mimR (version 1.0)

– An interface from **R** to **MIM** for graphical modelling in **R**

Søren Højsgaard*

December 16, 2002

Contents

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

1 Introduction and background

The **mimR** package provides an interface from **R** (`www.r-project.org`) to **MIM** (`www.hypergraph.dk`). Thereby one can use the functionality of **MIM** from within **R**. **mimR** grew out of the need for being able to work with (graphical) Mixed Interaction Models (as in **MIM**) within a general purpose statistical package (such as **R**). For information on mixed interaction models we refer to Edwards (1990) and Lauritzen and Wermuth (1984). For a comprehensive account of graphical models we refer to Lauritzen (1996). Mixed interaction models and the **MIM** program are described in Edwards (2000). The user is assumed familiar with the **MIM** program. The **mimR** package has its own homepage:

`http://www.jbs.agrsci.dk/~sorenh/mimR`

mimR is furthermore a part of the gR–project (see `www.r-project.org/gR`) which is a project to make graphical models available 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**. The immediate future of **mimR** also depends on whether the package is found useful in the **R** community: There is certainly room for improvements and enhancements of this first version of **mimR**. The extent to which this will happen depends on the response of the graphical modelling and **R** community.

2 Getting ready to use mimR

Use of **mimR** requires a few installation steps to be completed. These are described in the Sections 8.4 and 8.5. Note: Prior to using any function in **mimR**, make sure that **MIM** is running.

3 The mimR functionality

3.1 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(c("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**. For information on how the communication between **R** and **MIM** works, see Section 8.6.

3.2 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**.

3.3 Functions available in mimR

All other functions in **mimR** use the `mim.cmd` function, and then interprets the output from **MIM** and, whenever relevant, returns the output in an appropriate form (e.g. as lists of matrices etc.) in **R**.

The functions in **mimR** can be logically divided into three groups:

MIM command functions: (listed in Table 1) mimic **MIM** commands with the same name (apart from that the **mimR** functions all start with "`mim.`".

Enhancement functions: (listed in Table 2) provide additional functionality.

Internal functions: These a auxiliary functions for internal use only, and not described here.

4 MIM command functions

The functions in Table 1 all mimic **MIM** commands with the same name (apart from that the **mimR** functions all start with "`mim.`".

Table 1: **mimR** functions which mimic **MIM** commands with the same name.

mimR function	**MIM** command
mim.stepwise	stepwise
mim.testdelete	testdelete
mim.print	print
mim.display	display
mim.fit	fit
mim.emfit	emfit

Example 1 (**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 in the file `mathmark.dat` which comes with the **MIM** distribution. Suppose the file is located in `D:\`:

```
> mim.cmd("input d:\\mathmark.dat")
Reading completed.
```

```
> mim.cmd("show v")
Var  Label        Type  Levels In   In    Fixed  Block
                                Data Model
-------------------------------------------------------
 v  mechanics     cont      .    X    X      .      .
 w  vectors       cont      .    X    X      .      .
 x  algebra       cont      .    X    X      .      .
 y  analysis      cont      .    X    X      .      .
 z  statistics    cont      .    X    X      .      .
-------------------------------------------------------
> mim.cmd("model //vwxyz")
> mim.stepwise("sz")
Selected model: //vwx,xyz
> mim.cmd("test")
Test of H0: //vwx,xyz
against H:  //vwxyz
LR:   0.8957    DF:   4    P: 0.9252
> mim.testdelete("xw","s")
Test of H0: //vw,vx,xyz
against H:  //vwx,xyz
F:   20.4425    DF:   1,  85    P: 0.0000
> o1 <- mim.print("i")
> o1$cov
      v     w     x     y     z
v 1.000 0.332 0.235 0.000 0.000
w 0.332 1.000 0.327 0.000 0.000
x 0.235 0.327 1.000 0.451 0.364
y 0.000 0.000 0.451 1.000 0.256
z 0.000 0.000 0.364 0.256 1.000
> mim.display("wx","yz")
Parameters of the conditional distribution of  w,x given  y,z
     int     y     z
w 31.070 0.263 0.172
x 24.725 0.348 0.227
        w      x
w 133.322 34.404
x  34.404 45.607
```

5 Enhancement functions

Example 2 (**Continuation of Example 1**) We suppose there is a latent binary variable A and that the manifest variables are all independent conditional on A:

