Faculty Drive mcn/ 2011

MV association…

- Maximum likelihood – factor based approach
  - Mx
- Canonical Correlation approach
  - Plink
- Principal components approach
  - F-bat

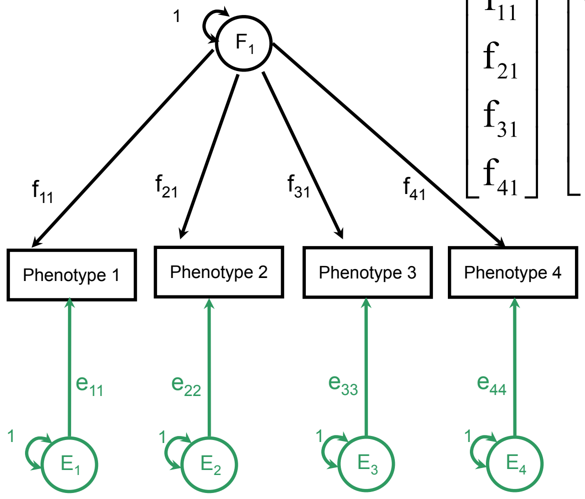
Funnel Web Spider

Maximum likelihood approach

- Unrelated individuals
  - Shared variance due to a common factor
  - Residual non-shared variance
- Family based data
  - ACE type models

Galah

Common factor model



$$\begin{bmatrix} f_{11} \\ f_{21} \\ f_{31} \\ f_{41} \end{bmatrix} \begin{bmatrix} e_{11} & 0 & 0 & 0 \\ 0 & e_{22} & 0 & 0 \\ 0 & 0 & e_{33} & 0 \\ 0 & 0 & 0 & e_{44} \end{bmatrix}$$

Great white

Factor level association

$$\begin{bmatrix}\hat{\mu}_{11}\\ \hat{\mu}_{21}\\ \hat{\mu}_{31}\\ \hat{\mu}_{41}\end{bmatrix} = \left(\left[\beta_{factor}\right]\bullet\left[Genotype\right]\right)\otimes\begin{bmatrix}f_{11}\\ f_{21}\\ f_{31}\\ f_{41}\end{bmatrix}+\begin{bmatrix}m_{11}\\ m_{21}\\ m_{31}\\ m_{41}\end{bmatrix}$$

- Estimate a factor level beta
- Use the factor loadings as weights
- Add the uncorrected or grand mean
- 1 df

Green Tree Frog

Factor level association

$$\begin{bmatrix}\hat{\mu}_{11}\\ \hat{\mu}_{21}\\ \hat{\mu}_{31}\\ \hat{\mu}_{41}\end{bmatrix} = \left(\left[\beta_{factor}\right] \bullet \left[Genotype\right]\right) \otimes \begin{bmatrix}f_{11}\\ f_{21}\\ f_{31}\\ f_{41}\end{bmatrix} + \begin{bmatrix}m_{11}\\ m_{21}\\ m_{31}\\ m_{41}\end{bmatrix}$$

Dot product
It's just a * in R

Green Tree Frog

Factor level association

$$\begin{bmatrix} \hat{\mu}_{11} \\ \hat{\mu}_{21} \\ \hat{\mu}_{31} \\ \hat{\mu}_{41} \end{bmatrix} = \left(\left[\beta_{factor}\right] \bullet \left[Genotype\right]\right) \otimes \begin{bmatrix} f_{11} \\ f_{21} \\ f_{31} \\ f_{41} \end{bmatrix} + \begin{bmatrix} m_{11} \\ m_{21} \\ m_{31} \\ m_{41} \end{bmatrix}$$

