The **mimR**[*] package for Graphical Modelling in **R**

Søren Højsgaard[†]

February 9, 2004

Abstract

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

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 for **mimR** 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. Edwards (2000) describes the models and the **MIM** program 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 (1989).

The reader (and user of **mimR**) is assumed 1) familiar with mixed interaction models, and 2) to have a working knowledge of the **MIM** program.

2 Preliminaries

2.1 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.[5]

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

[*] Version 1.1.4

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

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

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

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

[4] available from `http://www.hypergraph.dk`

[5] To access the help function in **MIM** go to the **MIM** program and press F1.

with. However, in practice it is worth keeping in mind that **MIM** is a stand alone program: In **MIM** there can be only 1) one specification of a data set, 2) one data set and 3) one model at any time. This means that in general when fitting a new model, a lot of information may have to be conveyed to **MIM**, and this may take some time (if the data set is large). Likewise, receiving fitted values from **MIM** to **R** may take some time too.

2.3 Ways of accessing MIM from R

There are two levels of accessing **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 C.1 and C.2.

3 The `rats` dataset

Some features of **mimR** will be illustrated in the present paper on the basis of the `rats` dataset in the **mimR** package. 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 Objects in mimR

The core of **mimR** are the `gmData` and `mim` objects.

`gmData` **objects:** 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. The name "`gmData`" can be taken to be short for "graphical model data" or "graphical meta data" (where we prefer the latter). The idea behind separating the specification of the variables from data is that some properties of a model can be investigated without any reference to data, for example decomposability and collapsibility.

`mim` **objects:** Links a model formula to a `gmData` object. Since a `gmData` object need not contain any data, fitting a `mim` model is separate process. (This is an important difference between model in **mimR** and e.g. linear models in **R**.) When the model has been fitted (provided that there are data in the `gmData` object), the `mim` object also contains the fitted values, parameter estimates etc.

5 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.

5.1 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")))
> gmd.rats.nodata
```

```
  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:      table
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. This letter is used in connection with the internal representation of models and variables in MIM and the user should not be concerned with this. It is possible use the letters in specifying models (see the examples below) but it is not recommended.

5.2 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)
> gmd.rats
```

```
  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:      data.frame
To see the values of the factors use the 'vallabels' function
To see the data use the 'observations' function
```

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

```
  name letter factor levels
1 Hair      a   TRUE      4
2  Eye      b   TRUE      4
3  Sex      c   TRUE      2
Data origin:      table
To see the values of the factors use the 'vallabels' function
To see the data use the 'observations' function
```

6 Models in **mimR**

Currently, only undirected models are available in **mimR**. That is, models in which all variables are treated on equal footing as response variables. (Thus models where a possible response structure has not been accounted for can currently not be dealt with in **mimR**).

An undirected model is created using the `mim` function (which returns a `mim` object). The following two specifications are equivalent:

```
> 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 1) the main effects, 2) the saturated and 3) 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"))
```

7 Model fitting

Model fitting is separated from model specification, so 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)
```

```
Deviance:      27.8073 DF: 15
```

```
> m1f
```

```
Formula: Sex:Drug/Sex:Drug:W1+Sex:Drug:W2/W1:W2
likelihood: 273.705 DF: 15
```

8 Model summary

A summary (including certain model properties) of a `mim` can be achieved using the `summary()` function:

```
> summary(m1f)
```

```
Formula: Sex:Drug/Sex:Drug:W1+Sex:Drug:W2/W1:W2
Formula(letter): ab/abc,abd/cd
deviance: 27.8073 DF: 15 likelihood: 273.705
Model properties:
 Variables in model:        Sex Drug W1 W2
 Is graphical:              TRUE
 Is decomposable:           TRUE
 Is mean linear:            TRUE
 Is homogeneous:            TRUE
 Is delta-collapsible:      TRUE
 Degrees of freedom:        15
 Cliques:
[1] "Sex:Drug:W1:W2"
```

9 Model selection and model editing

9.1 Editing models directly

Models can be edited directly, 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")
```

9.2 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")
> m.s <- stepwise(m.sat, "s")
```

