Statistics/Data Analysis

help runmlwin

Title

runmlwin - Run the MLwiN multilevel modelling software from within Stata

Syntax

```
        runmlwin responses_and_fixed_part, random_part [discrete(discrete_options)]
                  [mcmc(mcmc_options)] [general_options]

    where the syntax of responses_and_fixed_part is one of the following

        for univariate continuous, binary, proportion and count response models

            depvar indepvars [if] [in]

        for univariate ordered and unordered categorical response models

            depvar indepvars1 [(indepvars2, contrast(numlist)) ... ] [if] [in]

            where indepvars1 are those independent variables which appear with
            separate coefficients in each of every log-odds contrast, while
            indepvars2 are those independent variables which appear with common
            coefficients for those log-odds contrasts specified in
            contrast(numlist).  Contrasts can be thought of as the separate
            "subequations" or "arms" of a multinomial response model.  These
            contrasts are indexed 1,2,... up to the total number of contrasts
            included in the model.  The total number of contrasts will be one less
            than the number of response categories.

        for multivariate response models

            (depvar1 indepvars1, equation(numlist))
                        (depvar2 indepvars2, equation(numlist))
                        [(depvar3 indepvars3, equation(numlist))]
                        [ ... ]
                        [if] [in]

            where equation(numlist) specifies equation numbers.  Equation numbers
            are indexed 1,2,... up to the total number of equations (i.e. response
            variables) included in the model.

    and the syntax of random_part is

        [ ... ] [level2(levelvar: [varlist] [, random_part_options])]
                  level1(levelvar: [varlist] [, random_part_options])

    where levelvar is a variable identifying the groups or clusters for the random
    effects at each level.  varlist is the list of variables with random
    coefficients at each level.

    random_part_options - (R)IGLS and MCMC  Description
    ---------------------------------------------------------------------------
    Model - (R)IGLS and MCMC
      diagonal                             set all covariances to zero

    Model - (R)IGLS only
      elements(matrix)                     set specific (co)variances to zero
      weightvar(varname)                   specify variable containing sampling
                                             weights
```

Model - MCMC only	
mmids(*varlist*)	specify variables containing multiple membership unit identifiers
mmweights(*varlist*)	specify variables containing multiple membership weights
carids(*varlist*)	specify variables containing conditional autoregressive (CAR) unit identifiers
carweights(*varlist*)	specify variables containing conditional autoregressive (CAR) weights
flinits(*matrix*)	specify initial values for factor loadings
flconstraints(*matrix*)	constrain specific factor loadings to their initial values
fvinits(*matrix*)	specify initial values for factor (co)variances
fvconstraints(*matrix*)	constrain specific factor (co)variances to their initial values
fscores(stub*)	where stub* is the variable stub for factor scores and their standard errors at this level
parexpansion	parameter expansion
Estimation - (R)IGLS only	
reset(*resetname*)	reset (co)variances to zero when variances turn negative
Post-estimation - (R)IGLS and MCMC	
residuals(stub*, [*residuals options*])	where stub* is the variable stub for residuals and their standard errors at this level

discrete_options - (R)IGLS and MCMC	Description
Model((R)IGLS and MCMC)	
distribution(*distname*)	distribution(s) of *depvar(s)*
link(*linkname*)	link function
denominator(*varlist*)	denominator(s) of *depvar(s)*
extra	extra binomial variation
basecategory(#)	set the value of the response to be used as the base or reference category (multinomial distributions only)
offset(*varname*)	include *varname* in the model with coefficient constrained to 1 (Poisson or negative binomial distributions only)
proportion(*varname*)	specify variable containing multinomial proportions
Estimation - (R)IGLS only	
mql1	fit model using first order marginal quasi-likelihood linearization, the default
mql2	fit model using second order marginal quasi-likelihood linearization
pql1	fit model using first order penalised quasi-likelihood linearization
pql2	fit model using second order penalised quasi-likelihood linearization

mcmc_options - MCMC only	Description
Model - MCMC only	
logformulation	specify log formulation for level 1 variance
cc	specify cross-classified model
msubscripts	use multiple subscript notation
corresiduals(*restype***)**	specify the correlation structure of the level 1 random effects (i.e. the residual errors)
me(*varlist***, variances(***numlist***))**	specify subset of independent variables with measurement error
priormatrix(*matrix***)**	specify informative priors for fixed and random part parameters
rppriors(*rppriors spec***)**	specify prior distributions for the random part priors, the default is gamma(0.001,0.001)
savewinbugs(*savewinbugs options***)**	save specified model as a WinBUGS model
Estimation - MCMC only	
on	fit model using default MCMC options
burnin(*#***)**	specify number of iterations for the burn-in period, default is 500
chain(*#***)**	specify number of iterations for the monitoring chain period, default is 5000
thinning(*#***)**	store every # iteration, default is 1
refresh(*#***)**	refresh the MLwiN equation window every # iterations, default is 50
scale(*#***)**	set scale factor, default is 5.8
noadaptation	do not use adaptation
acceptance(*#***)**	set Metropolis Hastings acceptance rate, default is 0.5
tolerance(*#***)**	set the tolerance, default is 0.1
cycles(*#***)**	number of cycles, default is 1
femethod(*mcmc method***)**	specify fixed effects method, default depends on specified model
remethod(*mcmc method***)**	specify random effects method, default depends on specified model
levelonevarmethod(*mcmc method***)**	specify level 1 variance method, default depends on specified model
higherlevelsvarmethod(*mcmc method***)**	specify higher level variances method, default depends on specified model
smcmc	use structured MCMC methods
smvn	use structured multivariate normal (MVN) framework
orthogonal	use orthogonal parameterisation
hcentring(*#***)**	use hierarchical centring at level #
seed(*#***)**	set MCMC random number seed, default is 1
Post estimation - MCMC only	
savechains(*filename*[**, replace**]**)**	save MCMC parameter estimates for each iteration in filename.dta
imputeiterations(*numlist***)**	impute missing values at specified iterations
imputesummaries	for each missing value, calculate the mean and the standard deviation of the chain for that missing value

general_options - (R)IGLS and MCMC	Description
Model - (R)IGLS only	
weights(*weights options***)**	apply sampling weights options
constraints(*numlist***)**	apply specified linear constraints

Estimation - (R)IGLS only	
igls	fit model via iterative generalised least squares (equivalent to maximum likelihood), the default
rigls	fit model via restrictive iterative generalised least squares (equivalent to maximum restricted likelihood)
maxiterations(*#***)**	specifies the maximum number of (R)IGLS iterations, default is 20
tolerance(*#***)**	IGLS convergence tolerance, default is 2 (as in 10e-2)
seed(*#***)**	set IGLS random number seed, default is **seed(***1***)**
Estimation - (R)IGLS and MCMC	
initsprevious	use parameter estimates from previous model as initial values
initsmodel(*name***)**	use parameter estimates from specified model as initial values
initsb(*matrix***)**	initial parameter values vector
initsv(*matrix***)**	initial sampling (co)variance values matrix
SE/Robust - (R)IGLS only	
fpsandwich	sandwich estimates for fixed part standard errors
rpsandwich	sandwich estimates for random part standard errors
Reporting - (R)IGLS and MCMC	
level(*#***)**	set confidence level; default is level(95)
or	report fixed-effects coefficients as odds ratios
irr	report fixed-effects coefficients as incidence-rate ratios
rrr	report fixed-effects coefficients as relative-rate ratios
sd	show random-effects variance as standard deviations
correlations	show random-effects covariance as correlations
noheader	suppress output header
nogroup	suppress table summarizing groups
nocontrast	suppress table summarizing contrasts
nofetable	suppress fixed-effects table
noretable	suppress random-effects table
Reporting - MCMC only	
nodiagnostics	do not calculate MCMC diagnostics
mode	report parameter estimates as chain modes
median	report parameter estimates as chain medians
zratio	report classical z-ratios and p-values
Post-estimation - (R)IGLS and MCMC	
simulate(*newvar***)**	simulates a new response variable *newvar* based on the estimated model parameters
Export - (R)IGLS and MCMC	
viewmacro	view the MLwiN macro for the fitted model
savemacro(*filename*[**, replace**]**)**	save the MLwiN macro for the fitted model
saveworksheet(*filename*[**, replace**]**)**	save the MLwiN worksheet for the fitted model