Table 2: Enhancement functions in **mimR**

mimR function	effect
mim.cmd	submit commands to **MIM**
mcm	"**MIM** command mode" for entering **MIM** commands directly
mim.read	enters a data frame into **MIM**
mim.statread	enters a set of sufficient statistics into **MIM**
mim.diary.data	gets a listing of data i **MIM** into **R** as a dataframe
make.mim.stats	creates a set of sufficient statistics (to be read into **MIM**).

```
> mim.cmd("fact A2; calc A=ln(0)")
> mim.cmd("model A/Av,Aw,Ax,Ay,Az/Av,Aw,Ax,Ay,Az")
> mim.emfit(plot=TRUE)
EM algorithm: random start values.
 Cycle -2*Loglikelihood      Change
   1          3580.5111
   2          3543.7595  -36.751687
   3          3476.1469  -67.612538
   4          3456.4479  -19.698980
   5          3455.2915   -1.156400
   ...................
  18          3454.9348   -0.000125
  19          3454.9348   -0.000070
Successful convergence.
```

Setting `plot=TRUE` in `mim.emfit()` creates the plot in Figure 1.

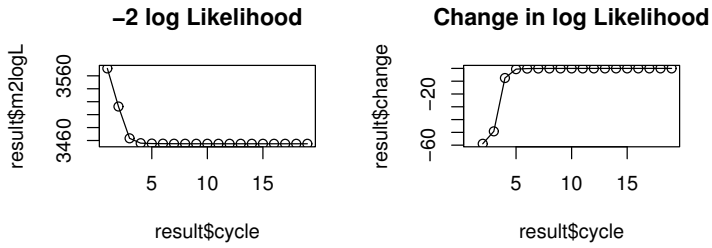

Figure 1: Convergence of the EM algorithm.

We save the predicted values of A and plot these against the observation number:

```
mim.cmd("impute")
m1 <- mim.print("d")
plot(m1$A)
```

The plot is shown in Figure 2. The grouping of the values of A 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."

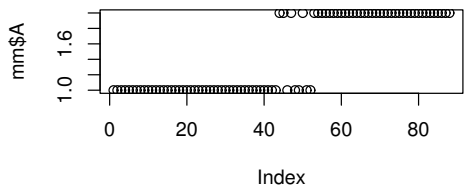

Figure 2: An index plot of the discrete latent variables.

6 MIM model objects

On top of the basic facilities described in the previous sections a preliminary system of model objects has been implemented.

6.1 `mimData` objects

A dataframe in **R** is turned into a `mimData` object which is the dataframe, a corresponding dataframe in which factors have been coded as 1,2,3..., and a table containing the name conversion between the original dataframe and the MIM dataframe. In addition `mimData` contains a unique name `mimData`.id which is subsequently used for linking `mimModel` and `mimBRModel` objects with a `mimData` object. A `mimData` object can subsequently be sent to **MIM**. For example:

```
data(rats)
md <- as.mimData(data=rats, mim.names=c("a","b","x","y"),
          mim.labels=names(rats))
submit.mimData(md)
```

(The `mim.read` function described previously is a wrapper for those two functions.)

6.2 Undirected models – `mimModel` objects

Models (or more precisely undirected models) can be turned into `mimModel` objects using the `mim.model` function. Here one can use the one–letter **MIM** names or the corresponding names in the dataframe for specifying the models. For example:

```
mm1 <- mim.model(mim.formula="ab/abx,aby/abxy", data=md)
mm2 <- mim.model("Sex, Drug/Sex:W1,Drug:W1,\
    Sex:W2,Drug:W2/Sex:W1:W2,Drug:W1:W2", data=md,
    mim.names=FALSE)
```

In the two calls of `mim.model` we could have set `submit.data=FALSE` whereby the data is not entered into **MIM** (data is already read into **MIM**). This saves some time. (Note that the `mim.model` function does not check whether the model is syntactically correct in the sense of the restrictions imposed on models in **MIM**.)

