Professor William Browne
School of Veterinary Science and Centre for Multilevel Modelling ( & many co-workers!)

The STAT-JR package
and it’s potential use with social network models
Summary

• Background to STAT-JR package
• Some screen shots of the program features
• Multiple membership models for spatial models/ social networks
• STAT-JR named in memory of Jon Rasbash whose ideas started project.
The E-STAT project and STAT-JR

STAT-JR developed jointly by LEMMA II and E-STAT ESRC nodes

Consists of a set of components many of which we have an alpha version for which contains:

Templates for model fitting, data manipulation, input and output controlled via a web browser interface.

Currently model fitting for 90% of the models that MLwiN can fit in MCMC plus some it can’t including greatly sped up REALCOM templates

Some interoperability with MLwiN, WinBUGS, R, Stata and SPSS (written by Camille Szmaragd)
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 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

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']

invars = ""
y = DataVector('response: ')
tau = ParamScalar()
sigma = ParamScalar()
x = DataMatrix('explanatory variables: ')
beta = ParamVector()
beta.ncols = len(x)

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)
    sigma <- 1 / sqrt(tau)
}
""

outlatex = r"
\begin{aligned}
\mbox{${y}}_i & \sim \mbox{N}(\mu_i, \sigma^2) \\
\mu_i & = \${mmulttex(x, r'\beta', 'i')} \\
\end{aligned}
""
Invars function

```python

    ""
    invars = ""
    y = DataVector('response: ')
    tau = ParamScalar()
    sigma = ParamScalar()
    x = DataMatrix('explanatory variables: ')
    beta = ParamVector()
    beta.ncols = len(x)
    ""
```
An example of STAT-JR – setting up a model

Stat-JR Demonstrator

Template: Regression1 Change Dataset tutorial Change View Summary

Configuration

Response: normexam
explanatory variables: cons, stand1
Name of output results: out

Random Seed: 1
Length of burnin: 1000
Number of iterations: 5000
Thinning: 1

Next

Set

Inputs: (Y, normexam, X, cons, stand1)
An example of STAT-JR – setting up a model

Configuration

response: normexam
explanatory variables: cons, standlrt
Name of output results: out
Random Seed: 1
length of burnin: 1000
number of iterations: 5000
thinning: 1
Equations for model and model code

\[ \text{normexam}_i \sim N(\mu_i, \sigma^2) \]
\[ \mu_i = \beta_0 \text{cons} + \beta_1 \text{standlrt}_i + u_{\text{school}_i} \]
\[ u_{\text{school}_i} \sim N(0, \sigma_u^2) \]
\[ \beta_0 \sim 1 \]
\[ \beta_1 \sim 1 \]
\[ \tau \sim \Gamma(0.001, 0.001) \]
\[ \sigma^2 = 1/\tau \]
\[ \sigma_u^2 = 1/\tau_u \]

Model

```r
model { 
  for (i in 1:length(normexam)) { 
    normexam[i] ~ dnorm(beta0 + beta1 * standlrt[i] + u_school[i] * cons[i], 
                         tau_u) 
  } 
  for (i in 1:length(u)) { 
    u[i] ~ dnorm(0, tau_u) 
  } 
  # Priors 
  beta0 ~ dnorm(0, tau_u) 
  beta1 ~ dnorm(0, tau_u) 
  tau ~ dgamma(0.001000, 0.001000) 
  sigma_u ~ dgamma(0.001000, 0.001000) 
} 
```

Note Equations use MATHJAX and so underlying LaTeX can be copied and paste. The model code is based around the WinBUGS language with some variation. This is a more complex template for 2 level models.
Equations for model and model code

\[
\text{normexam}_i \sim N(\mu_i, \sigma^2) \\
\mu_i = \beta_0 \text{cons}_i + \beta_1 \text{standlrt}_i + u_{\text{school}[i]} \\
u_{\text{school}[i]} \sim N(0, \sigma_u^2) \\
\beta_0 \propto 1 \\
\beta_1 \propto 1 \\
\tau \sim \Gamma(0.001, 0.001) \\
\sigma^2 = 1/\tau \\
\tau_u \sim \Gamma(0.001, 0.001) \\
\sigma_u^2 = 1/\tau_u
\]