Other - (R)IGLS and MCMC	
forcesort	forces the data to be sorted according to the model hierarchy
nosort	prevents checking that the data are sorted according to the model hierarchy
forcerecast	forces a recast of all variables saved as long or double to float
nodrop	prevents variables that do not appear in the model from being dropped prior to sending the data to MLwiN
nomlwin	prevent MLwiN from being run
mlwinpath(*string***)**	mlwin.exe file address, including the file name
mlwinscriptpath(*string***)**	mlnscript.exe file address, including the file name; advanced use only
mlwinsettings(*mlwin_settings***)**	manually override MLwiN settings
nopause	suppresses the two pause steps in MLwiN
batch	prevents any MLwiN GUI windows being displayed; advanced use only

resetname - (R)IGLS only	Description
all	reset all (co)variances to zero
variances	reset only the variances to zero
none	reset no (co)variances

weights_options - (R)IGLS only	Description
nostandardised	no standardisation of the level specific weight variables
nofpsandwich	no sandwich estimates for fixed part standard errors
norpsandwich	no sandwich estimates for random part standard errors

residuals_options - (R)IGLS and MCMC	Description
variances	posterior variances
standardised	standardised residuals
leverage	leverage residuals
influence	influence residuals
deletion	deletion residuals
sampling	sampling variance-covariance matrix
norecode	do not recode residuals exceedingly close or equal to zero to missing
reflated	unshrunken residuals

residuals_options - MCMC only	Description
savechains(*filename*[**, replace**]**)**	save MCMC residual estimates for each iteration in filename.dta

distname - (R)IGLS and MCMC	Description
normal	normal distribution
binomial	binomial distribution
multinomial	multinomial distribution
poisson	Poisson distribution
nbinomial	negative binomial distribution

linkname - (R)IGLS and MCMC	Description
logit	logit link function
probit	probit link function
cloglog	complementary log-log link function
log	log link function
ologit	ordered logit link function
oprobit	ordered probit link function
ocloglog	ordered complementary log-log link function
mlogit	unordered logit link function

restype - MCMC only	Description
unstructured	unstructured (co)variance matrix
exchangeable	structured errors with a common correlation parameter and a common variance parameter
areqvars	AR1 errors with a common variance parameter
eqcorrsindepvars	structured errors with a common correlation parameter and independent variance parameters
arindepvars	AR1 errors with independent variance parameters

rppriors_spec - MCMC only	Description
uniform	use the uniform prior distribution
gamma(*a b***)**	use the gamma prior distribution with shape a and scale b, the default is gamma(0.001,0.001)

mcmc_method - MCMC only	Description
gibbs	Gibbs algorithm
univariatemh	univariate Metropolis Hastings algorithm
multivariatemh	multivariate Metropolis Hastings algorithm

savewinbugs_options - MCMC only	Description
model(*filename*[**, replace**]**)**	save model as a WinBUGS model in filename.txt
inits(*filename*[**, replace**]**)**	save initial values in WinBUGS format in filename.txt
data(*filename*[**, replace**]**)**	save data in WinBUGS format in filename.txt
nofit	do not fit the model in MLwiN

mlwin_settings - (R)IGLS and MCMC	Description
size(*#***)**	specify maximum worksheet size allowed in MLwiN
levels(*#***)**	specify maximum number of levels allowed in MLwiN
columns(*#***)**	specify maximum number of data columns allowed in MLwiN
variables(*#***)**	specify maximum number of modelled variables allowed in MLwiN
tempmat	use memory allocated to the worksheet to store temporary matrices in MLwiN
optimat	limit the maximum matrix size allocated in MLwiN

Description

runmlwin allows Stata users to run the powerful MLwiN multilevel modelling software from within Stata.

See *Remarks on alternative Stata commands for fitting multilevel models* below for more information.

MLwiN has the following features:

(1) Estimation of multilevel models for continuous, binary, count, ordered categorical and unordered categorical data;

(2) Fast estimation via classical and Bayesian methods;

(3) Estimation of multilevel models for cross-classified and multiple membership non-hierarchical data structures;

(4) Estimation of multilevel multivariate response models, multilevel spatial models, multilevel measurement error models and multilevel multiple imputation models.

MLwiN is required to use **runmlwin**. See *Remarks on **runmlwin** installation instructions* below for more information.

A comprehensive range of **runmlwin** examples and a user forum are available at:

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

Options

Model

(a) random part model options

All options reported in this sub-section are specific to the level at which they are specified.

diagonal specifies a diagonal matrix for the random effects variance-covariance matrix. This implies zero correlations between the random effects at that level. Note that should a Toeplitz or other banded structures be desired (for example when modelling longitudinal data), these can be implemented using the **constraints** options.

elements(*matrix***)** sets specific (co)variances to zero. Note that the matrix must be a row vector with one column for each (co)variance where the lower diagonal elements of the variance covariance matrix have been vectorised row by row. If there are q random effects terms (variables with random coefficients), the unstructured covariance matrix has $q(q+1)/2$ unique parameters. The elements of this vector must be 0 or 1 where 0 sets the relevant (co)variance to zero and 1 specifies the parameter to be freely estimated. See *Examples* for an example of this option.

weightvar(*varname***)** specifies the variable containing the sampling weights.

mmids(*varlist***)** specifies the variables containing the multiple membership unit identifiers. The number of variables should equal the maximum number of higher level units that any lower level unit belongs to. The order in which the multiple membership unit identifier variables is specified is irrelevant other than it must correspond to the order in which the associated multiple membership weight variables is specified. Zero values rather than missing values must be assigned for redundant identifier variables.

mmweights(*varlist***)** specifies the variables containing the multiple membership weights. The number of variables should equal the maximum number of higher level units that any lower level unit belongs to. The ordering of the variables must correspond to the ordering of the multiple membership unit identifier variables. Zero values rather than missing values must be assigned for redundant weight variables.

carids(*varlist***)** specifies the variables containing the conditional autoregressive (CAR) unit identifiers. The number of variables should equal the maximum number of higher level units that any lower level unit belongs to. The order in which the CAR unit identifier variables is specified is irrelevant other than it must correspond to the order in which the associated CAR weight variables is specified. Zero values rather than missing values must be assigned for redundant identifier variables.

carweights(*varlist***)** specifies the variables containing the conditional autoregressive (CAR) weights. The number of variables should equal the maximum number of higher level units that any lower level unit belongs to. The ordering of the variables must correspond to the ordering of the CAR unit identifier variables. Zero values rather than missing values must be assigned for redundant weight variables.

flinits(*matrix***)** specifies initial values for factor loadings. The number of rows must equal the number of responses. The number of columns must equal the number of factors.

flconstraints(*matrix***)** constrain specific factor loadings to their initial values. The number of rows must equal the number of responses. The number of columns must equal the number of factors. The elements of this matrix must be 0 or 1 where 0 freely estimates the factor loading and 1 constrains the factor loading to its initial value.

fvinits(*matrix***)** specifies initial values for factor (co)variances. The number of rows and the number of columns must equal the number of factors. The diagonal elements correspond to the factor variances. The off-diagonal elements correspond to the factor covariances. Initial values for the factor variances must be positive. Intial values for the covariances must correspond to correlations that lie between -1 and +1.

fvconstraints(*matrix***)** constrain specific factor (co)variances to their initial values. The number of rows and the number of columns must equal the number of factors. The elements of this matrix must be 0 or 1 where 0 freely estimates the (co)variance and 1 constrains the (co)variance to its initial value.

fscores(*stub****)** calculates posterior estimates of the factor scores and their associated standard errors for all factors specified at the given level. runmlwin will name the factors. For example if there are three factors in the model, runmlwin would name the factors *stub1*, *stub2*, *stub3* and would name their associated standard errors as *stub1se*, *stub2se*, *stub3se*.

parexpansion uses parameter expansion. Note that parameter expansion cannot be specified at level 1.