6.3 Block recursive models – `mimBRModel` objects

In a similar fashion, block recursive models can be specified using the `mim.br.model` function. First, however, a block recursive structure must be defined. The following two calls to `mim.setblock` are equivalent:

```
mim.setblock(c("ab","x","y"),data=md)
mim.setblock(c("Sex,Drug","W1","W2"),data=md,mim.names=FALSE)
```

Next two models are specified:

```
br1 <- mim.br.model(c("ab", "ab/abx/abx","ab/abx,aby/abxy"),
        data=md)
br2 <- mim.br.model(c("Sex:Drug",
    "Sex:Drug/Sex:Drug:W1/Sex:Drug:W1",
    "Sex:Drug/Sex:Drug:W1,Sex:Drug:W2/Sex:Drug:W1:W2"),
    mim.names=FALSE, data=md)
```

7 Using the object structure

Information about the models created above is also stored in the `mimData` object `md` in the slot `model.list`. Hence, it is possible to get back to a model as follows:

```
> get.models(md, short=TRUE)
Model  1
MIM formula: ab/abx,aby/abxy
Model  2
MIM formula: a,b/ax,bx,ay,by/axy,bxy
Model  3
Block: 1  MIM formula: ab
Block: 2  MIM formula: ab/abx/abx
Block: 3  MIM formula: ab/abx,aby/abxy
Model  4
..........
```

It is now possible to make model 2 the "current model" as follows:

```
curr.mod <- get.models(md, model=2)
mim.setblock() ## Removes the block structure
submit.model(curr.mod, submit.data=FALSE)
```

It is possible to save a `mimData` object and all models associated with it:

```
save.mimData(md,file="RatsModels")
```

7.1 Space considerations

The implementation of the object structure outlined above means that a given data set is stored only once – in the `mimData` object. All the model objects have access to the data through their reference to the `mimData` object. This is different from how most other functions in **R** work: With `lm1 <- lm(y x,data=mydata)` a copy of `mydata` is stored in the `lm1` object.

8 Miscellaneous

8.1 mimR mailing list

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

8.2 Availability

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

8.3 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.

8.4 Installation of mimR

Installation of **mimR** is done by the following steps.

- 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

- The executable `mimBatch.exe` (located in the `exec` folder in the **mimR** package) must be placed somewhere on your path. We suggest putting the executable `mimBatch.exe` into the folder where the **MIM** program is located. To add the location to path, you can do the following: 1) Right-click on My Computer, 2) Select properties, 3) Select advanced, 4) Add a new variable called Path in the upper frame, together with the required path, 5) Click OK twice.
- Before starting using **mimR** make sure that **MIM** is running (it can be minimized, though).

8.5 Upgrades of MIM

Upgrades of **MIM** are frequently released and can be downloaded from www.hypergraph.dk. 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 like the following appears in **R**. From this one sees that **mimR** expects **MIM** version 3.1.2.9 or later.

```
------------------------------------------------------------
mimR: An R interface to MIM for graphical modelling in R
mimR, version 0.0001 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 installed on your
  computer (Windows only)
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 The executable mimBatch.exe (which comes with the mimR package)
  must be placed somewhere on your path.
o Before starting using mimR, make sure that MIM is
  running.

  For a demo of mimR, type demo(mimR)
------------------------------------------------------------
```

8.6 How mimR works

The communication between **R** and **MIM** is based on the COM automation server under Windows. The executable program `mimBatch.exe` (located in the `exec` folder in the **mimR** package) is a batch interface to **MIM**. If `cmdfile.txt` is a file with **MIM** commands then

```
mimBatch.exe < cmdfile.txt
```

causes **MIM** to execute the **MIM** commands in `cmdfile.txt` and return the results in the file `cmdfile_out.txt`. Thus in **mimR** the user specifies the **MIM** commands, these are written to a file and sourced into `mimBatch.exe`. The results are read back into **R** from a file and the results are interpreted in a suitable way.

8.7 Warnings and limitations

The following warnings should be observed:

- The print format in **MIM**: In **MIM** the print format is given as `printformat a,b` (can be abbreviated `pf a,b`) where `a` denotes the space allocated for printing a number and `b` denotes the number of digits allowed. If `a` is too small then an output in **MIM** could be `-713.4445-411.2344`. This will obviously cause an error in **mimR**. The way to avoid such problems is to issue the command

  ```
  mim.cmd("printformat 12,4")
  ```

  which will allow 12 spaces for printing a number with 4 digits such that the result will be `-713.4445  -411.2344`.

- The output from **MIM** is written directly into a text file. There is an upper limit of about 6000 values (1.231 is one value) for the length of the output If the output is too large (this can happen if printing the data set with `mim.print("d")`) then the last part of the output is not read into **mimR**. To avoid such problems in connection with printing a large dataset use `mim.diary.data()` which will return a data frame.

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.