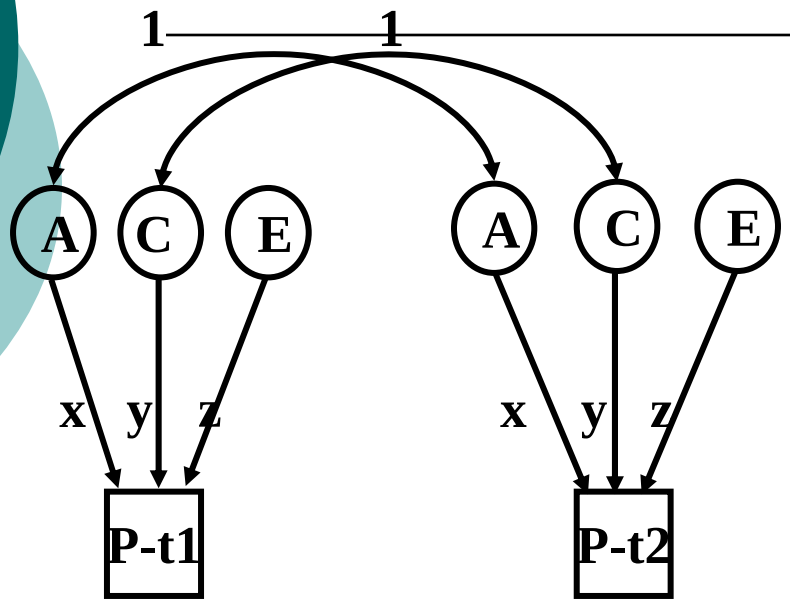
```
## Note that the rest of the mxModel statements do not change for bivariate/multivariate case
 # Matrix & Algebra for expected means vector
     mxMatrix( type="Full", nrow=1, ncol=nv, free=TRUE, values= 20, label="mean", name="M" ),
     mxAlgebra( expression= cbind(M,M), name="expMean"),
```

Classic Mx translation

M full 1 nv free
...
Start 20 M 1 1 1
...
Means (M|M) ;

Notice all the lines end in commas
No inconsistency in line endings

Setting up the script – MZ mxModel



MZ

	t1	t2
t1	$a^2+c^2+e^2$	a^2+c^2
t2	a^2+c^2	$a^2+c^2+e^2$

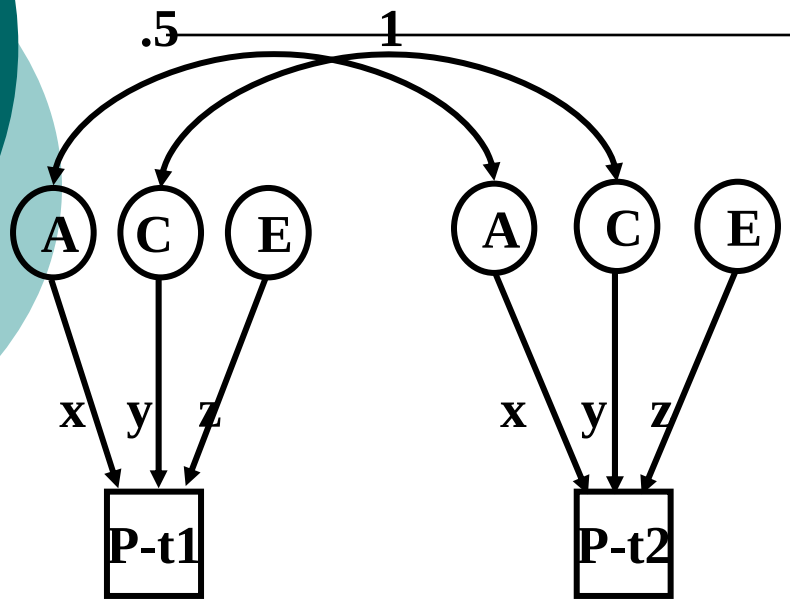
```
# Algebra for expected variance/covariance matrix in MZ
    mxAlgebra( expression= rbind  ( cbind(A+C+E , A+C),
                                    cbind(A+C   , A+C+E)), name="expCovMZ" ),
```

Classic Mx translation

```
Cov     A+C+E|A+C_
        A+C|A+C+E;
```

Notice the different adhesion styles cbind = | rbind = _

Setting up the script – DZ mxModel



DZ

	t1	t2
t1	$a^2+c^2+e^2$	$.5a^2+c^2$
t2	$.5a^2+c^2$	$a^2+c^2+e^2$

```
# Algebra for expected variance/covariance matrix in DZ
    mxAlgebra( expression= rbind  ( cbind(A+C+E      , 0.5%x%A+C),
                                    cbind(0.5%x%A+C , A+C+E)),  name="expCovDZ" )
```

Classic Mx translation

```
Cov     A+C+E|H@A+C_
        H@A+C|A+C+E;
```

Notice you can now use **numbers** in the algebra they don't have to be placed in matrices