(b) discrete response model options

distribution(*distname***)** specifies the distribution(s) of the *depvar(s)*. **normal** is applicable for continuous response variables, **binomial** for binary and proportion response variables, **multinomial** for ordinal and nominal categorical response variables and **poisson** and **nbinomial** (negative binomial) are applicable for count responses. In multivariate response models where one or more response variables are discrete, one distribution must be given for each response variable.

link(*linkname***)** specifies the link function. **logit**, **probit** and **cloglog** (complementary log-log) are applicable when the **binomial** distribution has been specified, **log** is applicable when either the **poisson** or **nbinomial** distributions have been specified, and **ologit** (ordered logit), **oprobit** (ordered probit), **ocloglog** (ordered complementary log-log) and **mlogit** (unordered logit) are applicable when the **multinomial** distribution has been specified. Note that it is not possible to specify multivariate response models which have a mixture of different types of discrete response. Thus, only one link function can be specified for the discrete responses. In multivariate response models where a mixture of continuous and discrete response variables are specified, only the link function for the discrete responses is specified.

denominator(*varlist***)** specifies the variables containing the **binomial** denominator(s) (i.e. the number of binomial trials) for the response variable(s). This option is only applicable when modelling proportions (i.e. counts with known totals). For example, to model school level data on the proportion of students who pass an exam, the data should have one row per school, the *depvar* is the proportion of students who passed the exam and the **denominator** is the number of students who took the exam. Note that this implementation differs from that for **[XT] xtmelogit** where one would specify a numerator (the number of children who passed the exam) as the response variable rather than the proportion.

extra specifies an extra binomial variation parameter to allow for over- or under-dispersion.

basecategory(*#***)** specifies the value of the response to be used as the base or reference category. This option is only applicable when modelling ordinal or nominal categorical response variables using the **multinomial** distribution. When modelling ordinal responses, the basecategory must be either the first or last response category.

offset(*varname***)** includes *varname* in the fixed part of the model with coefficient constrained to one. This option is only applicable when modelling count responses using the **poisson** or **nbinomial** distributions.

proportion(*varname***)** specifies the variable containing multinomial proportions. For example, to model neighbourhood level data on the proportion of individuals in good, average and poor health, the data should have three rows per neighbourhood. The first row should give the proportion of individuals in good health. The second row should give the proportion of individuals in average health. The third row should give the proportion of individuals in poor health. The total number of individuals in each neighbourhood is then specified with the **denominator** option.

(c) MCMC model options

logformulation specifies a log formulation model for the level 1 variance.

cc specifies that the model is a non-hierarchical cross-classified model rather than a hierarchal model. In cross-classified models, the levels are often referred to as classifications.

msubscripts uses multiple subscript notation.

corresiduals(*restype***)** specifies the correlation structure of the level 1 random effects (i.e. the residual errors).

unstructured, the default, is the most general structure; it estimates distinct variances for each residual error and distinct covariances for each residual error pair.

exchangeable fits structured errors with a common correlation parameter and a common variance parameter.

ar1eqvars fits AR1 errors with a common variance parameter.

eqcorrsindepvars fits structured errors with a common correlation parameter and independent variance parameters.

ar1indepvars fits AR1 errors with independent variance parameters.

me(*varlist*, **variances(***numlist***))** specifies the subset of independent variables with measurement error. The corresponding measurement error variances are specified in **variances(***numlist***)**.

priormatrix(*matrix***)** specifies informative priors for the fixed and random part parameters.

rppriors(*rppriors_spec***)** specifies prior distributions for the random part priors. **uniform** specifies the uniform distribution. **gamma(***a*, *b***)** specifies the gamma distribution. The default *rppriors_spec* is **gamma(0.001,0.001)**.

savewinbugs(*savewinbugs_options***)** saves the specified model as a WinBUGS model. The model can then be fitted in WinBUGS using the **winbugs** suite of commands if these are installed.

savewinbugs_options are

model(*filename* [, *replace*]**)** saves the model as a WinBUGS model in filename.txt

inits(*filename* [, *replace*]**)** saves the initial values in WinBUGS format in filename.txt

data(*filename* [, *replace*]**)** saves the data in WinBUGS format in filename.txt

nofit do not fit the model in MLwiN

(d) general model options

weights(*weights_options***)** specifies overall options for any sampling weights variables included in the model. This option can only be specified if the user has specified the **weightvar** option at one or more levels.

The **nostandardisation** option specifies that the level specific weight variables should not be standardised

The **nofpsandwich** option specifies that sandwich estimates should not be used for the standard errors of the fixed part parameter estimates at any level

The **norpsandwich** option specifies that sandwich estimates should not be used for the standard errors of the random part parameter estimates at any level

See *Remarks on using sampling weights* below for more information.

constraints(*clist***)** specifies the constraint numbers for the constraints to be applied to the model. Constraints are specified using the **[R] constraint** command. Only linear constraints can be specified. See *Examples* for an example of this option.

Estimation

(a) random part estimation options

All options reported in this sub-section are specific to the level at which they are specified.

reset(*resetname***)** specifies the action to be taken when during estimation a variance parameter at a particular iteration is estimated to be negative. **all** resets a negative variance to zero along with any associated covariances. **variances** resets a negative variance to zero but not the associated covariances. **none** ignores negative variances; no parameters are reset to zero.

(b) discrete response estimation options

mql1, the default, specifies that the model be fitted using a first order marginal quasi-likelihood linearization. See _Remarks on quasilikelihood estimates: MQL1, MQL2, PQL1 and PQL2_ below for more information.

mql2 specifies that the model be fitted using a second order marginal quasi-likelihood linearization.

pql1 specifies that the model be fitted using a first order penalised quasi-likelihood linearization.

pql2 specifies that the model be fitted using a second order penalised quasi-likelihood linearization.

(c) MCMC estimation options

on fits the specified model using default MCMC options.

burnin(_#_**)** specifies the number of iterations for the burn-in period. The default is **burnin(500)**. This option specifies the number of iterations necessary for the MCMC to reach approximate stationary or, equivalently, to converge to a stationary distribution. The required length of the burn-in period will depend on the initial values.

chain(_#_**)** specifies the number of iterations for the monitoring chain period. The default **is chain(5000)**. This is the number of iterations, after the burn-in period, for which the chain is to be run. Distributional summaries for the parameters are based on these iterations. Parameter estimates are given by the means of these chains, while the standard errors are given by the standard deviation of these chains.

thinning(_#_**)** stores every #-th iteration of the monitoring chains. The default is **thinning(1)**. Parameter means and standard deviations are based on the non-thinned monitoring chains. All other MCMC summary statistics (e.g. ESS and 95% credible intervals) are based on the thinned monitoring chain. The **or**, **irr**, **rrr**, **sd**, **correlation**, **mode** and **median** options also all only apply to the thinned monitoring chains. For example, fitting a model for a monitoring chain length of 50,000 and setting thinning to 10 will result in 5,000 iterations being stored. The parameter means and standard deviations will then be based on all 50,000 iterations, while the ESS's and 95% credible intervals will be based on the 5,000 stored iterations.

refresh(_#_**)** refreshes the MLwiN equation window every # stored iterations. The default is **refresh(50)**. When the **nopause** option is not used, refreshing the MLwiN equation window less frequently can speed up estimation time.

scale(_#_**)** sets the scale factor. The default is **scale(5.8)**.

noadaptation prevents adaptation from being used.

acceptance(_#_**)** sets the Metropolis Hastings acceptance rate. The default is **acceptance(0.5)** and permissible values range from zero to one.

tolerance(_#_**)** sets the tolerance rate used for adapting. The default is **tolerance(0.1)** and permissible values range from zero to one.

cycles(_#_**)** sets the number of Metropolis Hastings cycles per MCMC iteration. The default is **cycles(1)**. The higher # is, the more likely a new proposed parameter value will be accepted on each iteration.

femethod(_mcmc_method_**)** specifies the MCMC method for estimating the fixed effects. The default depends on the specified model.

gibbs uses the Gibbs algorithm and is the default for continuous response models.

univariatemh uses the univariate Metropolis Hastings algorithm and is the default for univariate discrete response models.

multivariatemh uses the multivariate Metropolis Hastings algorithm and is the default for multivariate response models with at least one discrete response.

remethod(*mcmc_method***)** specifies the MCMC method for estimating the random effects. The default depends on the specified model. *mcmc_method* is defined under the **femethod(***mcmc_method***)** option.

levelonevarmethod(*mcmc_method***)** specifies the MCMC method for estimating the level 1 variance. The default depends on the specified model. *mcmc_method* is defined under the **femethod(***mcmc_method***)** option.

higherlevelsvarmethod(*mcmc_method***)** specifies the MCMC method for estimating the level 2 and higher variances. The default depends on the specified model. *mcmc_method* is defined under the **femethod(***mcmc_method***)** option.

smcmc uses structured MCMC methods.

smvn uses the structured multivariate normal (MVN) framework.

orthogonal uses orthogonal parameterisation. Note that Browne (2012) recommends to always specify this option for discrete response models. There is little advantage to specifying this option for continuous response models as in these models the fixed effects are generally blocked updated.

hcentring(*#***)** uses hierarchical centring at level #.

seed(*#***)** specifies the initial value of the MCMC random number seed. The default is **seed(1)**. *#* should be a positive integer. Note that there are two random number seeds in MLwiN and therefore two seed options in **runmlwin**: The IGLS random number seed and the MCMC random number seed. In contrast to Stata commands, both seeds in MLwiN have default initial values. That is, these seeds are the same every time you call **runmlwin**. The only time you will therefore need to set the IGLS or M seeds is if you wish to replicate MLwiN analyses carried out when using MLwiN in the traditional point-and-click way.

