

Addendum to the paper
The **mimR** Package for Graphical Modelling in R

Søren Højsgaard
Danish Institute of Agricultural Sciences

November 5, 2006

Contents

1	Introduction and background	2
2	Preliminaries	2
2.1	Availability, information and installation	2
2.2	Limitations	3
2.3	Known problems	3
3	Specifying and displaying models	3
3.1	Discrete models	3
3.2	Continuous models	4
3.3	Mixed models	4
4	Models in mimR	5
4.1	Model formulae	6
4.2	Specification of special models	6
4.3	Model summary and model properties	6
4.4	Fitted values (parameter estimates)	7
5	Model editing	7
6	Model selection	8
7	Graphical meta data – gmData	9
7.1	Making a gmData object from a dataframe or a table	9
7.2	Creating a gmData object without data	10
7.3	Discrete data arranged as cumulated cell counts in dataframe	10
7.4	Creating gmData from sufficient statistics	11
7.4.1	Mixed data	11
7.4.2	Continuous data	12
7.4.3	Discrete data	13
8	Model fitting	13
8.1	Direct maximum likelihood estimation	13
8.2	EM algorithm	14
9	Latent variables	14
9.1	Fitting a model with a discrete latent variable	14
9.2	Controlling the EM algorithm	16
9.3	Fitting a model with a continuous latent variable	16

10 Discussion	17
11 Acknowledgements	18
A Miscellaneous	18
B Low level access to MIM from R	18
B.1 Primitive use of MIM from R – the <code>mim.cmd()</code> function	18
B.2 Using MIM directly from <code>mimR</code> – the <code>mcm()</code> function	18

1 Introduction to the addendum

The `mimR` package for graphical modelling in R was described by Højsgaard (2004). A major revision of the package has implied some changes in the functionality related to the description in Højsgaard (2004). Therefore, this addendum is the relevant document to use in connection with practical use of `mimR`.

The major changes relative to Højsgaard (2004) are:

- Models are fitted at the time of specification (unless one explicitly wants to avoid this).
- Models can be displayed graphically if the `Rgraphviz` package is installed.
- Facilities for reading data in various formats are available.

The addendum is organised differently from (Højsgaard 2004) but covers otherwise the same material.

1 Introduction and background

The `mimR` package is a package which provides facilities for graphical modelling in the statistical program R (R Development Core Team 2006). `mimR` is part of the `gR`-initiative (Lauritzen 2002) which aims to make graphical models available in R.

The statistical background for `mimR` is (M)ixed (I)nteraction (M)odels which is a general class of statistical models for mixed, discrete and continuous variables, where focus is on modelling conditional independence restrictions.

Statistical inference in mixed interaction models can be made with the program `MIM`, (Edwards 2000). The core of `mimR` is an interface from R to `MIM`.

This paper does not describe the statistical theory; instead the reader is referred to Edwards (2000). For a comprehensive account of graphical models we refer to Lauritzen (1996). Other important references are Edwards (1990) and Lauritzen and Wermuth (1989).

2 Preliminaries

2.1 Availability, information and installation

The `mimR` package uses the `MIM` program as inference engine. `MIM` is only available on Windows platforms and hence so is `mimR`. The `MIM` program itself (available from <http://www.hypergraph.dk>) must be installed on the computer. The

31 communication between R and MIM is based on the `rcom` package which is auto-
 32 matically installed when `mimR` is installed. The `mimR` package has a homepage,
 33 <http://gbi.agrsci.dk/~sorenh/mimR>.

34 In addition to the documentation in the `mimR` package, the MIM program itself
 35 contains a comprehensive help function which the user of `mimR` is encouraged to
 36 make use of. To access the help function in MIM either type `helpmim()` in R or
 37 switch to the MIM program window and press F1.

38 2.2 Limitations

39 The maximum number of variables in models in `mimR` is 52. This is because the
 40 internal representation of variables in MIM is as letters (MIM is case sensitive in this
 41 respect).

42 2.3 Known problems

43 MIM is automatically started by `mimR` if MIM is not already running. Sometimes (but
 44 not always) this causes a window to pop up with a text like "Access violation at
 45 address 00541FDD in module 'mim3206.exe'. Read of address 00EAE238."
 46 We do not know why this happens, but the problem can be avoided by simply start-
 47 ing up MIM manually before invoking `mimR`.

