R2MLwiN
Using the multilevel modelling software package MLwiN from R

Richard Parker
Zhengzheng Zhang
Chris Charlton
George Leckie
Bill Browne

Centre for Multilevel Modelling (CMM)
University of Bristol
R2MLwiN
Using the multilevel modelling software package MLwiN from R
R2MLwiN
Using the multilevel modelling software package MLwiN from R
R2MLwiN
Using the multilevel modelling software package MLwiN from R
R2MLwiN
Using the multilevel modelling software package MLwiN from R
R2MLwiN
Using the multilevel modelling software package MLwiN from R
Why operate one application from another?
Why operate one application from another?

Perhaps this package (B) has functionality / facilities of value to user that this package (A) doesn’t.
Why operate one application from another?

Perhaps this package (B) has functionality / facilities of value to user that this package (A) doesn’t

E.g. package B can:
Why operate one application from another?

Perhaps this package (B) has functionality / facilities of value to user that this package (A) doesn’t.

E.g. package B can:
- fit a certain type of statistical model
Why operate one application from another?

Perhaps this package (B) has functionality / facilities of value to user that this package (A) doesn’t

E.g. package B can:

• fit a certain type of statistical model
• use a quicker / less-biased / etc. means of estimation
Why operate one application from another?

E.g. package **B** can:
• fit a certain type of statistical model
• use a quicker / less-biased / etc. means of estimation
• produce a certain type of plot, table, statistic, etc.

Perhaps this package (**B**) has functionality / facilities of value to user that this package (**A**) doesn’t.
Why operate one application from another?

Perhaps this package (B) has functionality / facilities of value to user that this package (A) doesn’t.

E.g. package B can:
• fit a certain type of statistical model
• use a quicker / less-biased / etc. means of estimation
• produce a certain type of plot, table, statistic, etc.

...or perhaps package B is:
Why operate one application from another?

Perhaps this package (B) has functionality / facilities of value to user that this package (A) doesn’t

E.g. package B can:
• fit a certain type of statistical model
• use a quicker / less-biased / etc. means of estimation
• produce a certain type of plot, table, statistic, etc.

...or perhaps package B is:
• more familiar
Why operate one application from another?

E.g. package B can:
• fit a certain type of statistical model
• use a quicker / less-biased / etc. means of estimation
• produce a certain type of plot, table, statistic, etc.

...or perhaps package B is:
• more familiar
• has useful supporting resources
Why operate one application from another?
Why operate one application from another?

The user could (in principle) operate package B directly...
Why operate one application from another?

The user could (in principle) operate package B directly...
...but wants package A to do so on their behalf
Why operate one application from another?

The user could (in principle) operate package B directly...
...but wants package A to do so on their behalf

- The user might *not know* how to operate Package B
Why operate one application from another?

The user could (in principle) operate package B directly...
...but wants package A to do so on their behalf

- The user might *not know* how to operate Package B
  - and it’s not realistic to learn how to do so given time, etc., available
The user could (in principle) operate package **B** directly...
...but wants package **A** to do so on their behalf

- The user might *not know* how to operate Package **B**
  - and it’s not realistic to learn how to do so given time, etc., available
  - or the user wants to use **A** as a means of learning how to use **B** directly him/herself
Why operate one application from another?

The user could (in principle) operate package B directly...
...but wants package A to do so on their behalf.
Why operate one application from another?

The user could (in principle) operate package B directly...
...but wants package A to do so on their behalf

• Or perhaps the user *does know* how to directly operate package B, but it’s more convenient if A does so for him/her, because e.g.:
The user could (in principle) operate package B directly...
...but wants package A to do so on their behalf

- Or perhaps the user *does know* how to directly operate package B, but it’s more convenient if A does so for him/her, because e.g.:
  - data’s already in A and it’s time-consuming to translate it, oneself, into package B’s format
Why operate one application from another?

The user could (in principle) operate package B directly...
...but wants package A to do so on their behalf

- Or perhaps the user *does know* how to directly operate package B, but
  it’s more convenient if A does so for him/her, because e.g.:
  - data’s already in A and it’s time-consuming to translate it, oneself,
    into package B’s format
  - can more efficiently fit many models / produce many plots / etc. in
    B if run it from A
Why operate one application from another?

The user could (in principle) operate package B directly...
...but wants package A to do so on their behalf

- Or perhaps the user does know how to directly operate package B, but it’s more convenient if A does so for him/her, because e.g.:
  - data’s already in A and it’s time-consuming to translate it, oneself, into package B’s format
  - can more efficiently fit many models / produce many plots / etc. in B if run it from A
  - wish to use A’s functionality to post-process the results from B
Why operate one application from another?

The user could (in principle) operate package B directly...
...but wants package A to do so on their behalf

- Or perhaps the user *does know* how to directly operate package B, but it’s more convenient if A does so for him/her, because e.g.:
  - data’s already in A and it’s time-consuming to translate it, oneself, into package B’s format
  - can more efficiently fit many models / produce many plots / etc. in B if run it from A
  - wish to use A’s functionality to post-process the results from B
  - may be able to better document analyses in A
Why operate one application from another?

The user could (in principle) operate package B directly...
...but wants package A to do so on their behalf

• Or perhaps the user *does know* how to directly operate package B, but it’s more convenient if A does so for him/her, because e.g.:
  – data’s already in A and it’s time-consuming to translate it, oneself, into package B’s format
  – can more efficiently fit many models / produce many plots / etc. in B if run it from A
  – wish to use A’s functionality to post-process the results from B
  – may be able to better document analyses in A
  – may want to compare model fits from many packages...
R2MLwiN
Using the multilevel modelling software package MLwiN from R
R2MLwiN

Using the multilevel modelling software package MLwiN from R
R2MLwiN
Using the multilevel modelling software package MLwiN from R

- First released (on CRAN) December 2012
- Most recent release (0.8-0) March 2015...
  ...this has a number of new features / syntactical changes
  ...but back-compatibility maintained where possible
R2MLwiN

MLwiN

• Developed & maintained by Centre for Multilevel Modelling (now at Bristol)
• Estimated 18,000 users worldwide
• Large body of supporting documentation, examples, workshops, etc.
• As well as Windows, native versions of MLwiN engine for Mac OS X and Linux now available too
MLwiN

• Allows for a variety of response types to be modelled, including:
  – continuous
  – binary
  – count
  – ordinal
  – nominal
  – multivariate combinations (i.e., simultaneous equations)

• Estimation available via:
  – IGLS (iterative generalised least squares), which yields maximum likelihood estimates
  – MCMC (Markov chain Monte Carlo) estimation for Bayesian inference

• Supported data structures:
  – nested, cross-classified and/or multiple membership

• Other features include:
  – fitting of complex level 1 variance (heteroskedastic) models
  – multilevel factor analysis (MCMC only)
  – adjustments for measurement errors in predictors
  – spatial conditional auto regressive (CAR) models
  – autoregressive structures at level 1
  – a selection of MCMC algorithms to increase efficiency.
MLwiN

- GUI (graphical user interface) has number of innovative features, e.g.:
- Interactive equations window:
MLwiN

- GUI (graphical user interface) has number of innovative features, e.g.:
- Interactive equations window:
MLwiN

