Title

stata.com

mlexp — Maximum likelihood estimation of user-specified expressions

Syntax | Menu | Description | Options
Remarks and examples | Stored results | Methods and formulas | References
Also see

Syntax

```
mlexp (<lexp>) ... [ if ] [ in ] [ weight ] [ , options ]
```

where

<lexp> is a substitutable expression representing the log-likelihood function.

options	Description
Model	
`variables(`*varlist*`)`	specify variables in model
`from(`*initial_values*`)`	specify initial values for parameters
Derivatives	
`derivative(/`*name* `= <`*dexp*`>)`	specify derivative of *<lexp>* with respect to parameter *name*; can be specified more than once
SE/Robust	
`vce(`*vcetype*`)`	*vcetype* may be `oim`, `opg`, `robust`, `cluster` *clustvar*, `bootstrap`, or `jackknife`
Reporting	
`level(#)`	set confidence level; default is `level(95)`
`title(`*string*`)`	display *string* as title above the table of parameter estimates
`title2(`*string*`)`	display *string* as subtitle
display_options	control column formats
Maximization	
maximize_options	control the maximization process; seldom used
`coeflegend`	display legend instead of statistics

<lexp> may contain time-series operators; see **[U] 13.9 Time-series operators**.

`bootstrap`, `by`, `jackknife`, `rolling`, `statsby`, and `svy` are allowed; see **[U] 11.1.10 Prefix commands**.

Weights are not allowed with the `bootstrap` prefix; see [R] **bootstrap**.

`aweights` are not allowed with the `jackknife` prefix; see [R] **jackknife**.

`aweights`, `fweights`, `iweights`, and `pweights` are allowed; see **[U] 11.1.6 weight**.

`coeflegend` does not appear in the dialog box.

See **[U] 20 Estimation and postestimation commands** for more capabilities of estimation commands.

<lexp> and *<dexp>* are extensions of valid Stata expressions that also contain parameters to be estimated. The parameters are enclosed in curly braces and must otherwise satisfy the naming requirements for variables; `{beta}` is an example of a parameter. Also allowed is a notation of the

form {<*eqname*>:*varlist*} for linear combinations of multiple covariates and their parameters. For example, {xb: mpg price turn} defines a linear combination of the variables mpg, price, and turn. See *Substitutable expressions* under *Remarks and examples* below.

Menu

Statistics > Other > Maximum likelihood estimation of expression

Description

mlexp performs maximum likelihood estimation of models that satisfy the linear-form restrictions, which is to say models for which you can write down the log likelihood for an individual observation and for which the overall log likelihood is simply the sum of the individual observations' log likelihoods.

You express the observation-level log-likelihood function by using a substitutable expression. Unlike models fit using ml, you do not need to do any programming. However, ml can fit classes of models that cannot be fit by mlexp.

Options

Model

variables(*varlist*) specifies the variables in the model. mlexp ignores observations for which any of these variables has missing values. If you do not specify variables(), then mlexp assumes all the observations are valid. If the log likelihood cannot be calculated at the initial values for any observation, mlexp will exit with an error message.

from(*initial_values*) specifies the initial values to begin the estimation. You can specify a $1 \times k$ matrix, where k is the number of parameters in the model, or you can specify parameter names and values. For example, to initialize alpha to 1.23 and delta to 4.57, you would type

```
mlexp ..., from(alpha=1.23 delta=4.57) ...
```

Initial values declared using this option override any that are declared within substitutable expressions. If you specify a parameter that does not appear in your model, mlexp exits with an error. If you specify a matrix, the values must be in the same order in which the parameters are declared in your model. mlexp ignores the row and column names of the matrix.

Derivatives

derivative(/*name* = <*dexp*>) specifies the derivative of the observation-level log-likelihood function with respect to parameter *name*.

<*dexp*> uses the same substitutable expression syntax as is used to specify the log-likelihood function. If you declare a linear combination in the log-likelihood function, you provide the derivative for the linear combination; mlexp then applies the chain rule for you. See *Specifying derivatives* under *Remarks and examples* below for examples.

If you do not specify the derivative() option, mlexp calculates derivatives numerically. You must either specify no derivatives or specify all the derivatives; you cannot specify some analytic derivatives and have mlexp compute the rest numerically.

If you are estimating multiple parameters, you supply derivatives using multiple derivative() specifications.

SE/Robust

`vce(`*vcetype*`)` specifies the type of standard error reported, which includes types that are derived from asymptotic theory (`oim`, `opg`), that are robust to some kinds of misspecification (`robust`), that allow for intragroup correlation (`cluster` *clustvar*), and that use bootstrap or jackknife methods (`bootstrap`, `jackknife`); see [R] ***vce_option***.