The selected models are:

```
> m.m

Formula: //F11:F12:M12:F13+F11:F12:M12:M13+F11:F12:M13:LMP+M11:M12:M13
likelihood: 11405.13 DF: 7

> m.s

Formula: //F11:M11:F12:M12:M13+F11:M11:F12:F13:LMP+F11:M11:F12:M13:LMP
likelihood: 11370.74 DF: 3
```

10 Fitted values (parameter estimates)

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

```
> mf2 <- fit(m2)

Deviance:      92.2956 DF: 24

> parms <- fitted(mf2)
> parms

  Drug Sex Freq   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
```

11 Simulating data from a fitted model

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

```
> samp <- simulate(mf2, size = 10)
```

12 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

```
> linpredict(mf2, y = "W2", x = "W1:Sex")

Entering sufficient statistics... done
Deviance:      92.2956 DF: 24
Distribution of W2 given W1:Sex
Sex=1
   int W1
W2 8.5  0
         W2
W2 5.583333
Sex=2
        int W1
W2 8.833333  0
         W2
W2 9.638889

> linpredict(mf2, y = "Sex", x = "W1:W2")

Entering sufficient statistics... done
Deviance:      92.2956 DF: 24
Distribution of Sex given W1:W2
  Sex Constant W1         W2
1   1 0.000000  0  0.0000000
2   2 2.149589  0 -0.6059615
```

13 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 14.

14 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
likelihood: 3391.021 DF: 4
```

The model `math2` is shown in Figure 1.

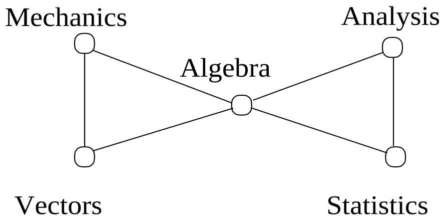

Figure 1: The "butterfly" model selected for the mathmarks data.

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"
```

With the specification above, `L` is a binary variable consisting of `NA` values. The command `latent(gmd.math)<- "L"` makes it explicit in the `gmData` object, that `L` is indeed a latent variable. One consequence is, that the model can not be fitted using the `fit` function. Instead, the `emfit` function which uses the EM algorithm, Dempster, Laird, and Rubin (1977), must be used:

```
> m1 <- mim("*", data = gmd.math)
> m2 <- editmim(m1, del = paste(names(math)[1:5], ":", collapse = ""))
> m2f <- emfit(m2, plot = TRUE)
> d.imp <- retrieveData(impute = TRUE)
```

The argument `plot=TRUE` in `emfit()` creates the plot in Figure 2.

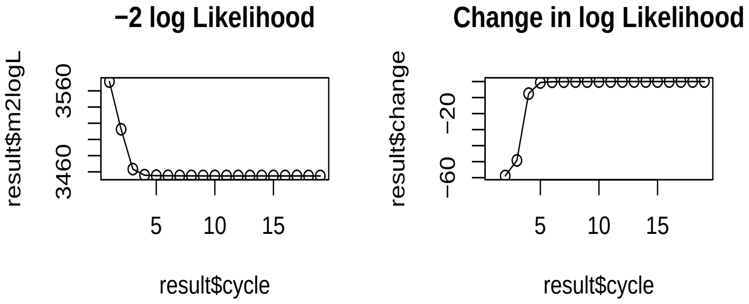

Figure 2: Convergence of the EM algorithm.

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

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

The plot is shown in Figure 3. 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.”

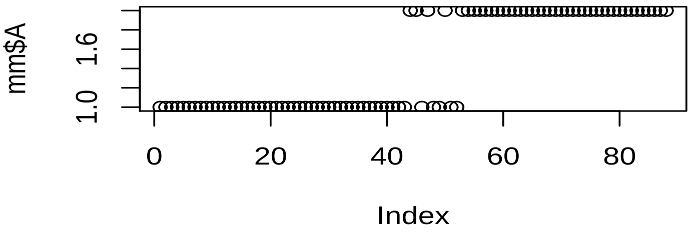

Figure 3: An index plot of the discrete latent variables.