48 When a dataframe is sent to MIM this is done by writing a file in the `tmpdir`
 49 of the current R session. This file is afterwards read into MIM. (This turns out to
 50 be the fastest way of getting larger amounts of data from R to MIM). MIM can not
 51 read such files if the `tmpdir` contains a hyphen ("-"). For example, if the `tmpdir` is
 52 `c:/my-tmp-dir/` then `mimR` will not work.

53 3 Specifying and displaying models

54 In this section we show how to specify and display models in `mimR` for data arranged
 55 in a dataframe (where each row represent a case) or in a table as cumulated counts
 56 (for discrete variables). It is also possible to work with data arranged in other forms.
 57 Details are given in Section 7.

58 3.1 Discrete models

59 The discrete models are hierarchical log-linear models for contingency tables. For
 60 example, the contingency table `HairEyeColor` (which comes with R) contains a cross
 61 classification of persons with respect to gender, hair colour and eye colour:

```
> HairEyeColor
```

```
, , Sex = Male
      Eye
Hair   Brown Blue Hazel Green
Black   32   11    10     3
Brown   38   50    25    15
Red     10   10     7     7
Blond    3   30     5     8

, , Sex = Female
      Eye
Hair   Brown Blue Hazel Green
```

Black	36	9	5	2
Brown	81	34	29	14
Red	16	7	7	7
Blond	4	64	5	8

62 The model with generating class "Eye:Hair+Sex" satisfies that $(Eye, Hair)$ are
63 independent of Sex and is specified with:

```
> hec1 <- mim("Eye:Hair+Sex//", data = HairEyeColor)
> hec1
```

```
Formula: Eye:Hair+Sex//
-2logL: 3643.191 DF: 15
```

64 If the **Rgraphviz** package is installed, the model can be displayed graphically as
65 in Figure 1 by:

```
> display(hec1)
```

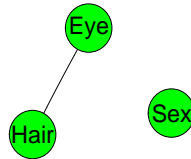


Figure 1: A graphical (log-linear) model for discrete data.

66 3.2 Continuous models

67 The following data set (taken from Mardia *et al.* (1979), see also Edwards (2000))
68 contains the examination marks for 88 students in 5 different subjects. Data is con-
69 tained the data set **math**. A stepwise backward model selection yields the “butterfly”
70 model shown in Figure 2 see also Whittaker (1990), p. 4.

71 This model can be specified as

```
> data(math)
> math2 <- mim("//me:ve:al+al:an:st", data = math)
```

72 3.3 Mixed models

73 Mixed models, or conditional Gaussian models (CG-models), arise by combining
74 log-linear models and graphical Gaussian models. The **rats** dataset is from a
75 hypothetical drug trial, where the weight losses of male and female rats under three
76 different drug treatments have been measured after one and two weeks. See Edwards
77 (2000) for more details. The first rows of the data are:

```
> data(rats)
> rats[1:5, ]
```

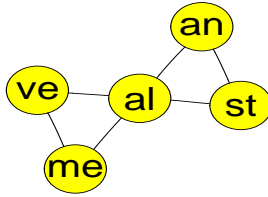


Figure 2: The selected graphical Gaussian “butterfly” model for the mathmarks data.

	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

78 For example, the model in Figure 3 is obtained with

```
> m1 <- mim("Sex:Drug/Sex:Drug:W2 + Drug:W1/W1:W2", data = rats)
> m1
```

```
Formula: Sex:Drug/Sex:Drug:W2 + Drug:W1/W1:W2
-2logL: 273.89 DF: 18
```

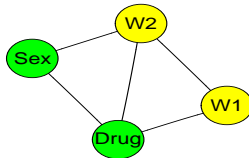


Figure 3: The model with generating class “Sex:Drug/Sex:Drug:W2 + Drug:W1/W1:W2”

79 4 Models in mimR

80 Only undirected models are available in `mimR`. That is, models in which all variables
 81 are treated on equal footing as response variables. Models where a possible response
 82 structure has to be accounted for can not be dealt with in `mimR`.

