mimR – A package for graphical modelling in **R**

Søren Højsgaard*

June 24, 2003

Abstract

The **mimR** package for graphical modelling in **R** is introduced. We present some facilities of **mimR**, namely those relating specifying models, editing models and doing model search. We also discuss the entities needed for flexible graphical modelling in terms of an object structure.

1 Introduction and background

The **mimR** package provides facilities for graphical modelling in the statistical program **R**[1]. The **mimR** package has its own homepage[2] and is furthermore a part of the gR–project[3] which is a project to make graphical models available in **R**.

The statistical foundation is Mixed Interaction Models, a very general class of statistical models for mixed discrete and continuous variables. Statistical inference in mixed interaction models can be made be the stand–alone program **MIM**[4], and the core of **mimR** is an interface from **R** to the **MIM** program. The reader is assumed familiar with mixed interaction models, and to have a working knowledge of the **MIM** program. Edwards (2000) described both in a very clear way. For a comprehensive account of graphical models we refer to Lauritzen (1996). Other important references are Edwards (1990) and Lauritzen and Wermuth (1984).

2 Preliminaries

2.1 Mixed Interaction Models *à vol d'oiseau*

Mixed interaction models include as special cases log–linear models for contingency tables and covariance selection models for the multivariate normal distribution. More

*Biometry Research Unit, Danish Institute of Agricultural Sciences, Research Centre Foulum, DK–8830 Tjele, Denmark. E–mail: sorenh@agrsci.dk

[1]available from `www.r-project.org`

[2]at `http://www.jbs.agrsci.dk/~sorenh/mimR`

[3]`www.r-project.org/gR`

[4]available from `www.hypergraph.dk`

importantly, however, is that the models allow a simultaneous modelling of discrete *and* continuous variables. Focus in mixed interaction models is often (although not exclusively) on conditional independence restrictions.

Within mixed interaction models, one can treat problems where all variables are treated on equal footing (i.e. there are no distinction between variables as being explanatory or responses). Such models are below referred to as *undirected models*. It is however also possible to work with problems where some variables are purely explanatory, other are purely responses and others play both roles. Such models are denoted *block recursive models*. Block recursive models contain undirected models as special cases, and – perhaps more importantly – can be established by a careful combination of undirected models through conditioning.

2.2 MIM as inference engine

From the users perspective, the **MIM** stand alone program can be regarded as an "inference engine" with which the user (at least in principle) needs not be concerned with. However, in reality the **mimR** package is not yet at such a mature level, and this implies that the user in some cases should be aware that there is a separate programming running with which **R** communicates.

2.3 Getting help

In addition to the documentation in the **mimR** package, the **MIM** program itself contains a comprehensive help function which the user of **mimR** is encouraged to make use of.

2.4 Ways of acessing MIM from R

There are two levels of acessing **MIM** from **R**.

- The "high level approach" is by creating model objects much like one does when working with linear, generalized linear and other models in **R**. This approach is the core contents of this paper.
- The "low level approach" is by sending commands directly to **MIM** using the `mim.cmd` and the `mcm` functions, see Appendices A.1 and A.2.

3 The `rats` dataset

Some features of **mimR** will be illustrated in the present paper on the basis of the `rats` dataset. The `rats` dataset is from a hypothetical drug trial, where the weight losses of male and female rats under three different drug treatments have been measured after one and two weeks. See Edwards (2000) for more details. The first rows of the data are:

```
   Sex Drug W1 W2
1    M   D1  5  6
2    M   D1  7  6
3    M   D1  9  9
4    M   D1  5  4
5    M   D2  9 12
6    M   D2  7  7
7    M   D2  7  6
8    M   D2  6  8
     ............
```

4 gmData objects – graphical meta data

A gmData object contains information about variables, their labels, their levels (for the discrete variables) etc. A gmData object may also contain data, but need not do so. gmData can be taken to be short for "graphical model data" or "graphical meta data".