Kronecker product
The OpenMx symbol for this is %x%

Green Tree Frog

Variable specific association

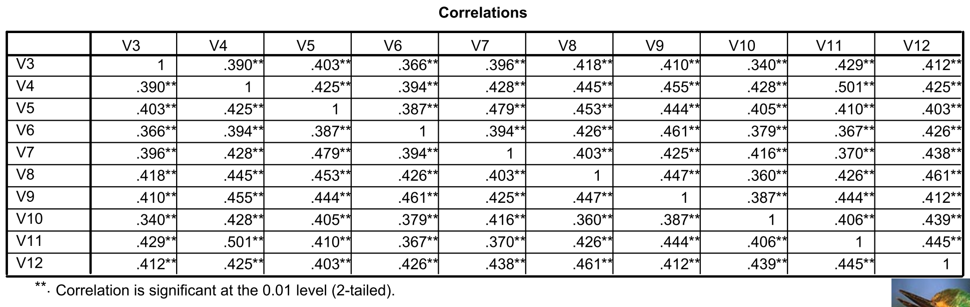
$$\begin{bmatrix} \hat{\mu}_{11} \\ \hat{\mu}_{21} \\ \hat{\mu}_{31} \\ \hat{\mu}_{41} \end{bmatrix} = \left([Genotype] \otimes \begin{bmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \\ \beta_4 \end{bmatrix} \right) + \begin{bmatrix} m_{11} \\ m_{21} \\ m_{31} \\ m_{41} \end{bmatrix}$$

- Estimate a separate beta for each trait
- Add the uncorrected or grand mean
- *n* df

Kookaburra

Simulated data set

- 10 traits, Moderately correlated ~.4
- 1 snp, MAF .2
- 500 individuals



Correlations

	V3	V4	V5	V6	V7	V8	V9	V10	V11	V12
V3	1	.390**	.403**	.366**	.396**	.418**	.410**	.340**	.429**	.412**
V4	.390**	1	.425**	.394**	.428**	.445**	.455**	.428**	.501**	.425**
V5	.403**	.425**	1	.387**	.479**	.453**	.444**	.405**	.410**	.403**
V6	.366**	.394**	.387**	1	.394**	.426**	.461**	.379**	.367**	.426**
V7	.396**	.428**	.479**	.394**	1	.403**	.425**	.416**	.370**	.438**
V8	.418**	.445**	.453**	.426**	.403**	1	.447**	.360**	.426**	.461**
V9	.410**	.455**	.444**	.461**	.425**	.447**	1	.387**	.444**	.412**
V10	.340**	.428**	.405**	.379**	.416**	.360**	.387**	1	.406**	.439**
V11	.429**	.501**	.410**	.367**	.370**	.426**	.444**	.406**	1	.445**
V12	.412**	.425**	.403**	.426**	.438**	.461**	.412**	.439**	.445**	1

**. Correlation is significant at the 0.01 level (2-tailed).

Kingfisher

readdrd4.R familydrd2.txt

```
#
#  OpenMx for ordinal factor analysis of drug data
#

# Load packages

require(psych)
require(OpenMx)
require(polycor)

# Read in original data

drd2data<-read.table('familydrd2.txt',header=T,na.strings='.')
describe(drd2data)
```

Numbat

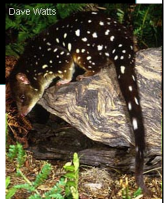
readdrd4.R familydrd2.txt

```
# Ignore correlated status of sibs (naughty naughty) to build big sample
size
sib1<-drd2data[,c(8,9,16:18)]
sib2<-drd2data[,c(10,11,19:21)]
sib3<-drd2data[,c(12,13,22:24)]
sib4<-drd2data[,c(14,15,25:27)]
names(sib1)<-c("SNP1","SNP2","Stimulants","Tranquilizers","Marijuana")
names(sib2)<-names(sib1)
names(sib3)<-names(sib1)
names(sib4)<-names(sib1)

sibsasInd<-rbind(sib1,sib2,sib3,sib4)
```