83 An undirected model is created using the `mim` function (which returns a `mim`
 84 object). Default is that the model is fitted to data, but fitting can be avoided
 85 by setting `fit=FALSE`. To explicitly fit a model, use the `fit()` function which is
 86 described in Section 8.

87 4.1 Model formulae

The general form of a model formula in `mimR` is

$$d_1 + d_2 + \cdots + d_r/l_1 + l_2 + \cdots + l_s/q_1 + q_2 + \cdots + q_t$$

88 where d_j , l_j and q_j are the respectively discrete, linear and quadratic generators.

89 A formula in `mimR` must be given as a string, i.e. in quotes ("`...`"). It is not
90 possible to specify models using the conventional R syntax, i.e. with `~....`. The
91 engine for specifying and fitting models is the `mim` function.

92 For example:

```
> gmdRats <- as.gmData(rats)
> mRats <- mim("Sex:Drug/Sex:Drug:W1+Sex:Drug:W2/W1:W2", data = gmdRats)
```

93 4.2 Specification of special models

94 It is possible to specify certain specific models (possibly for only a subset of the
95 variables) in short form. These are 1) the main effects model (as "`.`"), 2) the
96 saturated model (as "`..`") and 3) the homogeneous saturated model as (as "`..h`").

97 For example:

```
> mim(".", data = gmdRats, marginal = c("Sex", "Drug", "W1"))
> mim("..", data = gmdRats, marginal = c("Sex", "Drug", "W1"))
> mim("..h", data = gmdRats, marginal = c("Sex", "Drug", "W1"))
```

98 4.3 Model summary and model properties

99 A summary and a description of certain model properties of a `mim` model can be
100 achieved using the `summary()` and `properties()` functions:

```
> summary(mRats)
```

```
Formula: Sex:Drug/Sex:Drug:W1+Sex:Drug:W2/W1:W2
Variables in model : Sex Drug W1 W2
deviance: 27.807 DF: 15 likelihood: 273.705
```

101 Some properties of the model can be obtained with:

```
> properties(mRats)
```

```
Model properties:
Variables in model : Sex Drug W1 W2
Cliques: [1] "Sex:Drug:W1:W2"
Is graphical      : TRUE   Is decomposable: TRUE
Is mean linear    : TRUE   Is homogeneous : TRUE   Is delta-collapsible: TRUE
```

102 The model summary reads as follows: 1) The model is fitted to data. 2) The
103 model is graphical (such that there is a 1–1 correspondence between the model and
104 its interaction graph). 3) The model is decomposable meaning that the maximum
105 likelihood estimate exists in closed form (i.e. no iteration is needed). 4) The model is
106 mean linear meaning that the regressions of each continuous variable on the discrete

107 variables all have the same structural form. 5) The model is homogeneous meaning
 108 that the variance of the continuous variables does not vary with the levels of the
 109 discrete variables. 6) Finally, the model is Δ -collapsible which means that the
 110 model can be collapsed onto the discrete variables.

111 A more general function is `modelInfo()` which provides various model infor-
 112 mation as a list. The function can be given an additional argument to take out a
 113 specific slot in the list. For example, to take out the linear generators do:

```
> modelInfo(mRats, "mimGamma")
```

```
[1] "W1" "W2"
```

114 4.4 Fitted values (parameter estimates)

115 The fitted values (parameters estimates) can be obtained using the `fitted()` func-
 116 tion:

```
> fitted(mRats)
```

	Drug	Sex	Freq	W1	W2	W1:W1	W1:W2	W2:W1	W2:W2
1	1	1	4	7.50	8.25	3.938	3.187	3.187	4.75
2	2	1	4	7.75	8.75	3.938	3.187	3.187	4.75
3	3	1	4	13.50	8.50	3.938	3.187	3.187	4.75
4	1	2	4	6.50	6.25	3.938	3.187	3.187	4.75
5	2	2	4	7.25	8.25	3.938	3.187	3.187	4.75
6	3	2	4	16.00	12.00	3.938	3.187	3.187	4.75

117 The data frame contains for each configuration of the discrete variables 1) the
 118 number of cases with that configuration and 2) the estimated mean vector and
 119 covariance matrix.