4.1 Creating a gmData object manually

A gmData object (without data) can be created by

```
gmd.rats.nodata  <-  gmData(c("Sex","Drug","W1","W2"),
    factor=c(2,3,FALSE,FALSE),
    vallabels=list(c("M","F"), c("D1","D2","D3")))
```

The gmData object looks like

```
  name letter factor levels
1  Sex      a   TRUE      2
2 Drug      b   TRUE      3
3   W1      c  FALSE     NA
4   W2      d  FALSE     NA
Data origin:      no.data
To see the values of the factors use the 'vallabels' function
To see the data use the 'observations' function
```

To each variable, there is associated a letter. It is possible use the letters in specifying models, see the examples below.

4.2 Making a gmData object from a data frame or a table

Typically one will create a gmData object (with data) from a data frame (or a table) as follows:

```
data(rats)
gmd.rats <- as.gmData(rats)
data(HairEyeColor)
gmd.hec  <- as.gmData(HairEyeColor)
```

5 Models in mimR

Currently, only undirected models are available in **mimR**. That is models where a possible response structure has not been accounted for.

An undirected model is created using the `mim` function (which returns a `mim` object). For example:

```
m1 <- mim("Sex:Drug/Sex:Drug:W1+Sex:Drug:W2/W1:W2", data=gmd.rats)
m2 <- mim("ab/abc+abd/cd", data=gmd.rats, letter=TRUE)
```

It is possible to specify the main effects, the saturated and the homogeneous saturated models (possibly for only a subset of the variables) in short form:

```
m.main <- mim(".",  data=gmd.rats, marginal=c("Sex", "Drug", "W1"))
m.sat  <- mim("*",  data=gmd.rats, marginal=c("Sex", "Drug", "W1"))
m.hsat <- mim("*h", data=gmd.rats, marginal=c("Sex", "Drug", "W1"))
```

6 Model fitting

The models created above are not fitted to data. For model fitting two functions are available: `fit` and `emfit` (`emfit` will be discussed later).

```
m1f  <- fit(m1)
m1f
Model: Sex:Drug/Sex:Drug:W1+Sex:Drug:W2/W1:W2
Model(letter): ab/abc,abd/cd
likelihood: 273.705 DF: 15
```

7 Model selection and model editing

7.1 Editing models directly

Models can be edited directely, using the `editMIM` function by which one can 1) delete, 2) add and 3) homogeneously add interactions:

```
m.main <- mim(".",  data=gmd.rats)
m2 <- editmim(m.main, add=c("Sex:Drug", "Sex:W2"))
m3 <- editmim(m.main, add=c("Sex:Drug", "Sex:W2"),hadd="Drug:W1:W2")
```

7.2 Stepwise model selection

To a `mimModel` object the function `stepwise` applies which takes as additional arguments all arguments that the STEPWISE command in **MIM** does. The `stepwise` function returns a new `mimModel` object.

```
data(carcass)
gmd.carc <- as.gmData(carcass)
m.main <- mim(".",  data=gmd.carc)
m.sat  <- mim("*",  data=gmd.carc)
m.m <- stepwise(m.main, "f")    # forward
m.s <- stepwise(m.sat,  "s")    # backward, exact tests
```

8 Fitted values (parameter estimates)

The fitted values (parameters estimates) can be obtained using the `fitted` function:

```
mf2 <- fit(m2)
parms <- fitted(mf2)
parms
  Drug Sex Counts   W1    W2  W1:W1 W1:W2 W2:W1 W2:W2
1    1   1      4 9.75 8.500 17.104     0     0 5.583
2    2   1      4 9.75 8.500 17.104     0     0 5.583
3    3   1      4 9.75 8.500 17.104     0     0 5.583
4    1   2      4 9.75 8.833 17.104     0     0 9.639
5    2   2      4 9.75 8.833 17.104     0     0 9.639
6    3   2      4 9.75 8.833 17.104     0     0 9.639
```

9 Simulating data from a fitted model

Simulating data from a fitted model can be done by the `simulate` function:

```
samp <- simulate(mf2, size=10) # 'sample' is already used in R
```