Perentie Monitor

Set Up Factor Model

```
# Retain only those cases who are not missing on all variables
sibsWithData<-subset(sibsasInd,!(is.na(Stimulants) & is.na(Tranquilizers) & is.na
(Marijuana)))
sibsWithSNPs<-subset(sibsWithData,!is.na(SNP1) & !is.na(SNP2))
# Take a look at the correlations in two ways
cor(sibsWithData,use="pairwise")
tetrachoric(sibsWithData)

# Now try to build and fit factor model
# Step 5: make the ordinal variables into R factors
ordinalData <- cbind(mxFactor(sibsWithSNPs[,3:5],levels=c(0:1)),sibsWithSNPs[,1:2])
nVariables <- 3
nFactors <- 1
nThresholds <- 1
obsVarNames <- names(ordinalData)[1:3]
```



Quoll

Set up Factor Model

```
 Fit Factor Model with Raw Ordinal Data and Matrices Input
# -----------------------------------------------------------------------
oneFactorThresholdModel <- mxModel('oneFactorThresholdModel',

    mxMatrix( type="Full", nrow=nVariables, ncol=nFactors, free=TRUE, values=0.2,
lbound=-.99, ubound=.99, name="facLoadings" ),
    mxMatrix( type="Unit", nrow=nVariables, ncol=1, name="vectorofOnes" ),
    mxMatrix( type="Zero", nrow=1, ncol=nVariables, name="Zeroes" ),
    mxMatrix( type="Full", nrow=nThresholds, ncol=nVariables, free=TRUE, values=.2,
dimnames=list(c(), obsVarNames),
              name="thresholdDeviations" ),
    mxMatrix( type="Lower", nrow=nThresholds, ncol=nThresholds, free=FALSE,
values=1, name="unitLower" ),
```

Red back spider

readdrd4.R familydrd2.txt

```
    mxAlgebra( expression=vectorofOnes - (diag2vec(facLoadings %*% t
(facLoadings))) , name="resVariances" ),
    mxAlgebra( expression=facLoadings %*% t(facLoadings) + vec2diag(resVariances),
name="expCovariances" ),
    mxAlgebra( expression=unitLower %*% thresholdDeviations,
name="expThresholds" ),
    mxAlgebra( expression=Zeroes, name="expMeans" ),

    mxData( observed=ordinalData, type='raw' ),
    mxFIMLObjective( covariance="expCovariances", means="expMeans",
dimnames=obsVarNames, thresholds="expThresholds" )
)

oneFactorThresholdFit <- mxRun(oneFactorThresholdModel, suppressWarnings=TRUE)
summary(oneFactorThresholdFit)
```

Red back spider

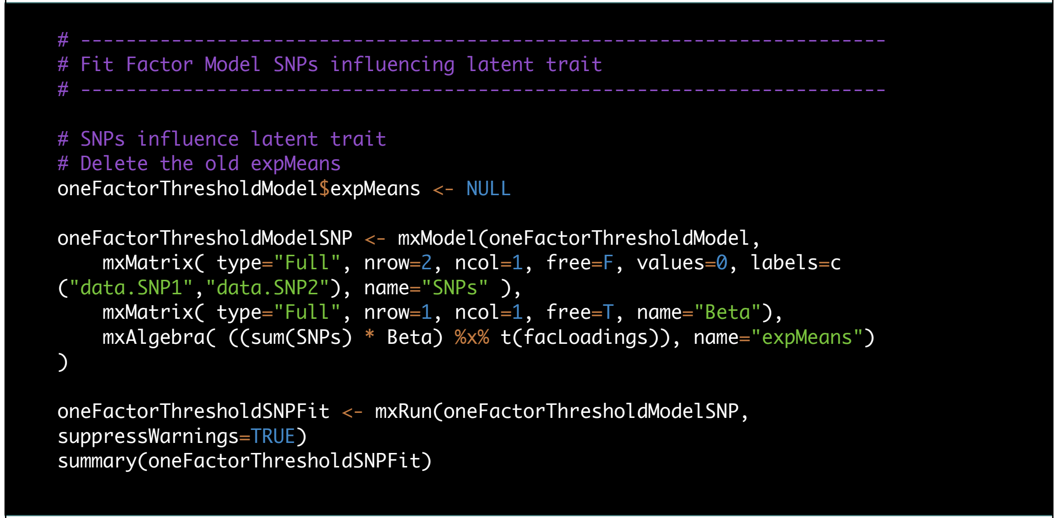
readdrd4.R familydrd2.txt

```
oneFactorThresholdFit <- mxRun(oneFactorThresholdModel,
suppressWarnings=TRUE)
summary(oneFactorThresholdFit)

# On executing this command you should, in principle, get some output!!!
```