Setting up the script – MZ mxModel

```
mxModel("MZ",
    mxData( observed=mzData, type="raw" ),
    mxFIMLObjective( covariance="ACE.expCovMZ", means="ACE.expMean", dimnames=selVars )
),
```

Classic Mx translation

MZ !Group Title
...
Rec file =mzData
Select variables ...
...
Covariance
Means

(No translation for the mxFLObjective this was black box in Mx Classic)

Setting up the script – DZ mxModel

```
mxModel("DZ",
    mxData( observed=dzData, type="raw" ),
    mxFIMLObjective( covariance="ACE.expCovDZ", means="ACE.expMean", dimnames=selVars )
),
```

Setting up the script – Optimisation

```
    mxAlgebra( expression=MZ.objective + DZ.objective, name="-2sumll" ),
    mxAlgebraObjective("-2sumll")
)

univACEFit <- mxRun(univACEModel)
univACESumm <- summary(univACEFit)
```

This section of the script calculates the -2 log likelihood and runs the optimisation

(Mx Classic equivalent is clicking run)

Setting up the script – Summarising output using helper functions

```
# Generate ACE Output
# ------------------------------------------------------------------------
parameterSpecifications(univACEFit)
expectedMeansCovariances(univACEFit)
tableFitStatistics(univACEFit)

# Generate Table of Parameter Estimates using mxEval
pathEstimatesACE <- print(round(mxEval(cbind(ACE.a,ACE.c,ACE.e), univACEFit),4))
varComponentsACE <- print(round(mxEval(cbind(ACE.A/ACE.V,ACE.C/ACE.V,ACE.E/ACE.V), univACEFit),4))
    rownames(pathEstimatesACE) <- 'pathEstimates'
    colnames(pathEstimatesACE) <- c('a','c','e')
    rownames(varComponentsACE) <- 'varComponents'
    colnames(varComponentsACE) <- c('a^2','c^2','e^2')
pathEstimatesACE
varComponentsACE
```

Checking individual matrices

- a matrix from the ACE mxModel

Type

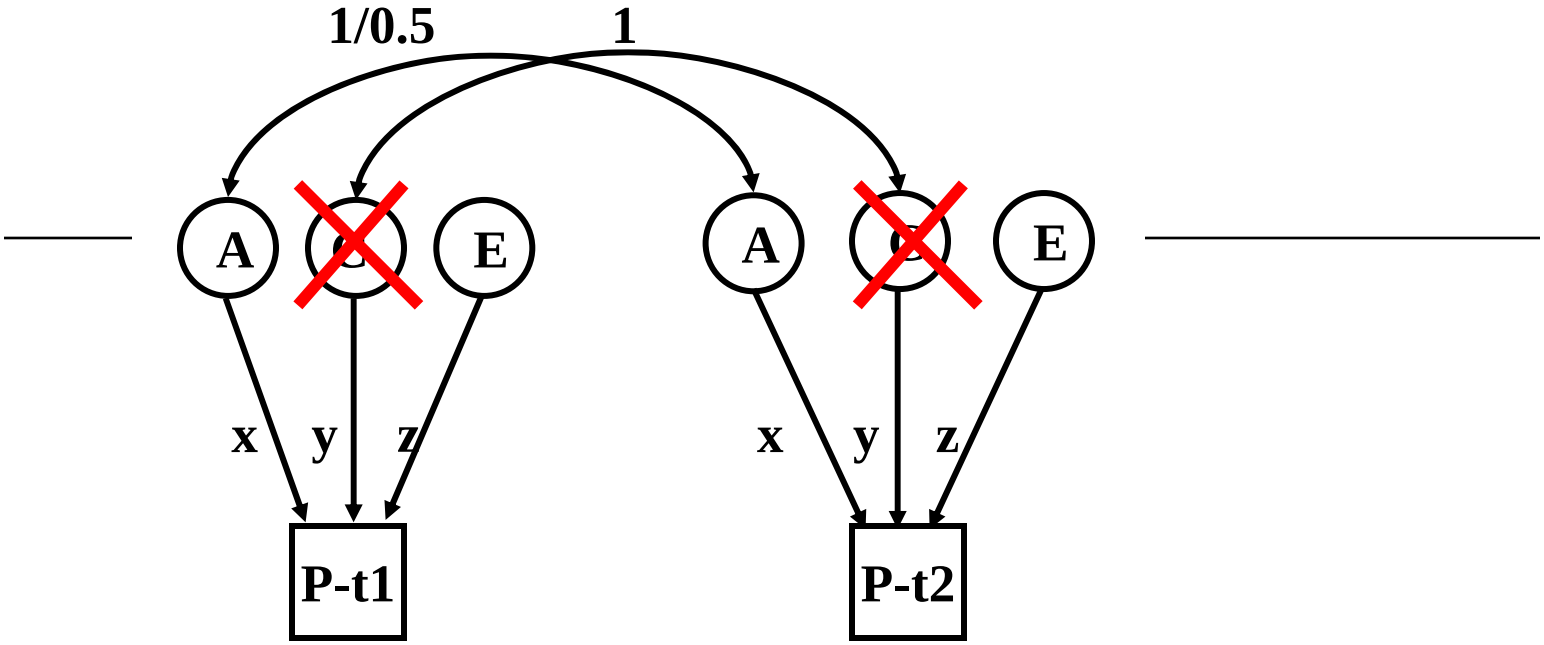
univACEModel@submodels$ACE@matrices$a

Compare to

univACEFit@submodels$ACE@matrices$a

Testing for significance

- Set (one or more) estimated parameters to zero
- Known as ‘dropping’ the parameter from the model
  - ie dropping C