- GUI (graphical user interface) has number of innovative features, e.g.:
- Interactive equations window:

\[
\begin{align*}
\text{normexam}_{ij} & \sim N(XB, \Omega) \\
\text{normexam}_{ij} & = \beta_{0ij}\text{cons} + \beta_{1ij}\text{standlrt}_{ij} + \beta_{2ij}\text{girl}_{ij} \\
\beta_{0ij} & = \beta_0 + u_{0j} + e_{0ij} \\
\beta_{1ij} & = \beta_1 + u_{1j} \\
\begin{bmatrix} u_{0j} \\ u_{1j} \end{bmatrix} & \sim N(0, \Omega_u) : \Omega_u = \begin{bmatrix} \sigma_{u0}^2 \\ \sigma_{u01} \sigma_{u1}^2 \end{bmatrix} \\
\begin{bmatrix} e_{0ij} \end{bmatrix} & \sim N(0, \Omega_e) : \Omega_e = \begin{bmatrix} \sigma_{e0}^2 \end{bmatrix} \\
-2\text{loglikelihood(IGLS Deviance)} & = 9287.387 (4059 of 4059 cases in use)
\end{align*}
\]
MLwiN

- GUI (graphical user interface) has number of innovative features, e.g.:

- Interactive equations window:

\[
\begin{align*}
normexam_{ij} & \sim N(XB, \Omega) \\
normexam_{ij} & = \beta_{0ij} + \beta_{1ij} + 0.176(0.032)girl_{ij} \\
\beta_{0ij} & = -0.112(0.043) + u_{0j} + e_{0ij} \\
\beta_{1ij} & = 0.553(0.020) + u_{1j} \\
\begin{bmatrix} u_{0j} \\ u_{1j} \end{bmatrix} & \sim N(0, \Omega_u) : \Omega_u = \begin{bmatrix} 0.086(0.017) \\ 0.019(0.007) & 0.015(0.004) \end{bmatrix} \\
\begin{bmatrix} e_{0ij} \end{bmatrix} & \sim N(0, \Omega_e) : \Omega_e = \begin{bmatrix} 0.550(0.012) \end{bmatrix} \\
-2*\text{loglikelihood}(IGLS Deviance) & = 9287.387 (4059 of 4059 cases in use)
\end{align*}
\]
MLwiN

• GUI (graphical user interface) has number of innovative features, e.g.:

• Interactive graphs:
Most users will likely operate MLwiN via GUI.
Macro language can be unwieldy.
R2MLwiN
Using the multilevel modelling software package MLwiN from R
Type 'demo()' for some demos, 'help()' for on-line help, or 'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

[Previously saved workspace restored]

> library("R2MLwiN")
Loading required package: stats4
Loading required package: lattice
Loading required package: coda
The MLwiN_path option is currently set to C:/Program Files (x86)/MLwiN v2.32/
To change this use: options(MLwiN_path="<path to MLwiN>")
Type 'demo()' for some demos, 'help()' for on-line help, or 'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

[Previously saved workspace restored]

> library("R2MLwiN")
Loading required package: stats4
Loading required package: lattice
Loading required package: coda
The MLwiN_path option is currently set to C:/Program Files (x86)/MLwiN v2.32/
To change this use: options(MLwiN_path="<path to MLwiN>")
Using the multilevel modelling software package MLwiN from R

Tells user where it's expecting to find MLwiN; easy to change path if it's elsewhere
Type 'demo()' for some demos, 'help()' for on-line help, or 'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

[Previously saved workspace restored]

> library("R2MLwiN")
Loading required package: stats4
Loading required package: lattice
Loading required package: coda
The MLwiN_path option is currently set to C:/Program Files (x86)/MLwiN v2.32/
To change this use: options(MLwiN_path="<path to MLwiN>")

> data("tutorial")
R2MLwiN

Type 'demo()' for some demos, 'help()' for on-line help, or 'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

[Previously saved workspace restored]

> library("R2MLwiN")
Loading required package: stats4
Loading required package: lattice
Loading required package: coda
The MLwiN_path option is currently set to C:/Program Files (x86)/MLwiN v2.32/
To change this use: options(MLwiN_path="<path to MLwiN>")

> data("tutorial")

- All sample datasets released with MLwiN available with R2MLwiN
Type 'demo()' for some demos, 'help()' for on-line help, or 'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

[Previously saved workspace restored]

> library("R2MLwiN")
Loading required package: stats4
Loading required package: lattice
Loading required package: coda
The MLwiN_path option is currently set to C:/Program Files (x86)/MLwiN v2.32/
To change this use: options(MLwiN_path="<path to MLwiN>")

> data("tutorial")
> F1 <- normexam ~ 1 + (1|school) + (1|student)
Type 'demo()' for some demos, 'help()' for on-line help, or 'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

[Previously saved workspace restored]

> library("R2MLwiN")
Loading required package: stats4
Loading required package: lattice
Loading required package: coda
The MLwiN_path option is currently set to C:/Program Files (x86)/MLwiN v2.32/
To change this use: options(MLwiN_path="<path to MLwiN>"

> data("tutorial")
> F1 <- normexam ~ 1 + (1|school) + (1|student)
R> F1 <- normexam ~ 1 + (1|school) +
   +       (1|student)
normexam_{ij} = \beta_0 + u_j + e_{ij}

u_j \sim \text{N}(0, \sigma_u^2)

e_{ij} \sim \text{N}(0, \sigma_e^2)
R2MLwiN

R> F1 <- normexam ~ 1 + (1|school) + 
+ (1|student)

\[ \text{normexam}_{ij} = \beta_0 + u_j + e_{ij} \]

\[ u_j \sim \text{N}(0, \sigma_u^2) \]

\[ e_{ij} \sim \text{N}(0, \sigma_e^2) \]

- Note: need to explicitly add intercept (as in MLwiN)
- Specify random part of model in order of hierarchy
Type 'demo()' for some demos, 'help()' for on-line help, or 'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

[Previously saved workspace restored]

> library("R2MLwiN")
Loading required package: stats4
Loading required package: lattice
Loading required package: coda
The MLwiN_path option is currently set to C:/Program Files (x86)/MLwiN v2.32/
To change this use: options(MLwiN_path="<path to MLwiN>")

> data("tutorial")
> F1 <- normexam ~ 1 + (1|school) + (1|student)"
Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

[Previously saved workspace restored]

> library("R2MLwiN")
Loading required package: stats4
Loading required package: lattice
Loading required package: coda
The MLwiN_path option is currently set to C:/Program Files (x86)/MLwiN v2.32/
To change this use: options(MLwiN_path="<path to MLwiN>")

> data("tutorial")
> F1 <- normexam ~ 1 + (1|school) + (1|student)
> (VarCompModel <- runMLwiN(Formula = F1, data = tutorial))
Type 'demo()' for some demos, 'help()' for on-line help, or 'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

[Previously saved workspace restored]

> library("R2MLwiN")
Loading required package: stats4
Loading required package: lattice
Loading required package: coda
The MLwiN_path option is currently set to C:/Program Files (x86)/MLwiN v2.32/
To change this use: options(MLwiN_path="<path to MLwiN>")

> data("tutorial")
> F1 <- normexam ~ 1 + (1|school) + (1|student)
> (VarCompModel <- runMLwiN(Formula = F1, data = tutorial))
r2mlwin

R> F1 <- normexam ~ 1 + (1|school) + (1|student)

R> (VarCompModel <- runMLwiN(
+    Formula = F1, data = tutorial))
R2MLwiN

R> F1 <- normexam ~ 1 + (1|school) + (1|student)

R> (VarCompModel <- runMLwiN(
+   Formula = F1, data = tutorial))
R2MLwiN

R> F1 <- normexam ~ 1 + (1|school) + 
+      (1|student)

R> (VarCompModel <- runMLwiN(
+    Formula = F1, data = tutorial))

runMLwiN function:
1. takes input & creates MLwiN macro file
2. calls MLwiN and executes macro script
3. output is returned to R for post-processing
runMLwiN function