(d) General estimation options

igls, the default, specifies that the model be fitted using iterative generalised least squares (equivalent to maximum likelihood). See *Remarks on IGLS vs. RIGLS* below for more information.

rigls specifies that the model be fitted using restrictive iterative generalised least squares (equivalent to maximum restricted likelihood, also referred to as residual maximum likelihood).

maxiterations(*#***)** specifies the maximum number of (R)IGLS iterations. The default is **maxiterations(20)**.

tolerance(*#***)** specifies the convergence tolerance for the IGLS algorithm. The default is **tolerance(***2***)** as in a tolerance of 10e-2. IGLS iterations will be halted once every parameter changes by a relative amount less than *#*.

seed(*#***)** specifies the initial value of the IGLS random number seed. The default is **seed(***1***)**. This option allows you to replicate your results if you use the **simulate(***newvar***)** option. *#* should be a positive integer. Note that there are two random number seeds in MLwiN and therefore two seed options in **runmlwin**: The IGLS random number seed and the MCMC random number seed. In contrast to Stata commands, both seeds in MLwiN have default initial values. That is, these seeds are the same every time you call **runmlwin**. The only time you will therefore need to set the IGLS or M seeds is if you wish to replicate MLwiN analyses carried out when using MLwiN in the traditional point-and-click way.

initsprevious specifies that the parameter estimates from the previous model are used as the initial values. This option is used: (1) when building a series of increasingly complex models using IGLS; (2) when moving from MQL1 estimation to PQL2 estimation; (3) to specify initial values for MCMC estimation. When the current model contains new parameters not specified in the previous model, new fixed part parameters are set to zero, random part variance parameters are set to one and random part covariances are set to zero.

Note that when this option is used to specify initial values for fitting the model using the Metropolis Hastings algorithm in MCMC (i.e. for discrete response models), this option will also feed the parameter sampling variance-covariance values into the algorithm.

initsmodel(*name***)** specifies that the parameter estimates from the model results saved under *name* are used as the initial values. See comments for **initsprevious**.

initsb(*matrix***)** specifies the parameter initial values vector. See comments for **initsprevious**.

initsv(*matrix***)** specifies the parameter initial sampling variance-covariance values matrix. Note this option is only relevant when fitting the model using the Metropolis Hastings algorithm in MCMC (i.e. for discrete response models). See comments for **initsprevious**.

SE/Robust

fpsandwich specifies the robust or sandwich estimates for the fixed part standard errors.

rpsandwich specifies the robust or sandwich estimates for the random part standard errors.

Reporting

level(*#***)** set confidence level (credible level if using MCMC); default is **level(95)**.

or reports the fixed-effects coefficients transformed to odds ratios, i.e., exp(b) rather than b. Standard errors and confidence intervals are similarly transformed. This option affects how results are displayed, not how they are estimated. **or** may only be specified when modelling binary responses when using the **binomial** distribution and logit link function or when modelling ordinal categorical responses using the **multinomial** distribution. **or** may be specified at estimation or when replaying previously estimated results.

irr reports the fixed-effects coefficients transformed to incidence-rate ratios, i.e., exp(b) rather than b. Standard errors and confidence intervals are similarly transformed. This option affects how results are displayed, not how they are estimated. **irr** may only be specified when modelling count responses using the **poisson** or **nbinomial** distributions. **irr** may be specified at estimation or when replaying previously estimated results.

rrr reports the fixed-effects coefficients transformed to relative-risk ratios, i.e., exp(b) rather than b. Standard errors and confidence intervals are similarly transformed. This option affects how results are displayed, not how they are estimated. **rrr** may only be specified when modelling nominal categorical responses using the **multinomial** distribution. **rrr** may be specified at estimation or when replaying previously estimated results.

sd shows random-effects variance parameter estimates as standard deviations. Standard errors and confidence intervals are similarly transformed. This option affects how results are displayed, not how they are estimated. **sd** may be specified at estimation or when replaying previously estimated results.

correlations shows random-effects covariance parameter estimates as correlations. Standard errors and confidence intervals are similarly transformed. This option affects how results are displayed, not how they are estimated. **correlations** may be specified at estimation or when replaying previously estimated results.

noheader suppresses the output header, either at estimation or upon replay.

nogroup suppresses the display of group summary information (number of groups, average group size, minimum, and maximum) from the output header. **nogroup** may be specified at estimation or when replaying previously estimated results.

nocontrast suppresses the display of contrast summary information (number of contrasts, description of each contrast) from the output header. This option is only relevant for multinomial response models. **nocontrast** may be specified at estimation or when replaying previously estimated results.

nofetable suppresses the fixed-effects table from the output, either at estimation or upon replay.

noretable suppresses the random-effects table from the output, either at estimation or upon replay.

nodiagnostics prevents MCMC diagnostics from being calculated. This is a helpful option to specify if you are running a simulation study using MCMC estimation as the calculation of the MCMC diagnostics can take some time for models fitted to large data sets. Note that specifying this option will not allow the **or**, **irr**, **rrr**, **sd**, **correlation**, **mode** and **median** options to be specified. **nodiagnostics** may be specified at estimation or when replaying previously estimated results.

mode reports the parameter estimates as the modes of the MCMC chains rather than the means. **mode** may be specified at estimation or when replaying previously estimated results.

median reports the parameter estimates as the medians of the MCMC chains rather than the means. **median** may be specified at estimation or when replaying previously estimated results.

zratio reports classical z-ratios and p-values (i.e. under the assumption that the chains are normally distributed) **zratio** may be specified at estimation or when replaying previously estimated results.

Post-estimation

simulate(*newvar***)** simulates a new response variable based on the estimated model parameters. Make sure to specify the IGLS random number **seed** to be able to replicate the simulated responses.

residuals(*stub*****, ***residuals_options***)** calculates posterior estimates of the residuals and their associated standard errors for all random effects specified at the given level. Posterior estimates are also known as empirical Bayes estimates or best linear unbiased predictions (BLUPs) of the random effects. **runmlwin** will name the residuals. For example if there are three random effects terms in the model, **runmlwin** would name the residuals *stub0*, *stub1*, *stub2* and would name their associated standard errors as *stub0se*, *stub1se*, *stub2se*.

The **variances** 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 postfix for these four types are *std*, *lev*, *inf* and *del* respectively. For example if there are two random effects terms in the model, **runmlwin** would name the standardised residuals *stub0std*, *stub1std* and *stub2std*.

The **sampling** option calculates the sampling variance covariance matrix for the residuals. For example if there are three random effects terms in the model, **runmlwin** would name the standardised residuals *stub0var*, *stub01cov*, *stub1var*, *stub02cov*, *stub12cov*, *stub2var*.

The **norecode** option prevents residuals with values exceedingly close or equal to zero from being recoded to missing.

The **reflate** option returns unshrunken residuals.

savechains(*filename* [**,** *replace*]**)** saves the MCMC parameter estimates for each iteration in filename.dta.

imputeiterations(*numlist***)** imputes missing values at specified iterations. It is important to specify a sufficiently high number of iterations between imputations to reduce the correlation between the sets of imputed values.

imputesummaries for each missing value, calculates the mean and the standard deviation of the chain for that missing value.

Export

viewmacro view the MLwiN macro for the fitted model. This option is useful if you wish to learn how to write your own MLwiN macros.

savemacro(*filename*[**,** *replace*]**)** saves the MLwiN macro for the fitted model. The **replace** option overwrites the MLwiN macro if it already exists.

saveworksheet(*filename*[**,** *replace*]**)** saves the MLwiN worksheet for the fitted model. The **replace** option overwrites the MLwiN worksheet if it already exists.