Note Equations use MATHJAX and so underlying LaTeX can be copied and paste. The model code is based around the WinBUGS language with some variation. This is a more complex template for 2 level models.
Outbug function

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)
sigma <- 1 / sqrt(tau)
Model code in detail

```r
model {
  for (i in 1:length(normexam)) {
    normexam[i] ~ dnorm(mu[i], tau)
    mu[i] <- cons[i] * beta0 + standlrt[i] * beta1 + u[school[i]] * cons[i]
  }
  for (j in 1:length(u)) {
    u[j] ~ dnorm(0, tau_u)
  }
  # Priors
  beta0 ~ dflat()
  beta1 ~ dflat()
  tau ~ dgamma(0.001000, 0.001000)
  tau_u ~ dgamma(0.001000, 0.001000)
}
```

For this template the code is, aside from the length function, standard WinBUGS model code.
Bruce’s (Demo) algebra system step for parameter u

\[
\text{Log posterior} = -\tau \left( \sum_{i=1}^{\text{length(norexam)}} \text{cons} \left( -\text{normexam}_i + \beta_0 \text{cons}_1 + \beta_1 \text{standlrt}_i \right) \right) - \left( \frac{\text{tau}_u}{2} + \frac{\sum_{i=1}^{\text{length(norexam)}} \text{cons}_i^2}{2} \right) u_{ji}^2
\]

\[
\text{Distribution dnorm} = -\tau \left( \sum_{i=1}^{\text{length(norexam)}} \text{cons} \left( -\text{normexam}_i + \beta_0 \text{cons}_1 + \beta_1 \text{standlrt}_i \right) \right)
\]

\[
\text{Match} A = -\tau \left( \sum_{i=1}^{\text{length(norexam)}} \text{cons} \left( -\text{normexam}_i + \beta_0 \text{cons}_1 + \beta_1 \text{standlrt}_i \right) \right)
\]

\[
\text{Match} B = -\left( \frac{\text{tau}_u}{2} + \frac{\sum_{i=1}^{\text{length(norexam)}} \text{cons}_i^2}{2} \right) u_{ji}^2
\]

\[
\text{Sampling parameter} \mu = -\left( \frac{\tau \sum_{i=1}^{\text{length(norexam)}} \text{cons} \left( -\text{normexam}_i + \beta_0 \text{cons}_1 + \beta_1 \text{standlrt}_i \right)}{2 \left( \text{tau}_u / 2 + \tau \sum_{i=1}^{\text{length(norexam)}} \text{cons}_i^2 / 2 \right)} \right)
\]

\[
\text{Sampling parameter} \tau = 2 \left( \frac{\text{tau}_u}{2} + \frac{\sum_{i=1}^{\text{length(norexam)}} \text{cons}_i^2}{2} \right)
\]

\[
\text{Sampling distribution} u_{ji} \sim \text{dnorm} = -\left( \frac{\tau \sum_{i=1}^{\text{length(norexam)}} \text{cons} \left( -\text{normexam}_i + \beta_0 \text{cons}_1 + \beta_1 \text{standlrt}_i \right)}{2 \left( \text{tau}_u / 2 + \tau \sum_{i=1}^{\text{length(norexam)}} \text{cons}_i^2 / 2 \right)} \right)^2 - \left( \frac{\text{tau}_u}{2} + \frac{\sum_{i=1}^{\text{length(norexam)}} \text{cons}_i^2}{2} \right)
\]
Bruce’s (Demo) algebra system step for parameter $u$ 

\[
\begin{align*}
\text{Sampling parameter } \mu & = - \frac{\sum_{i=1 \atop \text{school}_i = j} \text{length}(\text{normexam}) \cdot \text{cons}_i \left( -\text{normexam}_i + \beta_0 \text{cons}_i + \beta_1 \text{standlrt}_i \right)}{2 \left( \frac{\tau_u}{2 + \tau} \sum_{i=1 \atop \text{school}_i = j} \text{length}(\text{normexam}) \cdot \text{cons}_i^2 / 2 \right) + \frac{\text{length}(\text{normexam})}{\tau} \sum_{i=1 \atop \text{school}_i = j} \text{cons}_i^2} \\
\text{Sampling parameter } \tau & = 2 \left( \frac{\tau_u}{2} + \frac{1}{2} \frac{\text{length}(\text{normexam})}{\sum_{i=1 \atop \text{school}_i = j} \text{cons}_i^2} \right)
\end{align*}
\]
The package can output C++ code that can then be taken away by software developers and modified.
The package can output C++ code that can then be taken away by software developers and modified.
Output from the E-STAT engine