120 5 Model editing

121 Models can be edited using the `editmim()` function by which one can 1) delete
 122 edges, 2) add edges, 3) homogeneously add edges, 4) delete terms (interactions)
 123 and 5) add terms. We refer to Edwards (2000) for the precise definitions of these
 124 terms. It should be noted that operations are conducted in the order specified
 125 above. For example:

```
> m1 <- mim(".", data = rats)
> m2 <- editmim(m1, addEdge = c("Sex:Drug", "Sex:W2"))
```

126 Some properties of this model are

```
> properties(m2)
```

```
Model properties:
Variables in model : Sex Drug W1 W2
Cliques: [1] "Sex:Drug" "Sex:W2" "W1"
Is graphical      : TRUE   Is decomposable: TRUE
Is mean linear    : TRUE   Is homogeneous : FALSE   Is delta-collapsible: TRUE
```

127 The model specified this way is heterogeneous because the variance of W2 depends
128 on Sex). To add homogeneous terms, the `haddEdge` keyword can be used as in:

```
> m3 <- editmim(m1, addEdge = "Sex:Drug", haddEdge = "Drug:W1:W2")  
> properties(m3)
```

```
Model properties:  
Variables in model : Sex Drug W2 W1  
Cliques: [1] "Sex:Drug" "Drug:W1:W2"  
Is graphical      : TRUE   Is decomposable: TRUE  
Is mean linear    : TRUE   Is homogeneous : TRUE   Is delta-collapsible: TRUE
```

129 Note the difference between deleting edges and terms:

```
> h1 <- mim(".", data = HairEyeColor)  
> editmim(h1, deleteEdge = "Hair:Eye:Sex")
```

```
Formula: Sex + Eye + Hair//  
-2logL: 3789.635 DF: 24
```

```
> editmim(h1, deleteTerm = "Hair:Eye:Sex")
```

```
Formula: Eye:Sex + Hair:Sex + Hair:Eye//  
-2logL: 3622.028 DF: 9
```

130 Note that if the starting model is (un)fitted, then so are all subsequent models
131 derived using the `editmim()` function. To explicitly fit a model, use the `fit()`
132 function, see Section 8.

133 6 Model selection

134 The `stepwise()` function performs stepwise model selection. This function takes
135 as additional arguments all arguments that the `STEPWISE` command in MIM does.
136 The `stepwise()` function returns a new `mim` object.

137 We consider the pig carcass data and start with the independence model:

```
> data(car carcass)  
> gmdCarc <- as.gmData(car carcass)  
> mainCarc <- mim(".", data = gmdCarc)
```

138 A forward stepwise selection using significance testing as selection criterion with
139 0.001 as critical level is obtained with:

```
> carcForw <- stepwise(mainCarc, arg = "f", critlevel = 0.001)
```

140 The resulting model

```
> carcForw
```

```
Formula: //F11:F12:F13 + F11:F12:LMP + F11:M12:M13 + F11:M13:LMP + M11:M12:M13  
-2logL: 11438.84 DF: 10
```

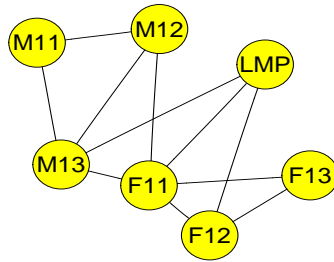



Figure 4: The covariance selection model obtained after a forward selection for the carcass data.

is shown in Figure 4.

Alternatively we can make a backward stepwise selection using BIC as selection criterion, make an unrestricted search (as opposed to searching among decomposable models, which is the default) and make a non-coherent search (which means that the same edge can be tested several times during the models search):

```
> satCarc <- mim("../", data = gmdCarc)
> carcBack <- stepwise(satCarc, arg = "snu", critlevel = 0.001)
```

The resulting model is:

```
> carcBack
```

```
Formula: //F11:F12:M12:M13 + F11:F12:F13:M13 + F11:F12:F13:LMP + M11:M12:M13 + M11:F13:M13 + M11:F13:LMP
-2logL: 11375.99 DF: 5
```

7 Graphical meta data – gmData

The internal representation of data in `mimR` is by `gmData` which is short for “graphical meta data”. A `gmData` object contains information about variables, their labels, their levels (for discrete variables) etc. A `gmData` object will typically also contain data, but need not do so. The idea behind separating the specification of the variables from data is that some properties of a model, for example decomposability and collapsibility, can be investigated without any reference to data.

