An introduction to the STAT-JR package
What will we cover?

• Some background to MLwiN and why at CMM we are developing new software
• Ideas behind STAT-JR
• Interoperability features
ML2, ML3 and MLN

- Jon Rasbash worked for Mike Healy at LSHTM and then transferred to Harvey Goldstein at Institute of Education.
- Harvey wrote his IGLS paper in 1986
- ML2 came out in 1988 (written in Fortran) built on top of Mike Healy’s NanoSTAT package
- ML3 followed in 1990 (converted to C code)
- MLN followed in 1995 (with new N level algorithms written in C++)
- Algorithms very fast for hierarchical models – good matrix routines for block diagonal structures.
- Bob Prosser completed the team.
Moving towards MLwiN - Jon

- MLwiN was another step change, this time to a Windows based program.
- ML2/3/N worked in a sequential way with each command performing an action and then the next etc.... Graphics were limited.
- MLwiN in contrast consists of a GUI front-end (written in Visual Basic) and all the objects are dynamic i.e. changing one window should change the contents of other windows.
- Jon worked with Bruce Cameron setting up this architecture and Bruce is still involved in STAT-JR.
Moving towards MLwiN – Bill in Bath

• I did my MSc. (Comp Stats) in 1995 and used S-Plus and C and BUGS to fit models for my dissertation.
• I then did my PhD. (Stats) between 1995 and 1998 supervised by David Draper. My PhD. included much comparison work between methods of fitting multilevel models (Bayesian & frequentist)
• I did this using both BUGS and MLwiN
• An artefact of the process was writing (limited) MCMC functionality into MLwiN!
MLwiN in 1998

What was good / What was new?

• Interactive Equations window
• Interactive Graphics
• Interactive Trajectories window
• Choice of estimation methods
• Adaptive Metropolis algorithms
Interactive Equations Window

\[
\text{normexam}_{ij} \sim N(XB, \Omega)
\]
\[
\text{normexam}_{ij} = \beta_{0ij}\text{cons} + 0.563(0.012)\text{standlrt}_{ij}
\]
\[
\beta_{0ij} = 0.002(0.040) + \mu_{0j} + \epsilon_{0ij}
\]

\[
\begin{bmatrix}
\mu_{0j} \\
\epsilon_{0ij}
\end{bmatrix} \sim N(0, \Omega_u) : \Omega_u = \begin{bmatrix}
0.092(0.018)
\end{bmatrix}
\]

\[
\begin{bmatrix}
\epsilon_{0ij}
\end{bmatrix} \sim N(0, \Omega_e) : \Omega_e = \begin{bmatrix}
0.566(0.013)
\end{bmatrix}
\]

\[-2*\text{loglikelihood(IQLS Deviance)} = 9357.242\text{(of 4059 cases in use)}\]

This is the 2011 version but the 1998 version had many of the same features. Could toggle between estimates and symbols. Numbers changing from blue to green on convergence. Clicking on the window would allow model construction – see X variable window.
Interactive Graphics

Graphics can instantly update as column content changes.

Highlighting of data points passed between graphs.

Highlighting can be done at different levels of the data structure.
Interactive Trajectories Window

Trajectories plot particularly useful for MCMC estimation

First software (to my knowledge) to have dynamic chains although they appeared in WinBUGS soon after.

Used to joke about taking a coffee break while MCMC ran and chains make you look busy!

Click on graph to get diagnostics
Sixway diagnostic plot

Plot of chain plus kernel density – code from my MSc!

Kernel would show informative priors as well.

Time series plots and originally MCSE and summary in bottom 2 boxes.