Red back spider

```
free parameters:
  name           matrix row        col   Estimate  Std.Error lbound ubound
1 <NA>        facLoadings   1           1 0.93993707 0.04287723  -0.99   0.99
2 <NA>        facLoadings   2           1 0.80496549 0.04152029  -0.99   0.99
3 <NA>        facLoadings   3           1 0.71703124 0.04061479  -0.99   0.99
4 <NA> thresholdDeviations   1    Stimulants 1.45122759 0.04444629
5 <NA> thresholdDeviations   1 Tranquilizers 1.43170707 0.04380764
6 <NA> thresholdDeviations   1     Marijuana 0.08953097 0.02960041

observed statistics:  5389
estimated parameters:  6
degrees of freedom:  5383
-2 log likelihood:  3995.851
saturated -2 log likelihood:  NA
number of observations:  1804
chi-square:  NA
p:  NA
Information Criteria:
    df Penalty Parameters Penalty Sample-Size Adjusted
AIC  -6770.149          4007.851                NA
BIC -36364.600          4040.838          4021.776
```

variablespecific.mx… dataset.ped



```
# -----------------------------------------------------------------------
# Fit Factor Model SNPs influencing latent trait
# -----------------------------------------------------------------------

# SNPs influence latent trait
# Delete the old expMeans
oneFactorThresholdModel$expMeans <- NULL

oneFactorThresholdModelSNP <- mxModel(oneFactorThresholdModel,
    mxMatrix( type="Full", nrow=2, ncol=1, free=F, values=0, labels=c
("data.SNP1","data.SNP2"), name="SNPs" ),
    mxMatrix( type="Full", nrow=1, ncol=1, free=T, name="Beta"),
    mxAlgebra( ((sum(SNPs) * Beta) %x% t(facLoadings)), name="expMeans")
)

oneFactorThresholdSNPFit <- mxRun(oneFactorThresholdModelSNP,
suppressWarnings=TRUE)
summary(oneFactorThresholdSNPFit)
```

readdrd4.R familydrd2.txt

```
# -----------------------------------------------------------------------
# Fit Factor Model SNPs influencing latent trait
# -----------------------------------------------------------------------

# SNPs influence latent trait
# Delete the old expMeans
oneFactorThresholdModel$expMeans <- NULL

oneFactorThresholdModelSNP <- mxModel(oneFactorThresholdModel,
    mxMatrix( type="Full", nrow=2, ncol=1, free=F, values=0, labels=c
("data.SNP1","data.SNP2"), name="SNPs" ),
    mxMatrix( type="Full", nrow=1, ncol=1, free=T, name="Beta"),
    mxAlgebra( ((sum(SNPs) * Beta) %x% t(facLoadings)), name="expMeans")
)

oneFactorThresholdSNPFit <- mxRun(oneFactorThresholdModelSNP,
suppressWarnings=TRUE)
summary(oneFactorThresholdSNPFit)
```

readdrd4.R familydrd2.txt

```
# SNPs influence each substance
# Delete the old expMeans
oneFactorThresholdModel$expMeans <- NULL

oneFactorThresholdModelSNPres <- mxModel(oneFactorThresholdModelSNP,
    mxMatrix( type="Full", nrow=1, ncol=nVariables, free=T, name="Beta"),
    mxAlgebra( sum(SNPs) %x% Beta, name="expMeans")
)

oneFactorThresholdSNPresFit <- mxRun(oneFactorThresholdModelSNPres,
suppressWarnings=TRUE)
summary(oneFactorThresholdSNPresFit)
```