Data represented as a dataframe or table (as in Section 3) are automatically converted to `gmData` in the `mim` function. Data in certain other can also be used in `mimR`. However, for such data, one needs to create a `gmData` object as described below. The generic function for creating `gmData` objects is the `as.gmData` function.

7.1 Making a gmData object from a dataframe or a table

To create a `gmData` object with from a dataframe do:

```
> gmdRats <- as.gmData(rats)
> gmdRats
```

```

  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

```

160 To each variable, there is associated a letter. This letter is used in connection
 161 with the internal representation of models and variables in MIM and the user should
 162 not be concerned with this. The procedure is the same for data arranged in a
 163 table. Observations in their original form can be extracted with the **observations**
 164 function. To extract the first 5 rows of data do:

```
> observations(gmdRats)[1:5, ]
```

```

  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

```

165 To see the labels of the discrete variables, do:

```
> vallabels(gmdRats)
```

```

$Sex
[1] "F" "M"

$Drug
[1] "D1" "D2" "D3"

```

166 7.2 Creating a gmData object without data

167 A **gmData** object (without data) can be created by the **gmData()** function:

```

> gmData(c("Sex", "Drug", "W1", "W2"), factor = c(2, 3, FALSE,
  FALSE), vallabels = list(Sex = c("M", "F"), Drug = c("D1",
  "D2", "D3")))

```

168 If no vallabels are given, default values are imposed.

169 With such a specification, one can afterwards specify models and have **mimR** to
 170 find important properties of these models, e.g. whether a given model is decompos-
 171 able.

172 7.3 Discrete data arranged as cumulated cell counts in dataframe

173 Sometimes discrete data are arranged as cumulated cell counts, for example

```

> library(MASS)
> housing[1:5, ]

```

	Sat	Infl	Type	Cont	Freq
1	Low	Low	Tower	Low	21
2	Medium	Low	Tower	Low	21
3	High	Low	Tower	Low	28
4	Low	Medium	Tower	Low	34
5	Medium	Medium	Tower	Low	22

174 Here `Freq` contains the counts. To use these data in `mimR`, first turn the dataframe
 175 into a table, and then turn the table into a `gmData` object, i.e.

```
> housingTab <- xtabs(Freq ~ Sat + Infl + Type + Cont, data = housing)
> as.gmData(housingTab)
```

176 7.4 Creating gmData from sufficient statistics

177 For mixed interaction models, 1) a list of cell counts for the discrete variables, 2)
 178 a mean vector for the continuous variables for each cell, and 3) and a covariance
 179 matrix for each cell are a set of sufficient statistics. Data represented in this form
 180 (as moment statistics) can be used in `mimR` as will be illustrated below.

181 7.4.1 Mixed data

182 For mixed data there are two options, both to be illustrated for the `rats` data.

183 **Option 1** Specify a list with as many elements as there are cells in the table.
 184 Each element of the list must consist of three items: 1) The covariance matrix,
 185 2) the mean vector, and 3) the number of observations in the cell (in that order).
 186 The covariances must be the estimate obtained by dividing the sum of products of
 187 residuals by the number of observations n per group, not $n - 1$.

188 For the `rats` data we can extract first splitting data by the levels of the discrete
 189 variables using the `doBy` package, (Højsgaard 2006):

```
> r <- splitBy(~Sex + Drug, data = rats)
```

190 The necessary list can be obtained by:

```
> cmc <- lapply(r, function(x) cov.wt(x[, c("W1", "W2")], method = "ML"))
```

```
> x <- momentstats(factor = c("Sex", "Drug"), level = c(2, 3),
  continuous = c("W1", "W2"), cmc = cmc)
> as.gmData(x)
```

	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:		momentstats		

191 **Option 2** Specify 1) a list of covariances matrices, 2) a list of mean vectors, and
 192 3) a list of cell counts:

```
> covmats <- lapply(r, function(x) cov.wt(x[, c("W1", "W2")], method = "ML")$cov)
> meanvecs <- lapply(r, function(x) mean(x[, c("W1", "W2")]))
> counts <- lapply(r, function(x) nrow(x))
> x <- momentstats(factor = c("Sex", "Drug"), level = c(2, 3),
  continuous = c("W1", "W2"), covariances = covmats, means = meanvecs,
  counts = counts)
> as.gmData(x)
```

```
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: momentstats
```