10 Obtaining the linear predictor

The `linpredict` function can be used to get the linear predictor for a set y given another set x (possibly empty) of variables, for example

```
d1  <- linpredict(mf2, y="W2",    x="W1:Sex")
d2  <- linpredict(mf2, y="Sex",   x="W1:W2")
d1
Distribution of W2 given W1:Sex
Sex=1
   int W1
W2 8.5  0
        W2
W2 5.58333
Sex=2
       int W1
W2 8.83333  0
        W2
W2 9.63922
d2
Distribution of Sex given W1:W2
  Sex Constant       W1      W2
1   1  0.00000  0.00000 0.00000
2   2 -0.37203 -0.01595 0.06087
```

11 Missing values and/or latent variables

To fit a model with to incomplete data or to fit a latent variable model, use the `emfit` function. See e.g. the Example in Section 12.

12 Example – Mathematics marks

This dataset (taken from Mardia, Kent and Bibby (1979)) contains the examination marks for 88 students in 5 different subjects. Data is contained the data set `mathmark` in the **mimR** package. Edwards (2000) also investigates these data.

We start out by specifying the saturated model and do a backward elimination:

```
data(mathmark)
gmd.math <- as.gmData(mathmark)
math1 <- mim("*", data=gmd.math)
math2 <- stepwise(math1)
math2
Formula: //mechanics:vectors:algebra+algebra:analysis:statistics
Formula(letter): //abc,cde
likelihood: 3391.021 DF: 4
```

The model `math2` is shown in Figures 1.

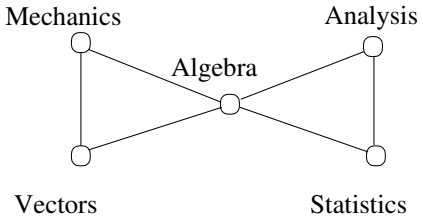

Figure 1: The “butterfly” model selected for the mathmarks data.

Next we consider a latent variable model: We suppose that there is a latent binary variable `L` such that the manifest variables are all conditionally independent given `L`. We fit such a model by:

```
math     <- mathmark
math$L   <- factor(NA, levels=1:2)
gmd.math <- as.gmData(math)
latent(gmd.math) <- "L"
m1  <- mim("*", data=gmd.math)
m2  <- editmim(m1, del=paste(names(math)[1:5],":",collapse=''))
m2f <- emfit(m2)
EM algorithm: random start values.
 Cycle -2*Loglikelihood      Change
   1          3580.5111
   2          3543.7595  -36.751687
   3          3476.1469  -67.612538
   ...................
  19          3454.9348   -0.000070
Successful convergence.
d.imp <- retrieveData(gmd.math,impute=TRUE)
```

We plot the predicted value of `L` against the observation number:

```
plot(d.imp$L)
```

The plot is shown in Figure 2. The grouping of the values of `L` suggests that data have been processed somehow prior to presentation. Edwards (2000), p. 181, conclude: “Certainly they (the data) have been mistreated in some way, doubtless by a statistician.”

13 Miscellaneous

mimR mailing list If you wish to be informed about updates of **mimR**, please send me an e–mail (to sorenh@agrsci.dk).

Availability **mimR** is available only on Windows platforms because **MIM** only runs on Windows platforms.

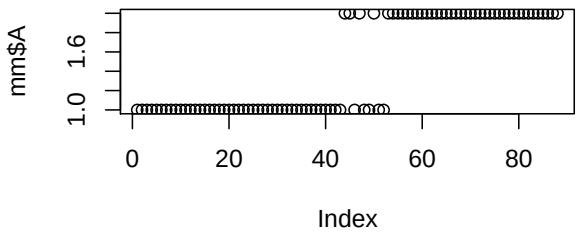

Figure 2: An index plot of the discrete latent variables.

mimR and Splus The current version of **mimR** is known NOT to run under Splus. If sufficient interest appears, it may be considered to remedy this situation.[5]

14 Using the correct version of MIM and R

To use **mimR**, the **MIM** program must be installed on your computer (Windows only). **MIM** (including a free student version and free updates) is available from http://www.hypergraph.dk.