Latent Class Analysis

```
# Fit Latent Class Model with Raw Ordinal Data and Matrices Input
# -----------------------------------------------------------------------

nVar<-3
nClass<-2

# -----------------------------------------------------------------------
# Create an MxModel object
# -----------------------------------------------------------------------
class1 <- mxModel("Class1",
    mxMatrix( type="Iden", nrow=nVar, ncol=nVar, name="S" ),
    mxMatrix( type="Full", nrow=1, ncol=nVar, values=c(0.1,0.6,0.9), free=T,
name="Thresholds" ),
    mxMatrix( type="Zero", nrow=1, ncol=nVar, name="Zeroes" ),
    mxData( observed=ordinalData, type='raw' ),
    mxFIMLObjective(covariance="S", means="Zeroes", dimnames=names
(ordinalData[,1:3]), thresholds="Thresholds", vector=T)
)
```

Latent Class Analysis

```
# Make a class 2 model that looks pretty much the same as class 1 but has a
different name
class2 <- mxModel(class1, name="Class2" )

# Nudge class2 endorsement probs away from those of class 1
class2@matrices$Thresholds@values[]<-0

# make a matrix of class probabilities
classP <- mxMatrix("Full", nClass, 1, free=c(TRUE, FALSE), values=2,
lbound=0.001, labels = c("p1", "p2"), name="Props")

# standardize them
classS <- mxAlgebra(Props%x%(1/sum(Props)), name="classProbs")
```

Latent Class Analysis

```
# build the algebra to compute -2lnL where L is sum of (wi * L(items |
class=i) ) with wi being class i's memb prob
algObj <- mxAlgebra(-2*sum( log(classProbs[1,1]%x%Class1.objective + classProbs
[2,1]%x%Class2.objective)), name="mixtureObj")

# make a mxAlgebraObjective object
obj <- mxAlgebraObjective("mixtureObj")

# bundle all the bits together into one mxModel
lcaModel <- mxModel("Latent Class Model",
                    mxData( observed=ordinalData[,1:3], type='raw' ),class1,
class2, classP, classS, algObj, obj )

lcaModelFit <- mxRun(lcaModel, suppressWarnings=TRUE)
summary(lcaModelFit)
# Take a look at some of the output
lcaModelFit$classProbs
```

Latent Class Analysis

```
# Modify model to allow SNPs to affect classProbs
# Trick is to put Definition variables into a matrix to evaluate classProps
accordingly

SNPsums <- as.matrix(ordinalData[,4]) + as.matrix(ordinalData[,5])
U <- mxMatrix( "Unit", nrow=dim(ordinalData)[1], ncol=1, free=F, name="U")
U22 <- mxMatrix( "Unit", nrow=2, ncol=2, name="U22")
SNP <- mxMatrix( type="Full", nrow=dim(ordinalData)[1], ncol=1, free=F,
values=SNPsums, name="SNP")
Beta <- mxMatrix( type="Full", nrow=1, ncol=2, free=c(T,F), name="Beta")
BetaSNP <- mxAlgebra( SNP %x% Beta, name="BetaSNP")
UProp <- mxAlgebra( U %x% t(Props), name="UProp")

classS <- mxAlgebra( (UProp + BetaSNP) / ((UProp + BetaSNP) %*% U22),
name="classProbs")
algObj <- mxAlgebra(-2*sum( log(classProbs[,1] * Class1.objective + classProbs[,2] *
Class2.objective)), name="mixtureObj")

lcaModelSNP <- mxModel("Latent Class Model",
                    mxData( observed=ordinalData[1:3,], type='raw' ),class1, class2,
classP, classS, SNP, Beta, BetaSNP, U, U22, UProp, algObj, obj )
lcaModelSNPFit <- mxRun(lcaModelSNP)
summary(lcaModelSNPFit)
```

Tree Kangaroo

Wombat

Tiger Snake

Tassie Tiger

Thorny Devil

Zebra finch

Tassie Devil