Other

forcesort forces the data sent to MLwiN to be sorted according to the model hierarchy. We recommend that users sort their data manually using the **sort** command prior to using **runmlwin**.

nosort prevents **runmlwin** from checking that the data are sorted according to the model hierarchy. When this option is used **runmlwin** does not report the table summarizing groups and the **residuals()** and **fscores()** options are not allowed.

forcerecast forces a recast of all variables saved as long or double to float. forcerecast should be used with caution. forcerecast is for those instances where you have a variable saved as a long or double but would now be satisfied to have the variable stored as a float, even though that would lead to a slight rounding of its values. An important example of when this is inappropriate is when identifiers variables are saved as long or double. A slight rounding of the values of identifiers will lead to a merging of units.

nodrop prevents variables that do not appear in the model from being dropped prior to sending the data to MLwiN.

nomlwin prevents MLwiN from being run. When used in conjunction with the **viewmacro** option, the user can examine the MLwiN macro that **runmlwin** writes, without having to fit the associated model in MLwiN.

mlwinpath(*string***)** specifies the file address for mlwin.exe, including the file name.

For example: **mlwinpath(C:\Program Files\MLwiN v2.26\i386\mlwin.exe)**.

mlwinscriptpath(*string***)** is an advanced option which specifies the file address for mlnscript.exe, including the file name.

For example: **mlwinscriptpath(C:\Program Files\MLwiN v2.26\i386\mlnscript.exe)**.

mlnscript.exe is a command line version of MLwiN, which only runs scripts. Note, that runmlwin will only call mlnscript.exe when **batch** is specified. See *Remarks on running runmlwin in batch mode* below for more information.

mlwinsettings(*mlwin_settings***)** manually overrides MLwiN's default settings. The default behaviour is that **runmlwin** will examine the specified model and then automatically override MLwiN's default **size**, **levels**, **columns** and **variables** settings with model specific settings. However, a potential issue is that **runmlwin** typically overrides MLwiN's default settings with settings that are conservative (i.e. settings that are typically higher than the minimum required to fit the model). Thus, **runmlwin** will assign more RAM to MLwiN than is strictly necessary. Users who run into error messages relating to a shortage of RAM on their computer, may therefore wish to experiment with manually overriding these settings to attempt to get the model to fit.

mlwin_settings are

size(*#***)** specifies the maximum worksheet size allowed in MLwiN

levels(*#***)** specifies the maximum number of levels allowed in MLwiN

columns(*#***)** specifies the maximum number of data columns allowed in MLwiN

variables(*#***)** specifies the maximum number of modelled variables allowed in MLwiN

tempmat instructs MLwiN to use memory allocated to the worksheet to store temporary matrices used by the (R)IGLS algorithm

optimat instructs MLwiN to limit the maximum matrix size that can be allocated by the (R)IGLS algorithm. Specify this option if MLwiN gives the following error message "Overflow allocating smatrix". This error message arises if one more higher-level units is extremely large (contains more than 800 lower-level units). In this situation **runmlwin**'s default behaviour is to instruct MLwiN to allocate a larger matrix size to the (R)IGLS algorithm than is currently possible. Specifying **optimat** caps the maximum matrix size at 800 lower-level units, circumventing the MLwiN error message, and allowing most MLwiN functionality.

nopause suppresses the two pause steps in MLwiN. This option is very useful if you want to run a do-file containing a series of **runmlwin** models. MLwiN will automatically launch and exit once each specified model has been fitted. MLwiN will not display the Equations window, but estimation progress is indicated by the progress gauges in the bottom left hand corner of the MLwiN software. See *Examples* for an example of this option.

batch suppresses the two pause steps in MLwiN and prevents the MLwiN software from being displayed. This option is very useful if you want to perform a simulation study. MLwiN will automatically launch and exit once each specified model has been fitted, but this will not be visible to the user. A limitation of this option is that there is no way of monitoring a model's estimation progress. For example, it is not possible to see for how many iterations a model has been iterating for.

This option can also be used in conjunction with running Stata in batch mode in an environment without an interactive session. Examples of this would be running Stata from a task scheduler (see http://www.stata.com/support/faqs/win/batch.html) or submitting jobs to a cluster. When this option is used any error messages produced by MLwiN are displayed in Stata after the model is run.

In addition, if you have used **mlwinscriptpath()** or the MLwiNScript_path **global** to specify the mlnscript.exe file address, then specifying **batch** will run the model using mlnscript.exe rather than mlwin.exe. See *Remarks on running runmlwin in batch mode* below for more information.

Remarks

Remarks are presented under the following headings:

Remarks on alternative Stata commands for fitting multilevel models

The multilevel models fitted by **runmlwin** are often considerably faster than those fitted by the Stata's **[XT] xtmixed**, **[XT] xtmelogit** and **[XT] xtmepoisson** commands. The range of models which can be fitted by **runmlwin** is also much wider than those available through those commands. **runmlwin** also allows fast estimation on large data sets for many of the more complex multilevel models available through the user written **gllamm** command. Rabe-Hesketh and Skrondal (2012) is an outstanding resource for readers wanting to first familiarise themselves with each of these pre-existing Stata commands. The Stata manual help pages for these commands also provide much useful information.

Remarks on downloading runmlwin

The recommended way to install **runmlwin** is to type the following from a net-aware version of Stata

```
. ssc install runmlwin
```

and this will install **runmlwin** from its official location on the Statistical Software Components (SSC) archive.

Remarks on downloading runmlwin manually

Some users will not be allowed to download Stata ado packages to their computer, for example students in computer labs. For these users we recommend that IT support at their institutions install **runmlwin** centrally for them. However, if this is not possible, then we recommend users manually change their default Stata ado package download location to one where they are allowed to save files. Users then need to instruct Stata where on their computer this location is. This process can be semi-automated by issuing the following five commands where users must change the directory path "C:\temp" to a path where they can save files.

(1) Store the original PLUS directory path in the **global** macro sysdir_plus

```
. global sysdir_plus = c(sysdir_plus)
```

(2) Change the PLUS directory path to a directory where you can save files

```
. sysdir set PLUS "C:\temp"
```

(3) Install **runmlwin** to this new directory

```
. ssc install runmlwin
```

(4) Revert back to the original PLUS directory path

. sysdir set PLUS "$sysdir_plus"

(5) Add the **runmlwin** directory to the ado-file directory path

. adopath + "C:\temp"

Note, you will need to run the last command every time you open Stata. Advanced users may wish do this by inserting the command into the profile do-file profile.do. See **profile**.

Remarks on getting runmlwin working for the first time

In order to get **runmlwin** working, you must:

(1) install the latest version of MLwiN on your computer;

(2) set the full MLwiN path using **mlwinpath(***string***)** or a **global** macro called MLwiN_path.

If you don't have the latest version of MLwiN, visit:

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

MLwiN is free for UK academics (thanks to support from the UK Economic and Social Research Council). A fully unrestricted 30-day trial version is available for non-UK academics.

Advanced users may wish to set the MLwiN path every time Stata is started by simply inserting the following line into the profile do-file profile.do. See **profile**.

. global MLwiN_path "C:\Program Files\MLwiN v2.26\i386\mlwin.exe"

Where you must substitute the MLwiN path that is correct for your computer for the path given in quotes in the above example.

Remarks on running runmlwin in batch mode

Advanced users may wish to additionally specify the **mlwinscriptpath(***string***)** or a **globa**
> l macro called MLwiNScript_path in order to run **runmlwin** using **batch**.
The situations where this may be useful include:

(1) Faster loading times/execution;

(2) Fitting models to very large datasets (via the 64-bit version of mlnscript.exe);

(3) Running **runmlwin** in environments where no graphical user interface (GUI) is available, for example when run as a scheduled task or on Unix (Linux or Mac OSX) type systems.

Users may wish to set the MLwiN script path every time Stata is started by simply inserting the following line into the profile do-file profile.do. See **profile**.

32-bit users should point to the 32-bit version of mlnscript.exe.

. global MLwiNScript_path "C:\Program Files\MLwiN v2.26\i386\mlnscript.exe"

64-bit users should point to the 64-bit version of mlnscript.exe.

. global MLwiNScript_path "C:\Program Files (x86)\MLwiN v2.26\x64\mlnscript.exe"

Where in both cases you must substitute the MLwiN script path that is correct for your computer for the path given in quotes in the above example.

Remarks on keeping runmlwin up-to-date

We are constantly improving **runmlwin**. To check that you are using the latest version of **runmlwin**, type the following command:

```
. adoupdate runmlwin
```

Remarks on how runmlwin works

runmlwin carries out the following steps:

(1) Writes an MLwiN macro for the specified multilevel model.

(2) Opens MLwiN and runs the MLwiN macro.

(3) Pauses MLwiN once the model is specified. This allows the user to check that the model is specified as expected. If the model is specified correctly, the user should click the "Resume macro" button (otherwise the user should click the "Abort macro" button to return control to Stata).

(4) Fits the model in MLwiN.