- Arguments include:
  - `Formula`
  - `data`
  - `D` ...since we don’t specify here, using default: `D = "Normal"
  - `estoptions` ...again using default, which is IGLS: `estoptions = list(EstM = 0)`
  - See `?runMLwiN` for full list of arguments

R> F1 <- normexam ~ 1 + (1|school) + (1|student)

R> (VarCompModel <- runMLwiN(
+   Formula = F1, data = tutorial))
R2MLwiN

R> F1 <- normexam ~ 1 + (1|school) +
  +     (1|student)
R> (VarCompModel <- runMLwiN(
  +     Formula = F1, data = tutorial))

runMLwiN function
• Arguments include:
  ➢ Formula
  ➢ data
  ➢ D ...since we don’t specify here, using default:
    D = “Normal”
  ➢ estoptions ...again using default, which is IGLS:
    estoptions = list(EstM = 0)
  ➢ See ?runMLwiN for full list of arguments
runMLwiN {R2MLwiN}

Description

This function executes MLwiN and then brings results back to R.

Usage

runMLwiN(Formula, levID = NULL, D = "Normal", data = NULL,
          estoptions = list(EstM = 0), BUGO = NULL, MLwinPath = NULL,
          stdout = "", stderr = "", workdir = tempdir(), checkversion = TRUE,
          indata = NULL)

Arguments

Formula

A formula object specifying the model formula. See Formula.translate (Formula.translate.comapt) details back-compatible functionality for deprecated syntax used in versions of R2MLwiN prior to 0.8-0 and also ‘Details’ below.

levID

A character vector specifying the level ID(s). Deprecated syntax: by default this is NULL and level ID(s) are specified in the Formula object.

D

A character string/vector specifying the type of distribution to be modelled, which can include 'Normal' (the default), 'Binomial', 'Poisson', 'Negbinom', 'Unordered Multinomial', 'Ordered Multinomial', 'Multivariate Normal', or 'Mixed'. In the case of the latter, 'Mixed' precedes the response types which also need to be listed in D, e.g. c("Mixed", "Normal", "Binomial"); these need to be be listed in the same order to which they are referred to in the Formula object (see Formula.translate, Formula.translate.comapt). For (R)IGLS estimation (i.e. EstM = 0 in estoptions) 'Mixed' combinations can consist of 'Normal' and 'Binomial' or 'Normal' and 'Poisson'; for MCMC estimation (i.e. EstM = 0), on the other hand, only a combination of 'Normal' and 'Binomial' is available.

data

A data.frame object containing the data to be modelled. Optional (but recommended): if empty, data taken from environment of formula.

estoptions

A list of options used for estimating the model. See ‘Details’ below.
### Distribution

<table>
<thead>
<tr>
<th>Distribution</th>
<th>Format of formula object</th>
</tr>
</thead>
<tbody>
<tr>
<td>'Normal'</td>
<td>( y_1 \sim 1 + x_1 + (1</td>
</tr>
<tr>
<td>'Poisson'</td>
<td>( \text{link}((y_1) \sim 1 + \text{offset}(\text{offs}) + x_1 + (1</td>
</tr>
<tr>
<td>'NegBinom*'</td>
<td>( \text{link}((y_1) \sim 1 + \text{offset}(\text{offs}) + (1</td>
</tr>
<tr>
<td>'Binomial'</td>
<td>( \text{link}((y_1, \text{denom}) \sim 1 + x_1 + (1</td>
</tr>
<tr>
<td>'Unordered Multinomial'</td>
<td>( \text{link}((y_1, \text{denom}, \text{ref_cat}) \sim 1 + x_1 + (1</td>
</tr>
<tr>
<td>'Ordered Multinomial'</td>
<td>( \text{link}((y_1, \text{denom}, \text{ref_cat}) \sim 1 + x_1 + x_2[\text{common}] + (1[\text{common}]</td>
</tr>
<tr>
<td>'Multivariate Normal'</td>
<td>( c((y_1, y_2, \ldots) \sim 1 + x_1 + x_2[\text{common}] + (1[\text{common}]</td>
</tr>
<tr>
<td>c('Mixed', 'Normal', 'Binomial')</td>
<td>( c((y_1, \ldots, \text{link}((y_2, \text{denom}, \ldots \sim 1 + x_1 + x_2[\text{common}] + (1[\text{common}]</td>
</tr>
<tr>
<td>c('Mixed', 'Normal', 'Poisson')*</td>
<td>( c((y_1, \ldots, \text{link}((y_2, \text{offset}, \ldots \sim 1 + x_1 + x_2[\text{common}] + (1[\text{common}]</td>
</tr>
</tbody>
</table>

* denotes IGLS only in the table below.

### Where link can equal...

- (identity link assumed)
- \( \log \)
- \( \log \)
- \( \logit, \probit, \text{cloglog} \)
- \( \logit \)
- \( \logit, \probit, \text{cloglog} \)
- \( \logit, \probit, \text{cloglog} \)
- \( \logit, \probit, \text{cloglog} \)

The argument estoptions is a list which can contain the following options used for estimating the model:

- \( \text{EstM} \): specifies estimation method. When \( \text{EstM} = 0 \) (default), estimation method is (R)IGLS, otherwise \( \text{EstM} = 1 \) specifies MCMC estimation.
- \( \text{resi.store} \): a logical value indicating whether residuals are to be stored or not. Defaults to FALSE.
- \( \text{resioptions} \): a string vector to specify the various residual options. The 'variance' option calculates the posterior variances instead of the posterior standard errors; the 'standardised', 'leverage', 'influence' and 'deletion' options calculate standardised, leverage, influence and deletion residuals respectively; the 'sampling' option calculates the sampling variance covariance matrix for the residuals; the 'norecode' option prevents residuals with values exceedingly close or equal to zero from being recoded to missing. When \( \text{EstM} = 1 \) (i.e. MCMC estimation) 'variance' is default value, and the only other permissible value is 'standardised' (else function call stopped with appropriate error message). When \( \text{EstM} = 0 \) (i.e. (R)IGLS estimation), 'variance' cannot be specified together.
**R2MLwiN**

---

MLwiN (version: 2.32) multilevel model (Normal)

N min mean max

school 65 2 62.44615 198

Estimation algorithm: IGLS Elapsed time : 0.37s

Number of obs: 4059 (from total 4059) The model converged after 3 iterations.