Expanded to 7 boxes later.
MCMC functionality

• In 1998 we had implemented Normal, Binomial and Poisson N level models by shoe horning my stand-alone C code into MLwiN in a couple of manic visits to London.
• For Normal responses used Gibbs sampling, for others used a mixture of Gibbs and random walk Metropolis including my ad-hoc adapting scheme that persists today.
• BUGS used AR sampling instead of MH at the time.
• Code was very model specific and optimised so far faster than BUGS as it still is today!
Changes to MLwiN in my time at IOE (1998-2003)

- Better handling of missing data, categorical variables.
- Lots more data manipulation windows.
- MCMC functionality for:
  - XC/MM/Spatial CAR models (Browne et al. 2001a)
  - Multilevel Factor Analysis (Goldstein & Browne 2002,2005)
  - Measurement Error (Browne et al 2001b)
  - Multivariate/Mixed responses/Complex variation (Browne 2006)
Part of the success of MLwiN is due to the quality and quantity of the accompanying documentation:

• Users Guide in 1998 (Goldstein et al. ~130 pages)

By 2003

• Users Guide (Rasbash et al. ~ 250 pages) & MCMC Guide (Browne et al.) & Command guide


Huge effort by colleagues in centre on web-based training materials (over 6,000 registered users)
MLwiN (2003-)

We still develop MLwiN even after Jon Rasbash’s death:

- The Users guide supplement shows Jon & Chris’s work on improved model specification, model comparison, customised predictions and auto-correlated error modelling.
- The last 5 chapters of the MCMC guide shows my work on MCMC algorithm improvements (SMVN, SMCMC, hierarchical centring, parameter expansion and orthogonal parameterisations) which we covered in the last lecture. See also Browne et al. (2009)
Jon Rasbash’s big vision

• Jon had been thinking hard on where the software went next.
• The frequentist IGLS algorithm was hard to extend further.
• WinBUGS showed that MCMC as an algorithm could be extended easily but the difficulty in MLwiN was in extending my MCMC code and possibly relying on the personnel!
• The big vision was an all-singing all-dancing system where expert users could add functionality easily and which interoperates with other software. Bruce was developing an underpinning algebra system.
• The ESRC LEMMA II and E-STAT grants would enable this to be achieved
Jon identified 3 groups of users:

- Novice practitioners who want to use statistical software that is user friendly and maybe tailored to their discipline
- Advanced practitioners who are the experts in their fields and also want to develop tools for the novice practitioners
- Algorithm Developers who want their algorithms used by practitioners.

STAT-JR component based approach

Below is an early diagram of how we envisioned the system. Here you will see boxes representing components some of which are built into the STAT-JR system. The system is written in Python with currently a VB.net algebra processing system. A team of coders (currently me, Chris, Danius, Camille Zhengzheng, Huanjia, Richard and Bruce) work together on the system.
Templates

Consist of a set of code sections for advanced users to write. For a model template it consists of at least:
• an invars method which specifies inputs and types
• An outbug method that creates (BUGS like) model code for the algebra system
• An (optional) outlatex method can be used for outputting LaTeX code for the model.
Other optional functions required for more complex templates
Regression 1 Example

```python
from EStat.Templating import *
from mako.template import Template as MakoTemplate
import re

class Regression1(Template):
    'A model template for fitting 1 level Normal multiple regression model in E-STAT only. To be used in documentation.'

tags = ['Model', '1-Level', 'e-STAT', 'Normal']
engines = ['eSTAT']

invars = ''
y = DataVector('response: ')
tau = ParamScalar()
sigma = ParamScalar()
sigma2 = ParamScalar()
x = DataMatrix('explanatory variables: ')
beta = ParamVector(parents=[x], as_scalar=True)
'''

outbug = '''
model{
    for (i in 1:length(${y})) {
        ${y}[i] ~ dnorm(mu[i], tau)
        mu[i] <- ${mmult(x, 'beta', 'i')}
    }
}

    # Priors
    % for i in range(0, x.ncols()):
    beta${i} ~ dflat()
    % endfor
    tau ~ dgamma(0.001000, 0.001000)
    sigma2 <- 1 / tau
    sigma <- 1 / sqrt(tau)
}
'''

outlatex = r'''
\begin{aligned}
  \mbox{${y}}_i & \sim \mbox{N}(\mu_i, \sigma^2) \\
  \mu_i & = \{\text{mmulttex}(x, r'\beta', 'i')\} \\
  \beta_0 & \propto 1 \\
  \tau & \sim \Gamma(0.001,0.001) \\
  \sigma^2 & = 1 / \tau
\end{aligned}
'''
```
An example of STAT-JR – setting up a model
An example of STAT-JR – setting up a model
Equations for model and model code