Upgrades of **MIM** are frequently released. It is IMPORTANT to make sure that your version of **MIM** is in accordance with what **mimR** expects. When loading the **mimR** package using `library(mimR)` a message similar to the one below appears in **R**. From this one sees the earliest version of **MIM** (and **R**) with which the current version of **mimR** works.

```
------------------------------------------------------------
mimR: An R interface to MIM for graphical modelling in R
mimR, version 1.0 is now loaded
Copyright (C) 2002, Søren Højsgaard
Maintained by Søren Højsgaard <sorenh@agrsci.dk>
Webpage: http://www.jbs.agrsci.dk/~sorenh/mimR

Built: R 1.6.1; Win32; Tue Nov 26 11:36:23 2002
NOTICE:
o To use mimR the MIM program must be running.
o The current version of mimR requires MIM version 3.1.2.9 or later
o MIM (including a free student version and free upgrades)
  is available from http://www.hypergraph.dk.
o mimR is available on Windows platforms only
------------------------------------------------------------
```

[5] CHANGE THIS!!!

15 Acknowledgements

David Edwards (the creator of **MIM**) is greatly acknowledged for his support in the creation of **mimR**. Also the members of the gR project are acknowleded for their inspiration.

References

Edwards, D. (1990). Hierarchical interaction models, *Journal of the Royal Statistical Society, Series B* **52**(1): 3–20.

Edwards, D. (2000). *Introduction to Graphical Modelling*, 2nd edition edn, Springer Verlag, New York.

Lauritzen, S. L. (1996). *Graphical Models*, Oxford University Press.

Lauritzen, S. L. and Wermuth, N. (1984). Mixed interaction models, *Technical Report R 84-8*, Institute for Electronic Systems, Aalborg University.

Mardia, K. V., Kent, J. T. and Bibby, J. M. (1979). *Multivariate Analysis*, Academic Press.

A Low level access to MIM from R

A.1 Primitive use of MIM from R – the `mim.cmd()` function

The core of **mimR** is the `mim.cmd` function. The arguments to `mim.cmd` are simply **MIM** commands (given as strings). For example:

```
>mim.cmd("fact a2 b2; statread ab; 25 2 17 8 !")
>mim.cmd("mod a,b; fit; print; print f")
```

The `mim.cmd` function returns the result of the commands submitted to **MIM**. The result of the last call of `mim.cmd` above is:

```
Deviance:       5.3111 DF: 1
The current model is: a,b.
Fitted counts, means and covariances.
 a b   Count
 1 1  21.808
 1 2   5.192
 2 1  20.192
 2 2   4.808
```

This is exactly the result that is printed in the **MIM** window. It is shown below how to make the output from `mim.cmd` tangible for further work in **mimR**.

A.2 Using MIM directly from mimR– the `mcm()` function

The `mcm` function (short for "**MIM** command mode") provides a direct interface to **MIM**, i.e. the possibility to write **MIM** commands directly. The `mcm` function returns no value to **R**, and is intended only as an easy way to submit **MIM** commands without the overhead of wrapping them into the `mim.cmd` function (or submitting the commands directly to **MIM**).

Hence, using `mcm`, the session above would be:

```
> mcm()
Enter MIM commands here. Type quit to return to R
MIM->fact a2 b2; statread ab
MIM->25 2 17 8 !
Reading completed.
MIM->mod a,b; fit
Deviance:       5.3111 DF: 1
MIM->print; print f
The current model is: a,b.
Fitted counts, means and covariances.
 a b   Count
 1 1  21.808
 1 2   5.192
 2 1  20.192
 2 2   4.808
MIM->quit
>
```

To return to **R** from the `mcm` function type 'quit', 'exit', 'end', 'q' or 'e' (i.e. the commands one would use to terminate **MIM**). These commands, however, do not terminate **MIM** – they only return control to **R**.