Reporting

`level(#)`; see [R] **estimation options**.

`title(`*string*`)` specifies an optional title that will be displayed just above the table of parameter estimates.

`title2(`*string*`)` specifies an optional subtitle that will be displayed between the title specified in `title()` and the table of parameter estimates. If `title2()` is specified but `title()` is not, then `title2()` has the same effect as `title()`.

display_options: `cformat(%`*fmt*`)`, `pformat(%`*fmt*`)`, and `sformat(%`*fmt*`)`; see [R] **estimation options**.

Maximization

maximize_options: `difficult`, `technique(`*algorithm_spec*`)`, `iterate(#)`, `[no]log`, `trace`, `gradient`, `showstep`, `hessian`, `showtolerance`, `tolerance(#)`, `ltolerance(#)`, `nrtolerance(#)`, and `nonrtolerance`; see [R] **maximize**. These options are seldom used.

The following option is available with `mlexp` but is not shown in the dialog box:

`coeflegend`; see [R] **estimation options**.

Remarks and examples

stata.com

Remarks are presented under the following headings:

Introduction
Substitutable expressions
Parameter constraints
Specifying derivatives

Introduction

`mlexp` performs maximum likelihood estimation of models that satisfy the linear-form restrictions, which is to say models for which you can write down the log likelihood for a single observation and for which the overall log likelihood is simply the sum of the individual observations' log likelihoods. Models designed for use with cross-sectional data usually meet the linear-form restrictions, including linear regression, many discrete-choice models, limited-dependent-variable models, and selection models. Examples of models that do not satisfy the linear-form restrictions are random-effects panel-data models (because the likelihood function is defined at the panel level) and Cox proportional hazards models (because the likelihood function is defined for risk sets).

Because of its straightforward syntax and accessibility from the menu system, `mlexp` is particularly suited to users who are new to Stata and to those using Stata for pedagogical purposes. You express the observation-level log-likelihood function by using a substitutable expression, which we explain below. Unlike models fit using `ml`, you do not need to do any programming. However, `ml` can fit classes of models that cannot be fit by `mlexp`, including those that do not meet the linear-form restrictions.

Substitutable expressions

You specify the log-likelihood function that `mlexp` is to maximize by using substitutable expressions that are similar to those used by `nl`, `nlsur`, and `gmm`. You specify substitutable expressions just as you would specify any other mathematical expression involving scalars and variables, such as those expressions you would use with Stata's `generate` command, except that the parameters to be estimated are bound in braces. See **[U] 13.2 Operators** and **[U] 13.3 Functions** for more information on expressions. Parameter names must follow the same conventions as variable names. See **[U] 11.3 Naming conventions**.

For example, say that you have observations on variable x, and the log likelihood for the ith observation is

$$\ln\ell_i = \ln\lambda - \lambda x_i$$

where λ is a parameter to be estimated. Then you would type

```
. mlexp (ln({lambda}) - {lambda}*x)
```

Because λ is a parameter, we enclosed it in braces. To specify initial values for a parameter, you can include an equal sign and the initial value after the parameter; for example,

```
. mlexp (ln({lambda = 0.75}) - {lambda}*x)
```

would initialize λ to be 0.75. If you do not initialize a parameter, `mlexp` initializes it to zero.

Frequently, even nonlinear functions contain linear combinations of variables. Continuing the previous example, say that we want to parameterize λ as

$$\lambda_i = \alpha_1 u_i + \alpha_2 v_i$$

where u and v are variables in the dataset. Instead of typing

```
. mlexp (ln({alpha1}*u + {alpha2}*v) - ({alpha1}*u + {alpha2}*v)*x)
```

you can instead type

```
. mlexp (ln({lambda: u v}) - {lambda:}*x)
```

The notation `{lambda: u v}` indicates to `mlexp` that you want a linear combination of the variables `u` and `v`. We named the linear combination `lambda`, so `mlexp` will name the parameters for the two variables `lambda_u` and `lambda_v`, respectively. Once you have declared a linear combination, you can subsequently refer to the linear combination by specifying its name and a colon inside braces, as we did with this example. You cannot use the same name for both an individual parameter and a linear combination. However, after a linear combination has been declared, you can refer to the parameter of an individual variable within that linear combination by using the notation {*lc*_*z*}, where *lc* is the name of the linear combination and z is the variable whose parameter you want to reference. Linear combinations do not include a constant term.

There are three rules to follow when defining substitutable expressions:

1. Parameters of the model are bound in braces: `{b0}`, `{param}`, etc.
2. Initial values for parameters are given by including an equal sign and the initial value inside the braces: `{b0=1}`, `{param=3.571}`, etc.
3. Linear combinations of variables can be included using the notation {*eqname*:*varlist*}: `{xb: mpg price weight}`, `{score: w x z}`, etc. Parameters of linear combinations are initialized to zero.