Log likelihood: -5505.3

Deviance statistic: 11010.6

---

The model formula:

normexam ~ 1 + (1 | school) + (1 | student)

Level 2: school Level 1: student

---

The fixed part estimates:

| Coef.  | Std. Err. | z     | Pr(>|z|) | [95% Conf. Interval] |
|--------|-----------|-------|----------|-----------------------|
| Intercept | -0.01317  | 0.05363 | -0.25 | 0.806 | -0.11827    | 0.09194 |

Signif. codes: 0 "***" 0.001 "**" 0.01 "*" 0.05 '. ' 1

---

The random part estimates at the school level:

<table>
<thead>
<tr>
<th>Coef.</th>
<th>Std. Err.</th>
</tr>
</thead>
<tbody>
<tr>
<td>var_Intercept</td>
<td>0.16863</td>
</tr>
</tbody>
</table>

---

The random part estimates at the student level:

<table>
<thead>
<tr>
<th>Coef.</th>
<th>Std. Err.</th>
</tr>
</thead>
<tbody>
<tr>
<td>var_Intercept</td>
<td>0.84776</td>
</tr>
</tbody>
</table>
R2MLwiN

MLwiN (version: 2.32) multilevel model (Normal)

N min mean max
school 65 2 62.44615 198

Estimation algorithm: IGLS Elapsed time: 0.37s
Number of obs: 4059 (from total 4059) The model converged after 3 iterations.
Log likelihood: -5505.3
Deviance statistic: 11010.6

The model formula:
normexam ~ 1 + (1 | school) + (1 | student)
Level 2: school Level 1: student

The fixed part estimates:

| Coef. | Std. Err. | z     | Pr(>|z|) | [95% Conf. Interval] |
|-------|-----------|-------|---------|----------------------|
| Intercept | -0.01317 | 0.05363 | -0.25 | 0.806 | -0.11827 0.09194 |

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

The random part estimates at the school level:

<table>
<thead>
<tr>
<th>Coef.</th>
<th>Std. Err.</th>
</tr>
</thead>
<tbody>
<tr>
<td>var_Intercept</td>
<td>0.16863 0.03245</td>
</tr>
</tbody>
</table>

The random part estimates at the student level:

<table>
<thead>
<tr>
<th>Coef.</th>
<th>Std. Err.</th>
</tr>
</thead>
<tbody>
<tr>
<td>var_Intercept</td>
<td>0.84776 0.01897</td>
</tr>
</tbody>
</table>
MLwiN (version: 2.32)  multilevel model (Normal)

N  min  mean  max
school 65  2  62.44615  198

Estimation algorithm:  IGLS  Elapsed time : 0.37s
Number of obs: 4059 (from total 4059)  The model converged after 3 iterations.
Log likelihood: -5505.3
Deviance statistic: 11010.6

The model formula:
normexam ~ 1 + (1 | school) + (1 | student)
Level 2: school  Level 1: student

The fixed part estimates:

| Coef.  | Std. Err. | z     | Pr(>|z|) | [95% Conf. Interval] |
|--------|-----------|-------|---------|----------------------|
| Intercept | -0.01317  | 0.05363 | -0.25   | 0.806                | -0.11827 to 0.09194 |

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

The random part estimates at the school level:

<table>
<thead>
<tr>
<th>Coef.</th>
<th>Std. Err.</th>
</tr>
</thead>
<tbody>
<tr>
<td>var Intercept</td>
<td>0.16863</td>
</tr>
</tbody>
</table>

The random part estimates at the student level:

<table>
<thead>
<tr>
<th>Coef.</th>
<th>Std. Err.</th>
</tr>
</thead>
<tbody>
<tr>
<td>var Intercept</td>
<td>0.84776</td>
</tr>
</tbody>
</table>
MLwiN (version: 2.32) multilevel model (Normal)

N min mean max
school 65 2 62.44615 198

Estimation algorithm: IGLS  
Elapsed time: 0.37s

Number of obs: 4059 (from total 4059)  
The model converged after 3 iterations.
Log likelihood: -5505.3
Deviance statistic: 11010.6

The model formula:
normexam ~ 1 + (1 | school) + (1 | student)

Level 2: school  
Level 1: student

The fixed part estimates:

| Coef.  | Std. Err. | z     | Pr(>|z|) | 95% Conf. Interval |
|--------|-----------|-------|----------|-------------------|
| Intercept | -0.01317 | 0.05363 | -0.25 | 0.806 | -0.11827 | 0.09194 |

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

The random part estimates at the school level:

<table>
<thead>
<tr>
<th>Coef.</th>
<th>Std. Err.</th>
</tr>
</thead>
<tbody>
<tr>
<td>var_Intercept</td>
<td>0.16863</td>
</tr>
</tbody>
</table>

The random part estimates at the student level:

<table>
<thead>
<tr>
<th>Coef.</th>
<th>Std. Err.</th>
</tr>
</thead>
<tbody>
<tr>
<td>var_Intercept</td>
<td>0.84776</td>
</tr>
</tbody>
</table>
MLwiN (version: 2.32)  multilevel model (Normal)

N min    mean    max
school 65  2 62.44615 198

Estimation algorithm: IGLS  Elapsed time: 0.37s

Number of obs: 4059 (from total 4059)  The model converged after 3 iterations.

Log likelihood: -5505.3

Deviance statistic: 11010.6

The model formula:
normexam ~ 1 + (1 | school) + (1 | student)

Level 2: school    Level 1: student

The fixed part estimates:

| Coef.  | Std. Err. | z     | Pr(>|z|) | [95% Conf. Interval] |
|--------|-----------|-------|---------|----------------------|
| Intercept | -0.01317  | 0.05363 | -0.25   | 0.806                |

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

The random part estimates at the school level:

<table>
<thead>
<tr>
<th>Coef.</th>
<th>Std. Err.</th>
</tr>
</thead>
<tbody>
<tr>
<td>var Intercept</td>
<td>0.16863</td>
</tr>
</tbody>
</table>

The random part estimates at the student level:

<table>
<thead>
<tr>
<th>Coef.</th>
<th>Std. Err.</th>
</tr>
</thead>
<tbody>
<tr>
<td>var Intercept</td>
<td>0.84776</td>
</tr>
</tbody>
</table>
R2MLwiN

R> F2 <- normexam ~ 1 + (1|student)
R2MLwiN

R> F2 <- normexam ~ 1 + (1|student)
R> OneLevelModel <- runMLwiN(
+   Formula = F2, data = tutorial)
R2MLwiN

R> F2 <- normexam ~ 1 + (1|student)
R> OneLevelModel <- runMLwiN(
  +    Formula = F2, data = tutorial)
R> library("lmtest")
R> lrtest(OneLevelModel, VarCompModel)

Model objects returned by R2MLwiN contain some generic S4 methods...

• e.g. has a method for the function logLik which allows us to conduct a likelihood ratio test using the lrtest function (part of the lmtest package)
Attaching package: ‘zoo’

The following objects are masked from ‘package:base’:

  as.Date, as.Date.numeric

Warning messages:
1: package ‘lmtest’ was built under R version 3.1.3
2: package ‘zoo’ was built under R version 3.1.3

> lrtest(OneLevelModel, VarCompModel)
Likelihood ratio test

Model 1: OneLevelModel
Model 2: VarCompModel

  Df LogLik Df Chisq Pr(>Chisq)
1  2  -5754.7
2  3  -5505.3  1  498.72  < 2.2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

> |
> lrtest(OneLevelModel, VarCompModel)
Likelihood ratio test

Model 1: OneLevelModel
Model 2: VarCompModel

<table>
<thead>
<tr>
<th></th>
<th>Df</th>
<th>LogLik</th>
<th>Df</th>
<th>Chisq</th>
<th>Pr(&gt;Chisq)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>-5754.7</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>-5505.3</td>
<td>1</td>
<td>498.72</td>
<td>&lt; 2.2e-16</td>
</tr>
</tbody>
</table>

---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

> slotNames(VarCompModel)

[1] "version"   "Nobs"   "DataLength"
[4] "Hierarchy" "D"     "Formula"
[7] "levID"     "FP"     "RP"
[10] "RP.cov"    "FP.cov" "LIKE"
[13] "elapsed.time" "call" "residual"
[16] "Converged" "Iterations" "Meth"
[19] "nonlinear" "data"
2: package 'zoo' was built under R version 3.1.3

> lrtest(OneLevelModel, VarCompModel)
Likelihood ratio test

Model 1: OneLevelModel
Model 2: VarCompModel

# Df LogLik Df Chisq Pr(>Chisq)
1  2  -5754.7
2  3  -5505.3  1 498.72  < 2.2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