\[
\text{model}() = \begin{align*}
\text{for } i & \in \text{length}(\text{normex}) \{ \\
& \text{normex}[i] = \text{dnorm}(\mu[i], \tau[i]) \\
& \mu[i] \leftarrow \text{const}_3 + \text{beta0} + \text{standi}[1] \times \text{beta1} \\
\} \\
\text{# Priors} \\
\text{beta0 } & \sim \text{dflat}() \\
\text{beta1 } & \sim \text{dflat}() \\
\tau[i] & \sim \text{dgamma}(0.001000, 0.001000) \\
\text{sigma} & \leftarrow 1 / \tau[i] \\
\text{sigma} & \leftarrow 3 / \text{sqrt}(\text{tau})
\end{align*}
\]

**normex** \(\sim N(\mu_i, \sigma^2)\)

\[
\begin{align*}
\mu_i &= \beta_0 + \beta_1 \text{standi}_i \\
\beta_0 & \times 1 \\
\beta_1 & \times 1 \\
\tau & \sim \Gamma(0.001, 0.001) \\
\sigma^2 & = 1/\tau
\end{align*}
\]
Equations for model and model code

\[
\text{normexam}_i \sim N(\mu_i, \sigma^2)
\]
\[
\mu_i = \beta_0 \text{cons}_i + \beta_1 \text{standlrt}_i
\]
\[
\beta_0 \propto 1
\]
\[
\beta_1 \propto 1
\]
\[
\tau \sim \Gamma(0.001, 0.001)
\]
\[
\sigma^2 = 1/\tau
\]

- Note: Equations use MATHJAX and so underlying LaTeX can be copied and pasted. The model code is based around the WinBUGS language with some variation.
Model code in detail

model{
  for (i in 1:length(normexam)) {
    normexam[i] ~ dnorm(mu[i], tau)
    mu[i] <- cons[i] * beta0 + standlrt[i] * beta1
  }
}

# Priors
  beta0 ~ dflat()
  beta1 ~ dflat()
  tau ~ dgamma(0.001000, 0.001000)
  sigma2 <- 1 / tau
}

For this template the code is, aside from the length function, standard WinBUGS model code.
Bruce’s Algebra system steps

Deviance Function

\[
\text{deviance} = 2 \times \left( \tau \times \frac{\sum_{i=1}^{n} \text{length}(\text{normexam}_{i})}{\text{standlrt}_{i}} \right)^{1/2} + 0.5 \times (\ln(\sigma) - \ln(\sigma)) \times \text{length}(\text{normexam}) + 0.34667890279973 \times \text{length}(\text{normexam})
\]

Conditional posterior for \( \beta_1 \) for Gibbs sampling

\[
\beta_1 \sim N\left( \frac{\tau \times \sum_{i=1}^{n} \text{length}(\text{normexam}_{i}) \text{standlrt}_{i} \times (\text{normexam}_{i} - \beta_0 \times \text{cons}_{i})}{\tau \times \left( \sum_{i=1}^{n} \text{length}(\text{normexam}_{i}) \text{standlrt}_{i}^{2} \right)}, \tau \times \left( \sum_{i=1}^{n} \text{cons}_{i}^{2} \right) \right)
\]