Variance/covariance matrices

MZ

	t1	t2
t1	$a^2+c^2+e^2$	a^2+c^2
t2	a^2+c^2	$a^2+c^2+e^2$

DZ

	t1	t2
t1	$a^2+c^2+e^2$	$0.5a^2+c^2$
t2	$0.5a^2+c^2$	$a^2+c^2+e^2$

Testing for significance

```
# Fit AE model
# ------------------------------------------------------------------------
univAEModel <- mxModel(univACEFit, name="univAE",
    mxModel(univACEFit$ACE,
        mxMatrix( type="Full", nrow=1, ncol=1, free=FALSE, values=0, label="c11", name="c" )
    )
)
univAEFit <- mxRun(univAEModel)
univAESumm <- summary(univAEFit)

# Fit CE model
```

- Create a new mxModel “univAE” which draws from the “univACE” mxModel

Testing for significance

```
# Fit AE model
# ------------------------------------------------------------------------
univAEModel <- mxModel(univACEFit, name="univAE",
    mxModel(univACEFit$ACE,
        mxMatrix( type="Full", nrow=1, ncol=1, free=FALSE, values=0, label="c11", name="c" )
    )
)
univAEFit <- mxRun(univAEModel)
univAESumm <- summary(univAEFit)

# Fit CE model
```

- Redefine the c matrix within univAE
- Free=FALSE, values=0

Mx Classic equivalent is:

```
Drop C 1 1 1
End
```

Saving your output

- Save the R workspace
  - On closing click yes
  - Very big
  - Saves everything
- Save the fitted model
  - Equivalent to save in classic Mx
  - save(univACEFit, file="test.omxs")
  - load("test.omxs") – need to load OpenMx first

What to report

- Summary statistics
  - Usually from a simplified ‘saturated’ model
- Standardized estimates
  - Easier to conceptualise
    - ie 40% of the phenotypic variance vs a genetic effect of 2.84
    - Can easily be returned to original scale if summary statistics are provided

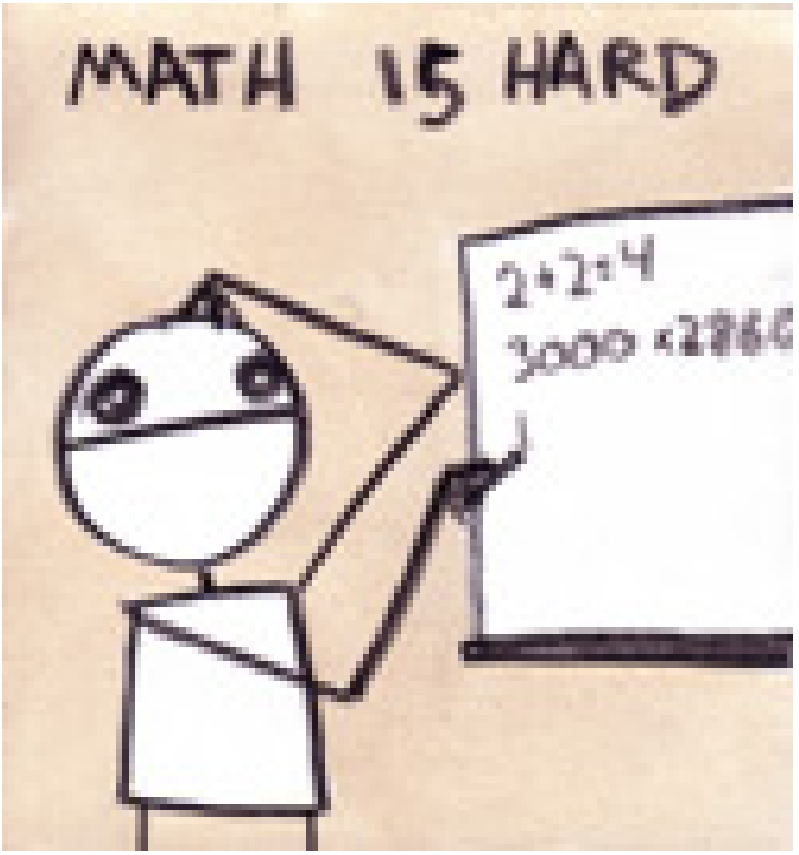
What to report

- Path coefficients
  - Very important in multivariate analyses
    - Gives a much clearer picture of the directionality of effects
- Variance components/proportion of variance explained
- Genetic correlations

General Advice/Problem solving

- Scripting styles differ
- Check the sample description
- Learn to love the webpage
- Comments are your friends

Time for coffee



explodingdog.com