> slotNames(VarCompModel)
[1] "version"  "Nobs"   "DataLength"
[4] "Hierarchy" "D"     "Formula"
[7] "levID"     "FP"     "RP"
[10] "RP.cov"    "FP.cov" "LIKE"
[13] "elapsed.time" "call" "residual"
[16] "Converged" "Iterations" "Meth"
[19] "nonlinear" "data"
Model 1: OneLevelModel
Model 2: VarCompModel

<table>
<thead>
<tr>
<th>Df</th>
<th>LogLik</th>
<th>Df</th>
<th>Chisq</th>
<th>Pr(&gt;Chisq)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>2</td>
<td>-5754.7</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>1</td>
<td>498.72</td>
<td>&lt; 2.2e-16</td>
</tr>
</tbody>
</table>

Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

> slotNames(VarCompModel)

[1] "version"  "Nobs"  "DataLength"
[4] "Hierarchy" "D"    "Formula"
[7] "levID"    "FP"    "RP"
[10] "RP.cov"   "FP.cov" "LIKE"
[13] "elapsed.time" "call"  "residual"
[16] "Converged" "Iterations" "Meth"
[19] "nonlinear" "data"

> LRT <- OneLevelModel["LIKE"] - VarCompModel["LIKE"]
> pchisq(LRT, 1, lower.tail = F)

[1] 1.807882e-110
> |
Model 1: OneLevelModel
Model 2: VarCompModel

#Df  LogLik Df  Chisq Pr(>Chisq)
1   2  -5754.7
2   3  -5505.3  1  498.72  < 2.2e-16 ***

---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

> slotNames(VarCompModel)
[1] "version" "Nobs" "DataLength"
[4] "Hierarchy" "D" "Formula"
[7] "levID" "FP" "RP"
[10] "RP.cov" "FP.cov" "LIKE"
[13] "elapsed.time" "call" "residual"
[16] "Converged" "Iterations" "Meth"
[19] "nonlinear" "data"

> LRT <- OneLevelModel["LIKE"] - VarCompModel["LIKE"]
> pchisq(LRT, 1, lower.tail = T)
[1] 1.807882e-110

>
R> F3 <- normexam ~ 1 + standlrt +
+ (1 + standlrt | school) + (1 | student)

R> (RandomSlopeModel <- runMLwiN(
+ Formula = F3, estoptions = list(
+ resi.store = TRUE), data = tutorial))
R> F3 <- normexam ~ 1 + standlrt +
+    (1 + standlrt | school) + (1 | student)

R> (RandomSlopeModel <- runMLwiN(
+    Formula = F3, estoptions = list(
+        resi.store = TRUE), data = tutorial))
R> F3 <- normexam ~ 1 + standlrt + 
+     (1 + standlrt | school) + (1 | student)

R> (RandomSlopeModel <- runMLwiN(
+    Formula = F3, estoptions = list(
+        resi.store = TRUE), data = tutorial))

R> predLines(RandomSlopeModel,
+    xname="standlrt", lev = 2, legend = F)
```r
R> normexam ~ 1 + standlrt + (1 + standlrt | school) + (1 | student)
R> runMLwiN(
R>   Formula = F3, estoptions = list(
R>     resi.store = TRUE), data = tutorial)
R> predLines(RandomSlopeModel,
R>   xname="standlrt", lev = 2, legend = F)
```
R> predLines(RandomSlopeModel,
+    xname = "standlrt", lev = 2,
+    selected = c(30, 44, 53, 59))
R> predLines(RandomSlopeModel,
+   xname = "standlrt", lev = 2,
+   selected = c(30, 44, 53, 59))
R> F4 <- normexam ~ 1 + standlrt +
+   (1 + standlrt | school) +
+   (1 + standlrt | student)

R> ComplexLevOneModel <- runMLwiN(
+   Formula = F4, data = tutorial,
+   estoptions = list(debugmode = TRUE))
R2MLwiN

R> F4 <- normexam ~ 1 + standlrt +
+   (1 + standlrt | school) +
+   (1 + standlrt | student)

R> ComplexLevOneModel <- runMLwiN(
+   Formula = F4, data = tutorial,
+   estoptions = list(debugmode = TRUE),
\[ \text{normexam}_{ij} \sim N(XB, \Omega) \]
\[ \text{normexam}_{ij} = \beta_{0ij}\text{Intercept} + \beta_{1ij}\text{standlrt}_{ij} \]
\[ \beta_{0ij} = -0.012(0.040) + u_{0j} + e_{0ij} \]
\[ \beta_{1ij} = 0.558(0.020) + u_{1j} + e_{1ij} \]

\[
\begin{bmatrix}
u_{0j} \\
u_{1j}
\end{bmatrix} \sim N(0, \Omega_u) : \Omega_u = \begin{bmatrix}0.091(0.018) \\
0.019(0.007) & 0.014(0.004)\end{bmatrix}
\]

\[
\begin{bmatrix}e_{0ij} \\
e_{1ij}
\end{bmatrix} \sim N(0, \Omega_e) : \Omega_e = \begin{bmatrix}0.553(0.015) \\
-0.015(0.006) & 0.001(0.009)\end{bmatrix}
\]

\[-2\text{loglikelihood(IGLS Deviance)} = 9311.569(4059 of 4059 cases in use)\]
R2MLwiN
R> data("bang1")
R> F5 = logit(use, cons) ~ 1 + age +
+    lc + urban + (1 + urban | district)
R> (binomialMCMC <- runMLwiN(Formula = F5,
+    D = "Binomial", data = bang1,
+    estoptions = list(EstM = 1)))
R> data("bang1")
R> F5 = logit(use, cons) ~ 1 + age +
+ lc + urban + (1 + urban | district)
R> (binomialMCMC <- runMLwiN(Formula = F5,
+ D = "Binomial", data = bang1,
+ estoptions = list(EstM = 1)))
**R2MLwiN**

MLwiN (version: 2.32) multilevel model (Binomial)

N min mean max
district 60 2 32.2333 118

Estimation algorithm: MCMC Elapsed time : 11.33s
Number of obs: 1934 (from total 1934) Number of iter.: 5000 Burn-in: 500
Bayesian Deviance Information Criterion (DIC)
Dbar D(thetabar) pD DIC
2329.716 2272.662 56.054 2384.769

The model formula:
logit(use, cons) ~ 1 + age + lc + urban + (1 + urban | district)
Level 2: district Level 1: llid

The fixed part estimates:

| Coef.  | Std. Err. | z     | Pr(>|z|) | [95% Cred. Interval] | ESS |
|--------|-----------|-------|---------|----------------------|-----|
| Intercept | -1.71218  | 0.16771 | -10.21  | 1.828e-24 *** | -2.07668 | -1.39602 | 68 |
| age     | -0.02655  | 0.00821 | -3.24   | 0.001215 **   | -0.04252 | -0.01035 | 219 |
| lOne_child | 1.13062  | 0.16236 | 6.96    | 3.327e-12 ***| 0.80553 | 1.44866 | 155 |
| lTwo_children | 1.36400  | 0.18341 | 7.44    | 1.037e-13 ***| 0.99801 | 1.70792 | 144 |
| lThree_plus | 1.36138  | 0.19376 | 7.03    | 2.134e-12 ***| 0.99136 | 1.74588 | 91 |
| urbanUrban | 0.80805  | 0.18466 | 4.38    | 1.211e-05 ***| 0.43792 | 1.17175 | 100 |

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

The random part estimates at the district level:

<table>
<thead>
<tr>
<th>Coef.</th>
<th>Std. Err.</th>
<th>[95% Cred. Interval]</th>
<th>ESS</th>
</tr>
</thead>
<tbody>
<tr>
<td>var_Intercept</td>
<td>0.42317</td>
<td>0.13547</td>
<td>0.21589</td>
</tr>
<tr>
<td>cov_Intercept urbanUrban</td>
<td>-0.43507</td>
<td>0.17899</td>
<td>-0.86104</td>
</tr>
<tr>
<td>var_urbanUrban</td>
<td>0.72068</td>
<td>0.33219</td>
<td>0.27022</td>
</tr>
</tbody>
</table>

The random part estimates at the llid level:

<table>
<thead>
<tr>
<th>Coef.</th>
<th>Std. Err.</th>
<th>[95% Cred. Interval]</th>
<th>ESS</th>
</tr>
</thead>
<tbody>
<tr>
<td>var_bcons_1</td>
<td>1.00000</td>
<td>0.00000</td>
<td>1.00000</td>
</tr>
</tbody>
</table>
R> data("bang1")
R> F5 = logit(use, cons) ~ 1 + age +
+    lc + urban + (1 + urban | district)
R> (binomialMCMC <- runMLwiN(Formula = F5,
+    D = "Binomial", data = bang1,
+    estoptions = list(EstM = 1)))
```r
R> data("bang1")
R> F5 = logit(use, cons) ~ 1 + age +
+ lc + urban + (1 + urban | district)
R> (binomialMCMC <- runMLwiN(Formula = F5,
+ D = "Binomial", data = bang1,
+ estoptions = list(EstM = 1)))
R> print(binomialMCMC, z.ratio = FALSE)
```
MLwiN (version: 2.32) multilevel model (Binomial)

N  min  mean  max
district  60  2  32.23333  118

Estimation algorithm: MCMC  Elapsed time: 11.33s
Number of obs: 1934 (from total 1934)  Number of iter.: 5000  Burn-in: 500
Bayesian Deviance Information Criterion (DIC)
Dbar  D(thetabar)  pD  DIC
2328.716  2272.662  56.054  2384.769

The model formula:
logit(use, cons) ~ 1 + age + lc + urban + (1 + urban | district)
Level 2: district  Level 1: llid

The fixed part estimates:

<table>
<thead>
<tr>
<th>Coef.</th>
<th>Std. Err.</th>
<th>pMCMC(1-sided)</th>
<th>[95% Cred. Interval]</th>
<th>ESS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intercept</td>
<td>-1.71218</td>
<td>0.16771</td>
<td>0</td>
<td>-2.07668</td>
</tr>
<tr>
<td>age</td>
<td>-0.02655</td>
<td>0.00821</td>
<td>0.0004</td>
<td>-0.04252</td>
</tr>
<tr>
<td>lcOne_child</td>
<td>1.13062</td>
<td>0.16236</td>
<td>0</td>
<td>0.80553</td>
</tr>
<tr>
<td>lcTwo_children</td>
<td>1.36400</td>
<td>0.18341</td>
<td>0</td>
<td>0.99801</td>
</tr>
<tr>
<td>lcThree_plus</td>
<td>1.36138</td>
<td>0.19376</td>
<td>0</td>
<td>0.99136</td>
</tr>
<tr>
<td>urbanUrban</td>
<td>0.80805</td>
<td>0.18466</td>
<td>0</td>
<td>0.43792</td>
</tr>
</tbody>
</table>

The random part estimates at the district level:

<table>
<thead>
<tr>
<th>Coef.</th>
<th>Std. Err.</th>
<th>[95% Cred. Interval]</th>
<th>ESS</th>
</tr>
</thead>
<tbody>
<tr>
<td>var_Intercept</td>
<td>0.42317</td>
<td>0.13547</td>
<td>0.21589</td>
</tr>
<tr>
<td>cov_Intercept_urbanUrban</td>
<td>-0.43507</td>
<td>0.17899</td>
<td>-0.86104</td>
</tr>
<tr>
<td>var_urbanUrban</td>
<td>0.72068</td>
<td>0.33219</td>
<td>0.27022</td>
</tr>
</tbody>
</table>

The random part estimates at the llid level:

<table>
<thead>
<tr>
<th>Coef.</th>
<th>Std. Err.</th>
<th>[95% Cred. Interval]</th>
<th>ESS</th>
</tr>
</thead>
<tbody>
<tr>
<td>var_bcons_1</td>
<td>1.00000</td>
<td>0.00000</td>
<td>1.00000</td>
</tr>
</tbody>
</table>
R2MLwiN

MLwiN (version: 2.32) multilevel model (Binomial)

<table>
<thead>
<tr>
<th></th>
<th>N</th>
<th>min</th>
<th>mean</th>
<th>max</th>
</tr>
</thead>
<tbody>
<tr>
<td>district</td>
<td>60</td>
<td>2</td>
<td>32.2333</td>
<td>118</td>
</tr>
</tbody>
</table>

Estimation algorithm: MCMC

Elapsed time : 11.33s

Number of obs: 1934 (from total 1934)  Number of iter.: 5000  Burn-in: 500

Bayesian Deviance Information Criterion (DIC)

<table>
<thead>
<tr>
<th></th>
<th>Dbar</th>
<th>D(theta_bar)</th>
<th>pD</th>
<th>DIC</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2328.716</td>
<td>2272.662</td>
<td>56.054</td>
<td>2384.769</td>
</tr>
</tbody>
</table>

The model formula:

logit(use, cons) ~ 1 + age + l0 + urban + (1 + urban | district)

Level 2: district  Level 1: llid

The fixed part estimates:

<table>
<thead>
<tr>
<th></th>
<th>Coef.</th>
<th>Std. Err.</th>
<th>pMCMC(1-sided)</th>
<th>[95% Cred. Interval]</th>
<th>ESS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intercept</td>
<td>-1.71218</td>
<td>0.16771</td>
<td>0</td>
<td>-2.07668 -1.39602</td>
<td>68</td>
</tr>
<tr>
<td>age</td>
<td>-0.02655</td>
<td>0.00821</td>
<td>0.0004</td>
<td>-0.04252 -0.01035</td>
<td>219</td>
</tr>
<tr>
<td>l0One_child</td>
<td>1.13062</td>
<td>0.16236</td>
<td>0</td>
<td>0.80553 1.44866</td>
<td>155</td>
</tr>
<tr>
<td>l0Two_Children</td>
<td>1.36400</td>
<td>0.18341</td>
<td>0</td>
<td>0.99801 1.70792</td>
<td>144</td>
</tr>
<tr>
<td>l0Three_plus</td>
<td>1.36138</td>
<td>0.19376</td>
<td>0</td>
<td>0.99136 1.74588</td>
<td>91</td>
</tr>
<tr>
<td>urbanUrban</td>
<td>0.80805</td>
<td>0.18466</td>
<td>0</td>
<td>0.43792 1.17175</td>
<td>100</td>
</tr>
</tbody>
</table>

The random part estimates at the district level:

<table>
<thead>
<tr>
<th></th>
<th>Coef.</th>
<th>Std. Err.</th>
<th>[95% Cred. Interval]</th>
<th>ESS</th>
</tr>
</thead>
<tbody>
<tr>
<td>var_Intercept</td>
<td>0.42317</td>
<td>0.13547</td>
<td>0.21589 0.73267</td>
<td>254</td>
</tr>
<tr>
<td>cov_InterceptUrbanUrban</td>
<td>-0.43507</td>
<td>0.17899</td>
<td>-0.86104 -0.16201</td>
<td>130</td>
</tr>
<tr>
<td>var_UrbanUrban</td>
<td>0.72068</td>
<td>0.33219</td>
<td>0.27022 1.51644</td>
<td>100</td>
</tr>
</tbody>
</table>