(5) Pauses MLwiN once the model has been fitted (i.e. converged). This allows the user to examine the model results. If the model has fitted correctly, the user should click the "Resume macro" button (otherwise the user should click the "Abort macro" button to return control to Stata).

(6) Stores and displays the model results in Stata

Note that advanced users can use the **nopause** option to suppress steps (3) and (5). This is essential when running simulation studies.

Remarks on estimation procedures in MLwiN: (R)IGLS and MCMC

MLwiN uses two principal estimation procedures:

(1) Iterative Generalised Least Squares (IGLS) equivalent to maximum likelihood under normality and Restrictive Iterative Generalised Least Squares (RIGLS) which is formally equivalent to residual maximum likelihood (REML) under normality. See *Remarks on IGLS vs. RIGLS* below for more information.

(2) Markov Chain Monte Carlo (MCMC) estimation. See *Remarks on MCMC* below for more information.

In addition, for discrete response models fitted by (R)IGLS, quasilikelihood estimates are calculated. See *Remarks on quasilikelihood estimates: MQL1, MQL2, PQL1 and PQL2* below for more information.

Remarks on IGLS vs. RIGLS

igls and **rigls** will give almost identical results in models where the number of units at each level is high. The methods give different results, particularly for the random part parameters, when the number of units at a given level are few. For example, the **rigls** estimate for the level 2 variance in a two-level normal response model will be larger than the **igls** estimate when there are few level 2 units.

Remarks on quasilikelihood estimates: MQL1, MQL2, PQL1 and PQL2

mql1, **mql2**, **pql1** and **pql2** specify the linearization technique for fitting discrete response models by (R)IGLS. All four quasilikelihood methods are approximate: **pql2** is the most accurate but the least stable and the slowest to converge, **mql1** is the least accurate but the most stable and fastest to converge. We recommend that model exploration is conducted using **mql1**. For any final model, we recommend fitting the model using **pql2** (or preferably MCMC) as a two stage process. First fit the model using **mql1**, then refit the model using **pql2** where the **initsprevious** option (or **initsmodel(***name***)** or initsb(matrix)) is specified to use the **mql1** parameter estimates as the starting values for fitting the model using **pql2**.

Remarks on MCMC

Markov Chain Monte Carlo (MCMC) methods are Bayesian estimation techniques which can be used to estimate multilevel models.

MCMC works by drawing a random sample of values for each parameter from its probability distribution. The mean and standard deviation of each random sample gives the point estimate and standard error for that parameter.

We start by specifying the model and our prior knowledge for each parameter (we nearly always specify that we have no knowledge!). Next we specify initial values for the model parameters (nearly always the IGLS estimates). We then run the MCMC algorithm until each parameter distribution has settled down to its stationary distribution. (i.e. the burnin period when the chains are converging to their posterior distribution). We then run the MCMC algorithm for a further period (the monitoring period) in order to store a monitoring chain for each parameter. Point estimates and standard errors are given by the means and standard deviations of these monitoring chains.

An important aspect of MCMC is specifying initial values. Users can specify that the initial values are the parameter estimates from the previous model, **initsprevious**. Alternatively, they can specify that the initial values are the parameter estimates from some other previously stored model, **initsmodel(***name***)**. Or they can even manually specify any set of initial values they like, **initsb(***matrix***)**.

A second important aspect of MCMC is the prior knowledge (i.e. prior distribution) that we specify for each parameter. By default MLwiN sets diffuse or uninformative priors, and these can be used to approximate maximum likelihood estimation. Users can specify informative priors using the **priormatrix(***matrix***)** and **rppriors(***rppriors_spec***)** options.

We recommend users seeking further information, examples and references, to consult the comprehensive MLwiN MCMC manual by Browne (2012) and additionally the help system within MLwiN. The MLwiN MCMC manual also gives lengthier explanations for all MCMC options implemented in **runmlwin**.

Remarks on Bayesian DIC

The Bayesian Deviance Information Criterion (DIC) is an MCMC penalised goodness of fit measure. It is equivalent to the Akaike Information Criterion (AIC) used in maximum likelihood estimation.

The AIC is given by

    AIC = -2*logL + 2k = Deviance + 2k

where

    L is the maximized value of the likelihood function (i.e. the likelihood evaluated at the maximum likelihood point estimates of the model parameters)

    k is the number of model parameters

In MCMC estimation, the DIC statistic has an analogous definition:

    DIC = Deviance + 2*p_d

where:

The Deviance is evaluated at the posterior means of the model parameters

p_d is the effective number of model parameters

The four statistics reported in the standard **runmlwin** model output are:

Dbar: The average goodness of fit of the model over the iterations.

D(thetabar): The goodness of fit of the model evaluated at the posterior means of the model parameters.

pD: The effective number of parameters summarises the complexity of the model: pD = Dbar - D(thetabar).

DIC: The statistic of interest: DIC = Dbar + pD = D(thetabar) + 2pD.

Remarks on using sampling weights

Sampling weights are only available for estimation using (R)IGLS. Sampling weights should therefore only be used for continuous response variables as the quasilikelihood procedures available for (R)IGLS estimation of discrete response variables are only approximate. See *Remarks on quasilikelihood estimates: MQL1, MQL2, PQL1 and PQL2* above for more information. We recommend that sampling weights should always be standardised and that sandwich estimates should always be used for the sampling estimates of both fixed part and random part parameter estimates. These recommendations are implemented in the default settings for **runmlwin**, but can be changed using the **weights** option. Note also that if level 2 weights are specified then MLwiN requires the level 1 weights to be conditional level 1 weights.

Remarks on using multiple membership weights

Consider a two-level multiple membership model of students (level 1) who are multiple members of schools (level 2). In this example, the number of multiple membership unit identifier variables specified should equal the maximum number of schools attended by any given student. Suppose this maximum number of schools attended is three, then there should be three multiple membership unit identifier variables. Intuitively, these can be thought of as corresponding to the first school attended, the second school attended and the third school attended, respectively. However, the order in which the (potentially) three school IDs appears is irrelevant other than it must correspond to the ordering of the associated multiple membership weight variables.

For students who attend three schools, all *three* of the multiple membership unit identifier variables should take different non-zero values and these values should give the school IDs of the *three* different schools attended.

For students who attend two schools, *two* of the three multiple membership unit identifier variables should take non-zero values and these values should give the school IDs of the *two* different schools attended. The third multiple membership unit identifier variable must take the value zero to indicate that no third school was attended. No other value than zero (not even a missing value) is permitted to indicate that a third school was not attended. A consequence of this is that zero is the only invalid school identifier value; no school in the data should have a zero value.

Finally, for students who attend a single school, one of the three multiple membership unit identifier variables should take a non-zero value and this value should give the school ID of the *single* school attended. The second and third multiple membership unit identifier variables should take zero values to indicate that no second or third schools were attended.

Remarks on MLwiN estimation problems and error messages

Multilevel models are complex, often involving multiple sets of random effects at multiple levels. Users may sometimes run into MLwiN error messages. Help for a range of common MLwiN error messages are provided on the MLwiN website:

http://www.bristol.ac.uk/cmm/software/support/support-faqs/errors.html#error > message

Examples

IMPORTANT. The following examples will only work on your computer once you have installed MLwiN and once you have told **runmlwin** what the mlwin.exe file address is. See *Remarks on installing runmlwin* above for more information.