Here the six-way plot functionality is in part taken over to STAT-JR after the model has run. In fact graphs for all parameters are calculated and stored as picture files so can be easily viewed quickly.
Output from the E-STAT engine

Here the six-way plot functionality is in part taken over to STAT-JR after the model has run. In fact graphs for all parameters are calculated and stored as picture files so can be easily viewed quickly.
Output from the E-STAT engine

can be easily viewed quickly.
Interoperability in the user interface is obtained via a few extra inputs. In fact in the template code user written functions are required for all packages apart from WinBUGS. The transfer of data between packages is however generic.
Interoperability with WinBUGS

Interoperability in the user interface is obtained via a few extra inputs. In fact in the template code user written functions are required for all packages apart from WinBUGS. The transfer of data between packages is however generic.
Output from WinBUGS with multiple chains

STAT-JR generates appropriate files and then fires up WinBUGS. Multiple Chains are superimposed in the sixway plot output.
Output from WinBUGS with multiple chains

beta0

Parameter

Kernel density

ACF

PACF

MCSE

Output.
Multiple Membership Models

- Example is Scottish lip cancer data
- Response is Poisson (number of cases)
- Use as offset expected cases based on population size, makeup
- One predictor – percaff – percentage in agriculture, farming, fishing.
- Use the template MultipleMembershipNLev to allow both own random effect and neighbour random effects
- Template will allow fitting in STAT-JR engine, WinBUGS or MLwiN.
Inputs for model

Template: MultipleMembershipNL.ev  Change Dataset: lips Change View Summary

Configuration

response: obs
Number of Classifications: 1
Classification 1: area
MM IDs: neigh1, neigh2, neigh3, neigh4, neigh5, neigh6, neigh7, neigh8, neigh9, neigh10, neigh11
MM weights: weight1, weight2, weight3, weight4, weight5, weight6, weight7, weight8, weight9, weight10, weight11
specify distribution: Poisson
Is there an offset?: Yes
offset: offs
explanatory variables: cons, perc_aff
Name of output results: out

Next
LaTeX for Model

\begin{equation}
\begin{align*}
\text{obs}_i & \sim \text{Poisson}(\pi_i) \\
\ln(\pi_i) &= \text{offs}_i + \beta_0 \text{cons}_i + \beta_1 \text{perc}_\text{aff}_i + \sum_{j \in \text{neigh1}[i]} w_{(i,j)} u_j \\
+ \mathbf{u}_{\text{area}[i]}^{(2)} &\sim N(0, \sigma^2_u) \\
\mathbf{u}_{\text{area}[i]}^{(2)} &\sim N(0, \sigma^2_u) \\
\beta_0 &\sim 1 \\
\beta_1 &\sim 1 \\
\tau_u &\sim \Gamma(0.001, 0.001) \\
\sigma^2_u &= 1/\tau_u \\
\tau_{u1} &\sim \Gamma(0.001, 0.001) \\
\sigma^2_{u1} &= 1/\tau_{u1}
\end{align*}
\end{equation}

Model

```r
model {
  for (i in 1:length(obs)) {
    obs[i] ~ poisson(mu[i]) 
    sum(ones(19) * (col(i) - 12) * u[j] * w[i,j])
  } 
}
```
Model Code

```r
model {
  for (i in 1:length(ys)) {
    obs[i] ~ dnorm(p[i], tau)
  }
  for (j in 1:length(u)) {
    u[j] <- dnorm(0, tau_u)
  }
  for (k in 1:length(w)) {
    w[k] <- dnorm(0, tau_w)
  }
  # Priors
  beta0 ~ dflat()
  beta1 ~ dflat()
  tau ~ dgamma(0.001000, 0.001000)
  sigma2_u ~ 1 / tau_u
  tau_u ~ dgamma(0.001000, 0.001000)
  sigma2_w ~ 1 / tau_w
}
```
The E-STAT project – still to come

We have lots of work to do:

• Parallel processing.
• E-books.
• Optimising code generation.
• Improving algebra system.
• Suites of templates for missing data and social network models.
• Interoperability with SAS and hooking up more templates for other packages.