The random part estimates at the llid level:

<table>
<thead>
<tr>
<th></th>
<th>Coef.</th>
<th>Std. Err.</th>
<th>[95% Cred. Interval]</th>
<th>ESS</th>
</tr>
</thead>
<tbody>
<tr>
<td>var_bcons_1</td>
<td>1.00000</td>
<td>0.00000</td>
<td>1.00000 1.00000</td>
<td>5000</td>
</tr>
</tbody>
</table>

---
R> data("bang1")
R> F5 = logit(use, cons) ~ 1 + age +
+    lc + urban + (1 + urban | district)
R> (binomialMCMC <- runMLwiN(Formula = F5,
+    D = "Binomial", data = bang1,
+    estoptions = list(EstM = 1)))
R> print(binomialMCMC, z.ratio = FALSE)
R> data("bang1")
R> F5 = logit(use, cons) ~ 1 + age +
+ lc + urban + (1 + urban | district)
R> (binomialMCMC <- runMLwiN(Formula = F5,
+ D = "Binomial", data = bang1,
+ estoptions = list(EstM = 1)))
R> print(binomialMCMC, z.ratio = FALSE)
R> sixway(binomialMCMC["chains"]
+ [, "FP_Intercept", drop = FALSE],
+ "beta_0")
R2MLwiN

**Summary Statistics**
param name: beta_0 posterior mean = -1.712 SD = 0.168 mode = -1.696
quantiles: 2.5% = -2.077 5% = -2.005 50% = -1.706 95% = -1.449 97.5% = -1.396
5000 actual iterations storing every 1th iteration. Effective Sample Size (ESS) = 68

**Accuracy Diagnostics**
Raftery-Lewis (quantile): Nhat = (66556.47616)
when q=(0.025,0.975), r=0.005 and s=0.95
Brooks-Draper (mean): Nhat = 2687
when k=2 sigfigs and alpha=0.05
R> data("bang1")
R> F5 = logit(use, cons) ~ 1 + age +
+ lc + urban + (1 + urban | district)
R> (binomialMCMC <- runMLwiN(Formula = F5,
+ D = "Binomial", data = bang1,
+ estoptions = list(EstM = 1)))
R> print(binomialMCMC, z.ratio = FALSE)
R> sixway(binomialMCMC["chains"]
+ [, "FP_Interceptor", drop = FALSE],
+ "beta_0")
R> data("bang1")
R> F5 = logit(use, cons) ~ 1 + age +
+    lc + urban + (1 + urban | district)
R> (binomialMCMC <- runMLwiN(Formula = F5,
+    D = "Binomial", data = bang1,
+    estoptions = list(EstM = 1)))
R> print(binomialMCMC, z.ratio = FALSE)
R> sixway(binomialMCMC["chains"]
+    [, "FP_Intercept", drop = FALSE],
+    "beta_0")
R> trajectories(binomialMCMC["chains"]
+    [, "FP_Intercept", drop = FALSE])
R> data("bang1")
R> F5 = logit(use, cons) ~ 1 + age +
+   lc + urban + (1 + urban | district)
R> (binomialMCMC <- runMLwiN(Formula = F5,
+   D = "Binomial", data = bang1,
+   estoptions = list(EstM = 1)))
R> print(binomialMCMC, z.ratio = FALSE)
R> sixway(binomialMCMC["chains"]
+   [, "FP_Intercept", drop = FALSE],
+   "beta_0")
R> trajectories(binomialMCMC["chains"]
+   [, "FP_Intercept", drop = FALSE])
R2MLwiN

R> (OrthogbinomialMCMC <- runMLwiN(
+   Formula = F5, D = "Binomial",
+   data = bang1, estoptions = list(EstM = 1,
+   mcmcOptions = list(orth = 1))))
R> trajectories(OrthogbinomialMCMC["chains"]
+   [,"FP_Intercept", drop = FALSE])
R> (OrthogbinomialMCMC <- runMLwiN(
+     Formula = F5, D = "Binomial",
+     data = bang1, estoptions = list(EstM = 1,
+     mcmcOptions = list(orth = 1))))
R> trajectories(OrthogbinomialMCMC["chains"
+     [,"FP_Intercept", drop = FALSE])}
R2MLwiN

• As well as using MLwiN’s own MCMC estimation engine, R2MLwiN can fit models in WinBUGS / OpenBUGS
R2MLwiN

• As well as using MLwiN’s own MCMC estimation engine, R2MLwiN can fit models in WinBUGS / OpenBUGS

• With the aid of the rbugs package (Yan and Prates 2013), user can employ single runMLwiN function call to:
  - obtain starting values from an IGLS run in MLwiN,
  - automatically generate necessary BUGS model code, initial values, data files, and script,
  - fit the model in BUGS
R> WinBUGS <- "C:/WinBUGS14/WinBUGS14.exe"

R> BUGSmodel <- runMLwiN(Formula = F5,
+     D = "Binomial", data = bang1,
+     estoptions = list(EstM = 1),
+     BUGO = c(version = 4, n.chains = 1,
+               seed = 1, bugs = WinBUGS,
+               OpenBugs = FALSE))
R> WinBUGS <- "C:/WinBUGS14/WinBUGS14.exe"

R> BUGSmodel <- runMLwiN(Formula = F5,
+    D = "Binomial", data = bang1,
+    estoptions = list(EstM = 1),
+    BUGO = c(version = 4, n.chains = 1,
+             seed = 1, bugs = WinBUGS,
+             OpenBugs = FALSE))
# WINBUGS 1.4 code generated from MLwiN program