(a) Continuous response models

Two-level models

Setup

```
    . use http://www.bristol.ac.uk/cmm/media/runmlwin/tutorial, clear
```

Two-level random-intercept model, analogous to xtreg (fitted using IGLS)
(See Section 2.5 of the MLwiN User Manual)

```
    . runmlwin normexam cons standlrt, level2(school: cons) level1(student:
        cons) nopause
```

Two-level random-intercept and random-slope (coefficient) model (fitted using IGLS)
(See Section 4.4 of the MLwiN User Manual)

```
    . runmlwin normexam cons standlrt, level2 (school: cons standlrt) level1
        (student: cons) nopause
```

Refit the model, where this time we additionally calculate the level 2 residuals (fitted using IGLS)
(See Section 4.4 of the MLwiN User Manual)

```
    . runmlwin normexam cons standlrt, level2 (school: cons standlrt,
        residuals(u)) level1 (student: cons) nopause
```

Two-level random-intercept and random-slope (coefficient) model with a complex level 1 variance function (fitted using IGLS)
(See Section 7.3 of the MLwiN User Manual)

```
    . matrix A = (1,1,0,0,0,1)
    . runmlwin normexam cons standlrt girl, level2(school: cons standlrt)
        level1(student: cons standlrt girl, elements(A)) nopause
```

Two-level random-intercept and random-slope (coefficient) model using MCMC (where we first fit the model using IGLS to obtain initial values for the MCMC chains)
(See Section 6.0 of the MLwiN MCMC Manual)

```
    . runmlwin normexam cons standlrt, level2 (school: cons standlrt) level1
        (student: cons) nopause
    . runmlwin normexam cons standlrt, level2 (school: cons standlrt) level1
        (student: cons) mcmc(on) initsprevious nopause
```

Multivariate response models

Setup

```
    . use http://www.bristol.ac.uk/cmm/media/runmlwin/gcsemv1, clear
```

Random-intercept bivariate response model (fitted using IGLS)
(See Section 14.3 of the MLwiN User Manual)

```
    . runmlwin (written cons female, eq(1)) (csework cons female, eq(2)),
        level2(school: (cons, eq(1)) (cons, eq(2))) level1(student:   (cons,
        eq(1)) (cons, eq(2))) nopause
```

Cross-classified models

Setup

```
    . use http://www.bristol.ac.uk/cmm/media/runmlwin/xc, clear
```

Two-way cross-classified model (fitted using MCMC where starting values for the MCMC chains are manually specified by the user)
(See Section 15.4 of the MLwiN MCMC Manual)

```
    . matrix b = (0,.33,.33,.33)
    . runmlwin attain cons, level3(sid: cons) level2(pid: cons) level1(pupil:
        cons) mcmc(cc) initsb(b) nopause
```

(b) Discrete response models

Binary response multilevel models

Setup

```
    . use http://www.bristol.ac.uk/cmm/media/runmlwin/bang, clear
    . generate lc1 = (lc==1)
    . generate lc2 = (lc==2)
    . generate lc3plus = (lc>=3)
```

Two-level random intercepts logit model (fitted using IGLS MQL1)
(See Section 9.3 of the MLwiN User Manual)

```
    . runmlwin use cons lc1 lc2 lc3plus age, level2(district: cons)
        level1(woman) discrete(distribution(binomial) link(logit)
        denominator(cons)) nopause
```

Two-level random intercepts logit model (fitted using IGLS PQl2 where IGLS MQL1 estimates from previous model are used as initial values)
(See Section 9.3 of the MLwiN User Manual)

```
    . runmlwin use cons lc1 lc2 lc3plus age , level2(district: cons)
        level1(woman) discrete(distribution(binomial) link(logit)
        denominator(cons) pql2) initsprevious nopause
```

Two-level random intercepts probit model (fitted using IGLS PQL2)

```
    . runmlwin use cons lc1 lc2 lc3plus age, level2(district: cons)
        level1(woman) discrete(distribution(binomial) link(probit)
        denominator(cons) pql2) nopause
```

Two-level random intercepts probit model (fitted using MCMC where IGLS PQL2 estimates from previous model are used as initial values)
(See Section 10.2 of the MLwiN MCMC Manual)

```
    . runmlwin use cons lc1 lc2 lc3plus age, level2(district: cons)
        level1(woman) discrete(distribution(binomial) link(probit)
        denominator(cons)) mcmc(on) initsprevious nopause
```

Unordered multinomial response models

Setup

```
    . use http://www.bristol.ac.uk/cmm/media/runmlwin/bang, clear
    . generate lc1 = (lc==1)
    . generate lc2 = (lc==2)
    . generate lc3plus = (lc>=3)
```

Two-level random intercepts unordered multinomial model (fitted using IGLS MQL1)
(See Section 10.5 of the MLwiN User Manual)

```
    . runmlwin use4 cons lc1 lc2 lc3plus, level2(district: cons) level1(woman)
        discrete(distribution(multinomial) link(mlogit) denom(cons)
        basecategory(4)) nopause
```

Ordered multinomial response models

Setup

```
    . use http://www.bristol.ac.uk/cmm/media/runmlwin/alevchem, clear
    . egen school = group(lea estab)
    . generate gcseav = gcse_tot/gcse_no
    . egen gcseav_rank = rank(gcseav)
    . generate gcseav_uniform = (gcseav_rank - 0.5)/_N
    . generate gcseavnormal = invnorm(gcseav_uniform)
```

Two-level random intercepts ordered multinomial model with common coefficients for predictor variable gcseavnormal (fitted using IGLS PQL2)
(See Section 11.4 of the MLwiN User Manual)

```
    . runmlwin a_point cons (gcseavnormal, contrast(1/5)), level2(school: (cons,
        contrast(1/5))) level1(pupil) discrete(distribution(multinomial)
        link(ologit) denom(cons) base(6) pql2) nopause
```
(Click to run)

Two-level random intercepts ordered multinomial model with separate coefficients for predictor variable gcseavnormal (fitted using IGLS PQL2)
(See Section 12.3 of the MLwiN User Manual)

```
    . runmlwin a_point cons gcseavnormal, level2(school: (cons, contrast(1/5)))
        level1(pupil) discrete(distribution(multinomial) link(ologit) denom
        (cons) base(6) pql2) nopause
```

Count data model

Setup

```
    . use http://www.bristol.ac.uk/cmm/media/runmlwin/mmmec, clear
    . generate lnexpected = ln(exp)
```

Three-level random intercepts Poisson model (fitted using RIGLS MQL1)
(See Section 12.3 of the MLwiN User Manual)

```
    . runmlwin obs cons uvbi, level3(nation: cons) level2(region: cons)
        level1(county) discrete(distribution(poisson) offset(lnexpected)) rigls
        nopause
```

Three-level random intercepts Poisson model (fitted using MCMC where IGLS MQL1 estimates from previous model are used as initial values)
(See Section 11.3 of the MLwiN MCMC Manual)

```
    . runmlwin obs cons uvbi, level3(nation: cons) level2(region: cons)
        level1(county) discrete(distribution(poisson) offset(lnexpected))
        mcmc(burnin(5000) chain(50000) refresh(500)) initsprevious nopause
```
(Click to run)

(c) Multivariate response models

Multivariate discrete and mixed response models

Setup

```
    . use http://www.bristol.ac.uk/cmm/media/runmlwin/tutorial, clear
    . generate binexam = (normexam>0)
    . generate binlrt = (standlrt>0)
```

Two-level bivariate binary response probit model (fitted using IGLS MQL1)
(See Section 14.5 of the MLwiN User Manual)

```
    . runmlwin (binexam cons, equation(1)) (binlrt cons, equation(2)),
        level1(student:) discrete(distribution(binomial binomial) link(probit)
        denominator(cons cons)) nosort nopause
```

Two-level mixed bivariate continuous and binary response probit model (fitted using IGLS MQL1)
(See Section 14.5 of the MLwiN User Manual)

```
    . runmlwin (normexam cons, equation(1)) (binlrt cons, equation(2)),
        level1(student:(cons, equation(1)) (cons, equation(2)))
        discrete(distribution(normal binomial) link(probit) denominator(cons
        cons)) nosort nopause
```
(Click to run)

A full range of **runmlwin** examples using both (R)IGLS and MCMC is available at:

http://www.bristol.ac.uk/cmm/software/runmlwin/examples/

These include do-files which allow you to replicate all the analyses reported in the MLwiN User Manual (Rasbash et al., 2012) and the MCMC MLwiN Manual (Browne, 2012).

The log files for these two manuals are presented below.