193 It is wise to check that data have been entered correctly by:

```
> toMIM(x)
> mim.cmd("print s")
```

194 7.4.2 Continuous data

195 For continuous data the same two options as for mixed data are available. For
 196 example for the **math** data we can do:

```
> cmc <- cov.wt(math, method = "ML")
> x <- momentstats(continuous = names(math), cmc = cmc)
> as.gmData(x)
```

```
name letter factor levels
1 me      a FALSE     NA
2 ve      b FALSE     NA
3 al      c FALSE     NA
4 an      d FALSE     NA
5 st      e FALSE     NA
Data origin: momentstats
```

197 OR:

```
> x <- momentstats(continuous = names(math), counts = nrow(math),
  means = mean(math), covariances = cov.wt(math, method = "ML")$cov)
> as.gmData(x)
```

```
name letter factor levels
1 me      a FALSE     NA
2 ve      b FALSE     NA
3 al      c FALSE     NA
4 an      d FALSE     NA
5 st      e FALSE     NA
Data origin: momentstats
```

198 7.4.3 Discrete data

199 Schoener (1968) describes data concerning the perching behaviour of two species of
 200 lizards, see also Edwards (2000). Data is a three-way contingency. Data, repre-
 201 sented as a list of counts, can be turned into a `gmData` object with:

```
> x <- momentstats(factor = c("species", "diameter", "height"),
  level = c(2, 2, 2), counts = c(32, 86, 11, 35, 61, 73, 41,
    71), vallabels = list(species = c("anoli", "disticus"),
    diameter = c("<=4", ">4"), height = c(">4.75", "<=4.75")))
> z <- as.gmData(x)
> vallabels(z)
```

```
$species
[1] "anoli" "disticus"

$diameter
[1] "<=4" ">4"

$height
[1] ">4.75" "<=4.75"
```

202 The order of the cells are $(1, 1, 1), (1, 1, 2), (1, 2, 1), (1, 2, 2), \dots, (2, 2, 1), (2, 2, 2)$,
 203 i.e. the last index varies fastest.

204 8 Model fitting

205 8.1 Direct maximum likelihood estimation

206 The function for fitting models via direct maximum likelihood estimation is `fit`:

```
> m1 <- mim("...", data = rats, marginal = c("Sex", "Drug", "W1"),
  fit = FALSE)
> fit(m1)
```

```
Formula: Sex:Drug/Sex:Drug:W1/Sex:Drug:W1
-2logL: 178.873 DF: 0
```

207 8.2 EM algorithm

208 For data given as a dataframe, the EM algorithm (Dempster *et al.* 1977) is available
 209 to handle incomplete observations. For example

```
> r2 <- rats
> r2[1:2, 3] <- r2[3:4, 4] <- NA
> r2[1:5, ]
```

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

210 The EM algorithm is switched on by `fit="e"`:

```
> mim("..", data = r2, fit = "e")
```

```
Formula: Sex:Drug/Sex:Drug:W1 + Sex:Drug:W2/Sex:Drug:W1:W2
-2logL: 170.421 DF: 0
```

211 If the argument `fit="e"` is not given, then `fit` will try to use the EM algorithm
 212 if direct maximum likelihood estimation fails:

```
> m2 <- mim("..", data = r2)
```

```
Seems that there are incomplete observations - trying EMfit
```

213 The EM algorithm starts by substituting random starting values for missing data.

214 9 Latent variables

215 9.1 Fitting a model with a discrete latent variable

216 First we consider a latent variable model: We suppose that there is a latent binary
 217 variable **A** such that the manifest variables are all conditionally independent given
 218 **A**.

