mimR – a (small) contribution to gR

Søren Højsgaard
Biometry Research Unit
Danish Institute of Agricultural Sciences

March 26, 2003 DSC conference, Vienna, 2003
mimR – graphical models in R

- **MIM** ([www.hypergraph.dk](http://www.hypergraph.dk)) is a general (and unique) package for statistical inference (model fitting, model selection etc.) in *Mixed Interaction Models* – a large class of “graphical” models (where conditional independence restrictions are of interest).

- **R** is a general statistical package and programming environment.

- **mimR** integrates the functionality of **MIM** into **R**. **mimR** is available as an **R** package. For additional information, see

  http://www.jbs.agrsci.dk/~sorenh/mimR/
Outline of presentation

- Why mimR??
  1. In 2000, needed to access MIM from Splus
  2. In 2002, the gR–workshop lead to a proper R package

- Why the name mimR??

- Technicalities: Accessing MIM from R

- Primitive use of mimR

- The object structure of mimR:
  - mimData
  - mimModel

- The future of mimR

- Example

March 26, 2003 DSC conference, Vienna, 2003
Technicalities: Accessing \texttt{MIM} from \texttt{R}

Think of \texttt{MIM} as the “inference engine” accessed from \texttt{R}:

\begin{align*}
\text{mimR 1.0} & \quad \text{R} \quad \text{MIMTerm} \quad \text{MIM} \\
\text{mimR 1.01} & \quad \text{R} \quad \text{sockets} \quad \text{MIM} \\
\text{mimR 1.0?} & \quad \text{R} \quad ??? \quad "\text{MIM}"
\end{align*}

Presumably not the most elegant way one can think of – but it works...

March 26, 2003 DSC conference, Vienna, 2003
MIM Models – very briefly

Discrete variables $d$, continuous variables $y$.

Model: $p(d, y) = p(d)p(y|d)$ where

- $p(d; \theta)$: Log–linear model
- $p(y|d) = N(\mu_d, \Sigma_d)$

Structure of models arise by imposing constraints on $(\theta, \mu_d, \Sigma_d)$.

The syntax for a MIM model

$$d_1, d_2, \ldots, d_p/l_1, l_2, \ldots, d_q/q_1, q_2, \ldots, q_r$$

refering to the discrete, linear and quadratic generators.
“Primitive” use of mimR

1. Direct access to MIM via `mim.cmd`–function, e.g.

   ```
   mim.cmd("input d:\mathmark.dat; model //xyz,xzu; fit;")
   ```

   which returns whatever the last MIM command returns.

2. mimR has a collection of functions mimicing corresponding MIM functions but interprets the output in appropriate ways (creates data structures with the output):

<table>
<thead>
<tr>
<th>mimR function</th>
<th>MIM command</th>
</tr>
</thead>
<tbody>
<tr>
<td>mim.stepwise</td>
<td>stepwise</td>
</tr>
<tr>
<td>mim.testdelete</td>
<td>testdelete</td>
</tr>
<tr>
<td>mim.print</td>
<td>print</td>
</tr>
<tr>
<td>mim.display</td>
<td>display</td>
</tr>
<tr>
<td>mim.fit</td>
<td>fit</td>
</tr>
<tr>
<td>mim.emfit</td>
<td>emfit</td>
</tr>
</tbody>
</table>
mimR is object–oriented

- Variable names in MIM are 1–character, “a”, “b”, ...., which is an inconvenient restriction...

- To circumvene this restriction, a mechanism for mapping dataframe–names to MIM–names (and vice versa) was established by means of mimData–objects.

- From thereof the road was paved for creating mimModel–objects etc. such that (at least in principle) the user needs not be concerned with that model fitting, model selection etc. takes place in MIM.
**mimData-objects**

A data frame in R is turned into a `mimData`-object, which can be submitted to MIM:

```r
md <- as.mimData(data=...) submit(md)
```

`mimR` keeps track of that `md` is the *current MIM* data set (the data set currently loaded into MIM). (There can be at most one current data set at the time).

`md` is an object of class `mimData` to which certain *methods* are associated.

To a `mimData` object several `mimModel` objects can be associated.
mimModel–objects

A mimModel–object consists of

- A mimFormula–object (where the model formula follows MIM syntax).
- A (reference to) a corresponding mimData object.
- Summary statistics (currently, deviance and DF).
- Slot is.current (there can be at most one current model at the time)

```
mm <- mim.model( mim-formula, data=...)  
```
The future (if any) of mimR

*mimR* will hopefully be obsolete in a not too distant future – not because graphical modelling in *R* is not relevant, but because a more coherent approach to graphical modelling in *R* is needed.

– That is the aim of the gR–project.

Until that happens, proceed at slow pace:

- Move functionality from *MIM* to *R*
- Create graph modules
- ....

March 26, 2003 DSC conference, Vienna, 2003
Example

```r
library(mimR)
data(carcass)
md <- as.mimData(carcass)
submit.mimData(md)

mm2 <- stepwise(mm1,"f")
mm3 <- stepwise(mm2,"ub")

mp <- mim.print("i")
d1 <- display(mm2, "LMP", c("F11:M11:F12:M12:F13:M13"))
d2 <- display(mm2, c("LMP","F11"), c("M11:F12:M12:F13:M13"))
d3 <- display(mm2, c("LMP","F11"))

carcass2 <- carcass
carcass2[,"L"] <- rnorm(nrow(carcass2))
md2 <- as.mimData(carcass2)
submit.mimData(md2)
```

March 26, 2003 DSC conference, Vienna, 2003
mm4 <- mim.model("//L:F11,L:M11,L:F12,L:M12,L:F13,L:M13,L:LMP",
  data=md2, fit=F)
namesTable(md2)
mim.cmd("calc h=ln(0)")
mim.emfit(plot=T)
d4 <- display(mm4, "LMP", c("F11:M11:F12:M12:F13:M13"))

data(rats)
m.rats <- as.mimData(rats)
mm1 <- mim.model("Sex, Drug/Sex:W1,Drug:W1,
mm2 <- stepwise(mm1)