MLwiN User Manual
- 1 - Introducing Multilevel Models
- 2 - Introduction to Multilevel Modelling
- 3 - Residuals
- 4 - Random Intercept and Random Slope Models
- 5 - Graphical Procedures for Exploring the Model
- 6 - Contextual Effects
- 7 - Modelling the Variance as a Function of Explanatory Variables
- 8 - Getting Started with your Data
- 9 - Logistic Models for Binary and Binomial Responses
- 10 - Multinomial Logistic Models for Unordered Categorical Responses
- 11 - Fitting an Ordered Category Response Model
- 12 - Modelling Count Data
- 13 - Fitting Models to Repeated Measures Data
- 14 - Multivariate Response Models
- 15 - Diagnostics for Multilevel Models
- 16 - An Introduction to Simulation Methods of Estimation
- 17 - Bootstrap Estimation
- 18 - Modelling Cross-classified Data
- 19 - Multiple Membership Models

MLwiN MCMC Manual
- 1 - Introduction to MCMC Estimation and Bayesian Modelling
- 2 - Single Level Normal Response Modelling
- 3 - Variance Components Models
- 4 - Other Features of Variance Components Models
- 5 - Prior Distributions, Starting Values and Random Number Seeds
- 6 - Random Slopes Regression Models
- 7 - Using the WinBUGS Interface in MLwiN
- 8 - Running a Simulation Study in MLwiN
- 9 - Modelling Complex Variance at Level 1 - Heteroscedasticity
- 10 - Modelling Binary Responses
- 11 - Poisson Response Modelling
- 12 - Unordered Categorical Responses
- 13 - Ordered Categorical Responses
- 14 - Adjusting for Measurement Errors in Predictor Variables
- 15 - Cross Classified Models
- 16 - Multiple Membership Models
- 17 - Modelling Spatial Data
- 18 - Multivariate Normal Response Models and Missing Data
- 19 - Mixed Response Models and Correlated Residuals
- 20 - Multilevel Factor Analysis Modelling
- 21 - Using Structured MCMC
- 22 - Using the Structured MVN framework for models
- 23 - Using Orthogonal fixed effect vectors
- 24 - Parameter expansion
- 25 - Hierarchical Centring

Saved results

runmlwin saves the following in e():

Scalars

e(numlevels)	number of levels
e(N)	number of observations
e(k)	number of parameters
e(k_f)	number of FE parameters
e(k_r)	number of RE parameters
e(extrabinomial)	1 if extra binomial variation is used, 0 otherwise
e(ll)	log (restricted) likelihood
e(deviance)	deviance (= -2*e(ll))
e(iterations)	number of iterations
e(converged)	1 if converged, 0 otherwise
e(time)	estimation time (seconds)
e(burnin)	number of burn-in iterations
e(chain)	number of chain iterations
e(thinning)	frequency with which successive values in the chain are stored
e(mcmcnofit)	1 if MCMC model is not fitted, 0 otherwise
e(mcmcdiagnostics)	1 if MCMC diagnostics have been calculated, 0 otherwise
e(dbar)	average deviance across the chain iterations

e(dthetabar)	deviance at the mean values of the model parameters
e(pd)	effective number of parameters
e(dic)	Bayesian deviance information criterion
e(size)	maximum worksheet size allowed in MLwiN
e(maxlevels)	maximum number of levels allowed in MLwiN
e(columns)	maximum number of data columns allowed in MLwiN
e(variables)	maximum number of modelled variables allowed in MLwiN
e(tempmat)	1 if use memory allocated to the worksheet to store temporary matrices in MLwiN, 0 otherwise

Macros

e(cmd)	runmlwin
e(version)	runmlwin version
e(cmdline)	command as typed
e(title)	title in estimation output
e(depvars)	name(s) of dependent variable(s)
e(distribution)	distribution(s)
e(link)	link function
e(denominators)	denominator(s)
e(basecategory)	the value of *depvar* to be treated as the base category
e(respcategories)	the values of *depvar*
e(offsets)	offset(s)
e(ivars)	grouping variables
e(level1id)	level 1 identifier variable
e(weightvar)	sampling weights variables
e(weighttype)	sampling weights types
e(method)	estimation method: IGLS, RIGLS or MCMC
e(linearization)	linearization technique: MQL1, MQL2, PQL1 or PQL2
e(properties)	b V
e(chains)	MCMC chains for all parameters

Matrices

e(b)	coefficient vector
e(V)	variance-covariance matrix of the estimators
e(N_g)	group counts
e(g_min)	group size minimum
e(g_avg)	group size averages
e(g_max)	group size maximums
e(RP2)	level 2 matrix of random part parameters
e(RP1)	level 1 matrix of random part parameters
e(P)	priors
e(sd)	standard deviation for each chain
e(mode)	mode for each chain
e(meanmcse)	Monte Carlo standard error (MCSE) evaluated at the mean of each chain
e(ess)	effective sample size (ESS) for each chain
e(quantiles)	quantiles for each chain
e(lb)	lower credible interval bound for each chain
e(ub)	upper credible interval bound for each chain
e(KD1)	kernel density for each chain
e(KD2)	kernel density for each chain
e(ACF)	autocorrelation function (ACF) for each chain
e(PACF)	partial autocorrelation function (PACF) for each chain
e(MCSE)	Monte Carlo standard error (MCSE) for each chain
e(bd)	Brooks-Draper diagnostic for mean of each chain
e(rl1)	Raftery-Lewis diagnostic for each 2.5th quantile of each chain
e(rl2)	Raftery-Lewis diagnostic for each 97.5th quantile of each chain
e(pvalmean)	one-sided Bayesian p-value for each chain where chain mean is treated as parameter estimate
e(pvalmode)	one-sided Bayesian p-value for each chain where chain mode is treated as parameter estimate
e(pvalmedian)	one-sided Bayesian p-value for each chain where chain median is treated as parameter estimate

Functions

e(sample)	marks estimation sample

About the Centre for Multilevel Modelling

The MLwiN software is developed at the Centre for Multilevel Modelling. The Centre was established in 1986, and has been supported largely by project grants from the UK Economic and Social Research Council. The Centre has been based at the University of Bristol since 2005.

The Centre s website:

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

contains much of interest, including new developments, and details of courses and workshops. This website also contains the latest information about the MLwiN software, including upgrade information, maintenance downloads, and documentation.

The Centre also runs a free online multilevel modelling course:

http://www.bristol.ac.uk/cmm/learning/course.html

which contains modules starting from an introduction to quantitative research progressing to multilevel modelling of continuous and categorical data. Modules include a description of concepts and models and instructions of how to carry out analyses in MLwiN, Stata and R. There is a also a user forum, videos and interactive quiz questions for learners self-assessment.

How to cite runmlwin and MLwiN

runmlwin is not an official Stata command. It is a free contribution to the research community, like a paper. Please cite it as such:

Leckie, G. and Charlton, C. 2013. **runmlwin** - A Program to Run the MLwiN Multilevel Modelling Software from within Stata. Journal of Statistical Software, 52 (11),1-40. http://www.jstatsoft.org/v52/i11

Similarly, please also cite the MLwiN software:

Rasbash, J., Charlton, C., Browne, W.J., Healy, M. and Cameron, B. 2009. MLwiN Version 2.1. Centre for Multilevel Modelling, University of Bristol.

For models fitted using MCMC estimation, we ask that you additionally cite:

Browne, W.J. 2012. MCMC Estimation in MLwiN, v2.26. Centre for Multilevel Modelling, University of Bristol.

The runmlwin user forum

Please use the **runmlwin** user forum to post any questions you have about **runmlwin**. We will try to answer your questions as quickly as possible, but where you know the answer to another user's question please also reply to them!

http://www.cmm.bristol.ac.uk/forum/

Authors

George Leckie
Centre for Multilevel Modelling
University of Bristol
g.leckie@bristol.ac.uk

Chris Charlton
Centre for Multilevel Modelling
University of Bristol

Acknowledgments

We are very grateful to colleagues at the Centre for Multilevel Modelling and the University of Bristol for their useful comments.

The development of this command was funded under the LEMMA project, a node of the UK Economic and Social Research Council's National Centre for Research Methods (grant number RES-576-25-0003).

Disclaimer

runmlwin comes with no warranty. We recommend that users check their results with those obtained through operating MLwiN by its graphical user interface. Users are also encouraged to check their results with those produced by other statistical software packages.

References

Browne, W.J. 2012. MCMC Estimation in MLwiN, v2.26. Centre for Multilevel Modelling, University of Bristol. http://www.bristol.ac.uk/cmm/software/mlwin/download/manuals.html

Leckie, G. and Charlton, C. 2013. **runmlwin** - A Program to Run the MLwiN Multilevel Modelling Software from within Stata. Journal of Statistical Software, 52 (11),1-40. http://www.jstatsoft.org/v52/i11

Rabe-Hesketh, S. and Skrondal, A. 2012. Multilevel and Longitudinal Modeling using Stata (Third Edition). College Station, TX: Stata Press.

Rasbash, J., Steele, F., Browne, W.J. and Goldstein, H. 2012. A User s Guide to MLwiN, v2.26. Centre for Multilevel Modelling, University of Bristol. http://www.bristol.ac.uk/cmm/software/mlwin/download/manuals.html

Also see

Manual: **[XT] xtmixed [XT] xtmelogit [XT] xtmepoisson [XT] xtreg**

Online: **[XT] xtmixed**, **[XT] xtmelogit**, **[XT] xtmepoisson**, **[XT] xtreg**, **mcmcsum**, **usewsz**, **savewsz**, **reffadjust**, **gllamm**