219 First we add a binary factor **A** (with missing values) to the `math` dataset:

```
> data(math)
> math$A <- factor(NA, levels = 1:2)
> gmdMath <- as.gmData(math)
```

220 Next, we make explicit in the `gmData` object that **A** is indeed a latent variable
 221 using the `latent()` function (in Section 9.2 it is explained why it must be specified
 222 explicitly that **A** is a latent variable):

```
> latent(gmdMath) <- "A"
> gmdMath
```

```
name letter factor levels
1 me a FALSE NA
2 ve b FALSE NA
3 al c FALSE NA
4 an d FALSE NA
5 st e FALSE NA
6 A f TRUE 2
Data origin: data.frame
Latent variables: A
```

223 The model can be specified as

```
> m1 <- mim("A/st:A+an:A+al:A+ve:A+me:A/st:A+an:A+al:A+ve:A+me:A",
  data = gmdMath)
```

```
Model has latent variable - trying EM algorithm
```

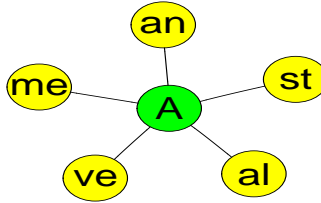


Figure 5: Latent variable model for `math` data.

224 The model is shown in Figure 5.

225 Predicted values for the latent variable under the model can be imputed in MIM
226 using

```
> imputeMissing()
```

227 To get the data (including the imputed values) from MIM to R do:

```
> d.imp <- retrieveData()
> d.imp[1:5, ]
```

```
me ve al an st A
1 77 82 67 67 81 2
2 63 78 80 70 81 2
3 75 73 71 66 81 2
4 55 72 63 70 68 2
5 63 63 65 70 63 2
```

228 and so we see that the first 5 cases are assigns A to have level 1.

229 Next, we plot the predicted value of A against the observation number:

```
> plot(as.numeric(d.imp$A))
```

230 The plot is shown in Figure 6. The grouping of the values of A suggests that
231 data have been processed somehow prior to presentation. (Edwards 2000), p. 181,
232 conclude: “Certainly they (the data) have been mistreated in some way, doubtless
233 by a statistician.”

234 9.2 Controlling the EM algorithm

235 The EM algorithm needs a set of initial values for the unobserved values to start
236 from when calculating the parameter estimates in the first iteration. The final
237 estimate of the EM algorithm may depend on the initial values and that (especially
238 in the case of latent variables) the likelihood may have multiple maxima. Default
239 is that random starting values are imputed and that was actually the case above,
240 where the factor A was given NA values.

241 An alternative is to specify starting values for the latent variables in the dataframe,
242 e.g. as

```
> data(math)
> math$A <- factor(1:2, levels = 1:2)
> latent(gmdMath) <- "A"
```

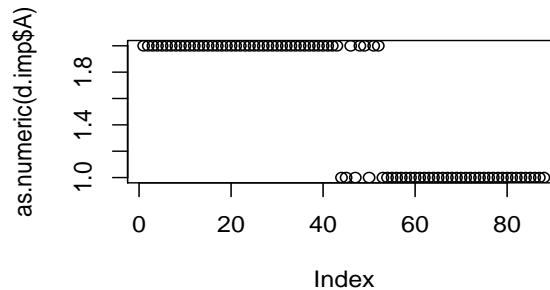


Figure 6: An index plot of the discrete latent variable A.

```
> m1 <- mim("A/st:A+an:A+al:A+ve:A+me:A/st:A+an:A+al:A+ve:A+me:A",
  data = gmdMath, fit = "es")
> m1
```

```
Formula: A/st:A+an:A+al:A+ve:A+me:A/st:A+an:A+al:A+ve:A+me:A
Latent variables in model: A
-2logL: 3454.935 DF: 20
```

243 The specification `fit='es'` means that the model should be fitted with the EM
 244 algorithm and that the given values of the latent variables should be used as starting
 245 values for the EM algorithm. Setting `fit='er'` means that random starting values
 246 will be used for the EM algorithm.

247 For this reason latent variables must be declared explicitly in a `gmData` object.
 248 By this approach the sensitivity of the EM algorithm on starting values can be
 249 investigated.