#---MODEL Definition-------------

model
{
  # Level 1 definition
  for(i in 1:N) {
    use[i] ~ dbin(p[i],denom[i])
  }
  # Higher level definitions
  for (j in 1:n2) {
    u2[j,1:2] ~ dmnorm(zero2[1:2],tau.u2[1:2,1:2])
  }
  # Priors for fixed effects
  for (k in 1:6) { beta[k] ~ dflat() }
  # Priors for random terms
  for (i in 1:2) { zero2[i] <- 0 }
  tau.u2[1:2,1:2] ~ dwish(R2[1:2, 1:2],2)
  sigma2.u2[1:2,1:2] <- inverse(tau.u2[,])
}
R2MLwiN

```r
u2[60,2]  -0.96108  -0.18080  1.792e-01  0.54900  1.30202

> cc = cbind(binomialMCMC["FP"], OrthogbinomialMCMC["FP"])
> dd = cbind(head(binomialMCMC["RP"], -1),
+     head(OrthogbinomialMCMC["RP"], -1))
> ESS.binMCMC = effectiveSize(binomialMCMC["chains"], 2:10)
> ESS.orthogMCMC = effectiveSize(OrthogbinomialMCMC["chains"], 2:10)
> ESStable = round(rbind(cc, dd), 3)
> BUGSList <- c(1:6, 8, 9, 11)
> BUGS.Coeff <- round(summary(BUGSmodel)$statistics[BUGSList, 1], 3)
> BUGS.ESS <- as.data.frame(effectiveSize(BUGSmodel))
> ESStable = cbind(ESStable[, 1], round(ESS.binMCMC), ESStable[, 2],
+     round(ESS.orthogMCMC), BUGS.Coeff, round(BUGS.ESS[BUGSList, ]))
> colnames(ESStable) = c("A)Coeff.", "A)ESS", "B)Coeff.",
+     "B)ESS", "C)Coeff.", "C)ESS")
> cat("NB: A = MLwiN(non-orthog.), B = MLwiN(orthog.), C = WinBUGS\n"
NB: A = MLwiN(non-orthog.), B = MLwiN(orthog.), C = WinBUGS
> ESStable

<table>
<thead>
<tr>
<th></th>
<th>A)Coeff.</th>
<th>A)ESS</th>
<th>B)Coeff.</th>
<th>B)ESS</th>
<th>C)Coeff.</th>
<th>C)ESS</th>
</tr>
</thead>
<tbody>
<tr>
<td>FP_Intercept</td>
<td>-1.712</td>
<td>68</td>
<td>-1.723</td>
<td>188</td>
<td>-1.726</td>
<td>274</td>
</tr>
<tr>
<td>FP_age</td>
<td>-0.027</td>
<td>219</td>
<td>-0.027</td>
<td>1064</td>
<td>-0.027</td>
<td>884</td>
</tr>
<tr>
<td>FP_TcOne_child</td>
<td>1.131</td>
<td>155</td>
<td>1.141</td>
<td>937</td>
<td>1.141</td>
<td>885</td>
</tr>
<tr>
<td>FP_TcTwo_children</td>
<td>1.364</td>
<td>144</td>
<td>1.365</td>
<td>931</td>
<td>1.369</td>
<td>668</td>
</tr>
<tr>
<td>FP_TcThree_plus</td>
<td>1.361</td>
<td>91</td>
<td>1.366</td>
<td>952</td>
<td>1.368</td>
<td>416</td>
</tr>
<tr>
<td>FP_urbanUrban</td>
<td>0.808</td>
<td>100</td>
<td>0.819</td>
<td>108</td>
<td>0.819</td>
<td>445</td>
</tr>
<tr>
<td>RP2_var_Intercept</td>
<td>0.423</td>
<td>254</td>
<td>0.416</td>
<td>117</td>
<td>0.423</td>
<td>482</td>
</tr>
<tr>
<td>RP2_cov_Intercept_urbanUrban</td>
<td>-0.435</td>
<td>130</td>
<td>-0.445</td>
<td>74</td>
<td>-0.437</td>
<td>357</td>
</tr>
<tr>
<td>RP2_var_urbanUrban</td>
<td>0.721</td>
<td>100</td>
<td>0.764</td>
<td>87</td>
<td>0.735</td>
<td>282</td>
</tr>
</tbody>
</table>
```

> ```
R2MLwiN

As well as usual help files...

...R2MLwiN comes with demos as well...

...these replicate all the examples in:

– MLwiN User Manual (IGLS)

– MLwiN MCMC Manual
R2MLwiN
To list demos:

```r
R> demo(package = "R2MLwiN")
```
To list demos:

```r
R> demo(package = "R2MLwiN")
```

---

R2MLwiN

---

Demos in package `R2MLwiN`:

- **MCMCGuide01**: Introduction to MCMC Estimation and Bayesian Modelling
- **MCMCGuide02**: Single Level Normal Response Modelling
- **MCMCGuide03**: Variance Components Models
- **MCMCGuide04**: Other Features of Variance Components Models
- **MCMCGuide05**: Prior Distributions, Starting Values and Random Number Seeds
- **MCMCGuide06**: Random Slopes Regression Models
- **MCMCGuide07**: Using the WinBUGS Interface in MLwiN
- **MCMCGuide08**: Running a Simulation Study in MLwiN
- **MCMCGuide09**: Modelling Complex Variance at Level 1 / Heteroscedasticity
- **MCMCGuide10**: Modelling Binary Responses
- **MCMCGuide11**: Poisson Response Modelling
- **MCMCGuide12**: Unordered Categorical Responses
- **MCMCGuide13**: Ordered Categorical Responses
- **MCMCGuide14**: Adjusting for Measurement Errors in Predictor Variables
- **MCMCGuide15**: Cross Classified Models
- **MCMCGuide16**: Multiple Membership Models
- **MCMCGuide17**: Modelling Spatial Data
- **MCMCGuide18**: Multivariate Normal Response Models and Missing Data
- **MCMCGuide19**: Mixed Response Models and Correlated Residuals
- **MCMCGuide20**: Multilevel Factor Analysis Modelling
- **MCMCGuide21**: Using Structured MCMC
- **MCMCGuide22**: Using the Structured MVN framework for models
- **MCMCGuide23**: Using Orthogonal fixed effect vectors
- **MCMCGuide24**: Parameter expansion
- **MCMCGuide25**: Hierarchical Centring
- **UserGuide02**: Introduction to Multilevel Modelling
- **UserGuide03**: Residuals
- **UserGuide04**: Random intercept and Random Slope Models
- **UserGuide05**: Graphical Procedures for Exploring the Model
- **UserGuide06**: Contextual Effects
R2MLwiN
R2MLwiN

To list demos:

\texttt{R> demo(package = "R2MLwiN")}
To list demos:

R> demo(package = "R2MLwiN")

To run a specific demo:

R> demo(MCMCGuide03)
To list demos:

R> demo(package = "R2MLwiN")

To run a specific demo:

R> demo(MCMCGuide03)

To show demo script:

R> file.show(system.file("demo", "MCMCGuide03.R", package = "R2MLwiN"))
To list demos:
```r
R> demo(package = "R2MLwiN")
```
To run a specific demo:
```r
R> demo(MCMCGuide03)
```
To show demo script:
```r
R> file.show(system.file("demo", "MCMCGuide03.R", package = "R2MLwiN"))
```
Obtaining R2MLwiN & finding out more:
Obtaining R2MLwiN & finding out more:

R2MLwiN available on CRAN, e.g. via:

R> install.packages("R2MLwiN",
+ repos = "http://cran.r-project.org")
Obtaining R2MLwiN & finding out more:

R2MLwiN available on CRAN, e.g. via:
```
R> install.packages("R2MLwiN",
+    repos = "http://cran.r-project.org")
```

& supporting info on Centre for Multilevel Modelling website:
http://www.bristol.ac.uk/cmm/software/r2mlwin/
...includes link to log files of demo runs
Obtaining R2MLwiN & finding out more:

R2MLwiN available on CRAN, e.g. via:

```R
R> install.packages("R2MLwiN",
+ repos = "http://cran.r-project.org")
```

& supporting info on Centre for Multilevel Modelling website:

http://www.bristol.ac.uk/cmm/software/r2mlwin/

...includes link to log files of demo runs

Leaflets available at the front too...
Obtaining R2MLwiN & finding out more:

Forum on Centre for Multilevel Modelling website:

www.cmm.bristol.ac.uk/forum/viewforum.php?f=7&sid=f343182d9739d5549085f119a1543129
Obtaining R2MLwiN & finding out more:

R2MLwiN available on CRAN, e.g. via:
R> install.packages("R2MLwiN",
+    repos = "http://cran.r-project.org")

& supporting info on Centre for Multilevel Modelling website:
http://www.bristol.ac.uk/cmm/software/r2mlwin/
...includes link to log files of demo runs

Leaflets available at the front too...
Obtaining R2MLwiN & finding out more:

R2MLwiN available on CRAN, e.g. via:

R> install.packages("R2MLwiN",
+    repos = "http://cran.r-project.org")

& supporting info on Centre for Multilevel Modelling website:
http://www.bristol.ac.uk/cmm/software/r2mlwin/
...includes link to log files of demo runs

Leaflets available at the front too...