The EM algorithm needs a set of initial values for the unobserved values to start from when calculating the first parameter estimates. It is well known, that the final estimate of the EM algorithm may depend on the initial values and that (especially in the case of latent variables) the likelihood may have multiple maxima. Default in the call of emfit is that **MIM** substitutes random values for the missing values. It is, however, possible to control the starting values of the EM algorithm as follows: The user can specify the values of L and subsequently declare L to be latent. In the call of emfit, the argument S causes the EM algorithm to take the specified values as starting point for the EM algorithm:

```
> math[, "L"] <- factor(1:2, 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, arg = "S", plot = TRUE)
```

For this specific case, it turns out that the result is very insensitive to the intial values.

An alternative analysis is to regard L as a continuous variable. To speed up the convergence of the EM algorithm, one can do a factor analysis to get good starting values:

```
> fa <- factanal(mathmark, factors = 1, scores = "regression")
> math[, "L"] <- fa$scores
> gmd.math <- as.gmData(math)
> latent(gmd.math) <- "L"
> m1 <- mim("*", data = gmd.math)
> m2 <- editmim(m1, del = paste(names(math)[1:5], ":", collapse = ""))
> m2f <- emfit(m2, arg = "S", plot = TRUE)
> d.imp <- retrieveData(impute = TRUE)
> plot(d.imp$L)
```

The plot of the imputed values for the latent variables are shown in Figure 4 and this also suggests that the data do not emerge in random order.

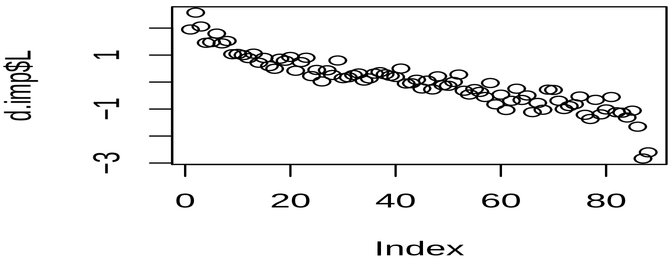

Figure 4: Plot of predicted value of L against index.

15 Discussion

In this paper we have 1) illustrated some aspects of **mimR** package for graphical modelling in **R**, and 2) presented preliminary ideas regarding an object structure for graphical modelling in **R**. It is the hope that **mimR** will be obsolete in a not too distant future – not because of lack of relevance of being able to work with graphical models in **R**. Rather, it is the hope that a more proper package with this functionality will be implemented as an integrated part of **R**. That is one of the aims of the gR–project. Until that happens we will continue to develop **mimR**. **mimR** is currently at level of development where it is likely that significant changes (e.g. of names of functions and/or object classes) will occur.

16 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 acknowledged for their inspiration.

References

A. P. Dempster, N. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm (with discussion). *Journal of the Royal Statistical Society, Series B*, 39:1–38, 1977.

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

David Edwards. *Introduction to Graphical Modelling*. Springer Verlag, New York, 2nd edition edition, 2000.

S. L. Lauritzen and N. Wermuth. Graphical models for associations between variables, some of which are qualitative and some quantitative. *Annals of Statistics*, 17:31–57, 1989.

Steffen L. Lauritzen. *Graphical Models*. Oxford University Press, 1996.

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

A Miscellaneous

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

mimR and Splus: The current version of **mimR** does not run under Splus. If sufficient interest appears, it may be considered to remedy this situation.

B Using the appropriate version of MIM and R

To use **mimR**, the **MIM** program must be installed on your computer (Windows only). **MIM** is available from `http://www.hypergraph.dk`. Upgrades of **MIM** are frequently released.

It is IMPORTANT to make sure that your version of **MIM** and **R** is in accordance with what **mimR** expects. When loading the **mimR** package using `library(mimR)` a message appears telling 1) which version of **MIM** that is expected and 2) with which version of **R** the current version of **mimR** has been checked.

C Low level access to MIM from R

C.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
```

C.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**.