250 9.3 Fitting a model with a continuous latent variable

251 To illustrate controlling of the EM algorithm, we make an alternative analysis,
 252 where A is regarded as a continuous variable. To speed up the convergence of the
 253 EM algorithm, we do a factor analysis to get good starting values:

```
> data(math)
> fa <- factanal(math, factors = 1, scores = "regression")
> math$A <- fa$scores
```

254 Then we create a `gmData` object with this new augmented data set and declares
 255 that A is to be regarded as a latent variable:

```
> gmdMath <- as.gmData(math)
> latent(gmdMath) <- "A"
> m1 <- mim("//st:A+an:A+al:A+ve:A+me:A", data = gmdMath)
```

```
Model has latent variable - trying EM algorithm
```


256 As before we impute the missing values, retrieve the data to R and plot the
257 imputed values for the latent variable:

```
> imputeMissing()  
> d.imp <- retrieveData()  
> plot(d.imp$A)
```

258 The plot of the imputed values for the latent variables are shown in Figure 7
259 and this also suggests that the data do not emerge in random order.

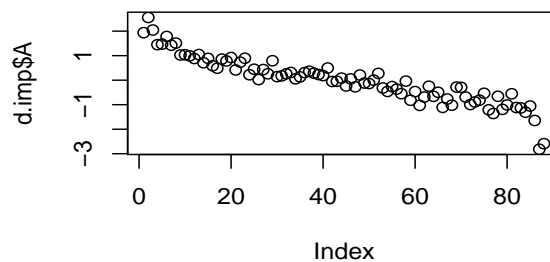


Figure 7: An index plot of the continuous latent variable A.

260 10 Discussion

261 In this manual we have illustrated some aspects of the `mimR` package for graphical
262 modelling in R. It is the hope that `mimR` will be obsolete in a not too distant future
263 – not because of lack of relevance of being able to work with graphical models in R.
264 Rather, it is the hope that a more proper package with with at least the functionality
265 of `mimR` will be created. That is one of the aims of the `gR`-project, which has lead to
266 the minimal package `gRbase`, (Dethlefsen and Højsgaard 2005), which is available
267 on CRAN. The functionality of `gRbase` is however very limited and as such `mimR` is
268 a relevant package to use for graphical modelling in R.

269 11 Acknowledgements

270 David Edwards (the creator of MIM) is greatly acknowledged for his support in the
271 creation of `mimR`. Claus Dethlefsen has made valuable comments to this addendum.
272 The members of the `gR` initiative are acknowledged for their inspiration.

273 A Miscellaneous

274 B Low level access to MIM from R

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

276 The core of `mimR` is the `mim.cmd` function. The arguments to `mim.cmd` are simply
277 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")
```

278 The `mim.cmd` function returns the result of the commands submitted to MIM.
 279 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
```

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

281 The `mcm` function (short for “MIM command mode”) provides a direct interface to
 282 MIM, i.e. the possibility to write MIM commands directly. The `mcm` function returns no
 283 value to R, and is intended only as an easy way to submit MIM commands without the
 284 overhead of wrapping them into the `mim.cmd` function (or submitting the commands
 285 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
>
```

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

289 References

- 290 Dempster, A. P., Laird, N., and Rubin, D. B. (1977). Maximum likelihood from
 291 incomplete data via the EM algorithm (with discussion). *Journal of the Royal*
 292 *Statistical Society, Series B*, **39**, 1–38.
- 293 Dethlefsen, C. and Højsgaard, S. (2005). A common platform for graphical models
 294 in r: The grbase package. *Journal of Statistical Software*, **14**, 1–12.

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

297 Edwards, D. (2000). *Introduction to graphical modelling*, (2nd edition edn). Springer
298 Verlag, New York.

299 Højsgaard, S. (2004). The mimR package for graphical modelling in R. *Journal of*
300 *Statistical Software*, **11**, (6).

301 Højsgaard, S. (2006). *doBy: Groupwise computations*. R package version 0.7.

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

303 Lauritzen, S. L. (2002). gRaphical models in R: A new initiative within the R
304 project. *Rnews*, **2**, 39.

305 Lauritzen, S. L. and Wermuth, N. (1989). Graphical models for associations be-
306 tween variables, some of which are qualitative and some quantitative. *Annals*
307 *of Statistics*, **17**, 31–57.

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

310 R Development Core Team (2006). *R: A language and environment for statistical*
311 *computing*. R Foundation for Statistical Computing, Vienna, Austria. ISBN
312 3-900051-00-3.

313 Schoener, T. W. (1968). The anolis lizards of bimini: Resource partitioning in a
314 complex fauna. *Ecology*, **49**, 704–26.

315 Whittaker, J. (1990). *Graphical models in applied multivariate statistics*. Wiley.