Conditional posterior for \( \beta_0 \) for Gibbs sampling

\[
\beta_0 \sim N\left( \frac{\tau \times \sum_{i=1}^{n} \text{length}(\text{normexam}_{i}) \text{cons}_{i} \times (\text{normexam}_{i} - \beta_1 \times \text{standlrt}_{i})}{\tau \times \left( \sum_{i=1}^{n} \text{length}(\text{normexam}_{i}) \text{cons}_{i}^{2} \right)}, \tau \times \left( \sum_{i=1}^{n} \text{standlrt}_{i}^{2} \right) \right)
\]

Deterministic formula for parameter \( \sigma \)

\[
\sigma = \frac{1}{\sqrt{\tau(\tau)}}
\]

Deterministic formula for parameter \( \sigma_2 \)

\[
\sigma_2 = \frac{1}{\tau}
\]
Bruce’s Algebra system steps

Use Gibbs sampling from conditional posterior for beta1:

\[
\beta_1 \sim N\left( \frac{\tau \times \left( \sum_{i=1}^{\text{length(normexam)}} \text{standlrt}_i \times (\text{normexam}_i - \beta_0 \times \text{cons}_i) \right)}{\tau \times \left( \sum_{i=1}^{\text{length(normexam)}} \text{standlrt}_i^2 \right)}, \tau \times \left( \sum_{i=1}^{\text{length(normexam)}} \text{standlrt}_i^2 \right) \right)
\]

\[
\beta_1 \sim N(0.000249800292718 \times (2382.11475784 + \beta_0 \times (-7.346046)), 4003.19787107 \times \tau)
\]

Here the first line is what is returned by Bruce’s algebra system – which works solely on the model code. The second line is what can be calculated when values are added for constants etc.
The package can output C++ code that can then be taken away by software developers and modified.
Output of generated C++ code (new screen)

// Update beta1
{
    beta1 = dnorm((0.000249799765395*(2382.12631198+(beta0*(-7.34783096611))))),(4003.20632175*tau));
}

// Update beta0
{
    beta0 = dnorm(((((-0.462375992909)+((-7.34783096611)*beta1))*0.000246366100025),(tau*4059.0));
}

• Note now that the code includes the actual data in place of constants and so looks less like the familiar algebraic expressions
Output from the E-STAT engine

Estimates and the fit diagnostics can be viewed for the model fitted.
Output from the E-STAT engine

E-STAT offers multiple chains so that we can use multiple chain diagnostics to aid convergence checking.

Otherwise the graphs are borrowed from the MLwiN 6-way plotting.
INTEROPERABILITY
Interoperability with Stata

Stata can be chosen as an estimation engine.

Here Stat-JR acts as a front end interface and constructs a Stata do file of commands to be run.

Then on running Stata is called in the background using the do file and data file that is also constructed by Stat-JR.
Interoperability with Stata

Stata produces a log file of all the commands typed and their textual output and this can be viewed within Stat-JR.
Interoperability with Stata

Estimates and fit diagnostics can then be extracted from the log file and displayed in a ModelResults object.
Interoperability with Stata

Graphical plots constructed by Stata can also be passed back to Stat-JR and viewed.
Interoperability with R

R can be chosen as another alternative. In fact here we have 2 choices – MASS or MCMCglmm.

You will see in the left pane the data file ready for input to R.
Interoperability with R

If written in to the code in the template – graphics from other software can be extracted.

Here for example is a residual plot associated with the R fit of the model.
Interoperability with MLwiN

MLwiN can also be chosen as an alternative estimation engine. Here a macro file to be run in MLwiN is constructed and the output from MLwiN is constructed into a ModelResults object. Currently we are unable to get windows back from MLwiN into Stat-JR.
There are also templates for plotting.

For example here is a plot using the Xyplot template.

The left pane gives the Python script and the right pane the plot.
Different forms of STAT-JR and E-books

• Webtest - the format we have demonstrated up to now. Allows user to investigate 1 template and 1 dataset. A dataset can be output from 1 template and then used by the next.
• Cmdtest – this format involves the use of a Python script and allows the template to be called from within a script. Helpful for our test suite and potential for tasks like simulations.
• E-book – mixing up templates with textboxes to make executable books