If you specify initial values by using the `from()` option, they override whatever initial values are given within the substitutable expression. Substitutable expressions are so named because once values are assigned to the parameters, the resulting expressions can be handled by `generate` and `replace`.

Regardless of whether you specify initial values, `mlexp` performs a search procedure for better starting values before commencing the first iteration of the maximization routine. If you specify initial values, the search procedure tries to improve upon those values. Otherwise, the search procedure begins with all parameters set to zero.

▷ Example 1: The gamma density function

The two-parameter gamma density function for $y \geq 0$ is

$$f(y) = \frac{\lambda^P}{\Gamma(P)} \exp(-\lambda y) y^{P-1} \qquad \lambda > 0,\ P > 0$$

so that the log likelihood for the ith observation is

$$\ln\ell_i = P \ln\lambda - \ln\Gamma(P) - \lambda y_i + (P-1) \ln y_i$$

The dataset `greenegamma.dta`, based on Greene (2012, 460–461), contains 20 observations drawn randomly from the two-parameter gamma distribution. We want to estimate the parameters of that distribution. We type

```
. use http://www.stata-press.com/data/r13/greenegamma
. mlexp ({P}*ln({lambda}) - lngamma({P}) - {lambda}*y + ({P}-1)*ln(y))
initial:       log likelihood =     -<inf>  (could not be evaluated)
feasible:      log likelihood = -363.37264
rescale:       log likelihood = -153.09898
rescale eq:    log likelihood = -88.863468
Iteration 0:   log likelihood = -88.863468
Iteration 1:   log likelihood = -85.405011
Iteration 2:   log likelihood = -85.375857
Iteration 3:   log likelihood = -85.375669
Iteration 4:   log likelihood = -85.375669
Maximum likelihood estimation
Log likelihood = -85.375669                     Number of obs   =         20
------------------------------------------------------------------------------
             |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
          /P |   2.410602   .7158452     3.37   0.001     1.007571    3.813633
    /lambda |   .0770702   .0254361     3.03   0.002     .0272164    .1269241
------------------------------------------------------------------------------
```

In our substitutable expression for the log-likelihood function, we enclosed the two parameters of our model, P and λ, in curly braces. We used the `lngamma()` function to compute $\ln\Gamma(P)$ because Stata (unlike Mata) does not have a built-in function to compute $\Gamma(P)$, and numerical algorithms for computing $\ln\Gamma(P)$ directly are more accurate than taking the natural logarithm of $\Gamma(P)$, anyway.

Because we did not specify initial values, `mlexp` initialized P and λ to be zero. When both parameters are zero, the log-likelihood function cannot be evaluated because $\ln(0)$ is undefined. Therefore, in the iteration log above the coefficient table, we see that `mlexp` reported the initial log likelihood to be `-<inf> (could not be evaluated)`. `mlexp` uses a search routine to find alternative initial values that do allow the log-likelihood function to be calculated.

◁

▷ Example 2: Obtaining alternative VCEs

In example 1, by default `mlexp` reported standard errors based on the observed information matrix of the log-likelihood function. See [R] ***vce_option*** for an overview or Gould, Pitblado, and Poi (2010) for an in-depth discussion of different ways of obtaining the VCE in maximum likelihood estimation. With `mlexp`, we can use the `vce()` option to obtain standard errors based on alternative VCEs. For example, to obtain the outer product of gradients (OPG) standard errors, we type

```
. mlexp ({P}*ln({lambda}) - lngamma({P}) - {lambda}*y + ({P}-1)*ln(y)), vce(opg)
initial:       log likelihood =     -<inf>  (could not be evaluated)
feasible:      log likelihood = -363.37264
rescale:       log likelihood = -153.09898
rescale eq:    log likelihood = -88.863468
Iteration 0:   log likelihood = -88.863468
Iteration 1:   log likelihood = -85.405011
Iteration 2:   log likelihood = -85.375857
Iteration 3:   log likelihood = -85.375669
Iteration 4:   log likelihood = -85.375669

Maximum likelihood estimation

Log likelihood = -85.375669                     Number of obs   =         20
```

	Coef.	OPG Std. Err.	z	P>|z|	[95% Conf.	Interval]
/P	2.410602	.8768255	2.75	0.006	.6920557	4.129149
/lambda	.0770702	.0270771	2.85	0.004	.0240001	.1301404

◁

Parameter constraints

In examples 1 and 2, we were lucky. The two-parameter gamma density function is defined only when both λ and P are positive. However, `mlexp` does not know this; when maximizing the log-likelihood function, it will consider all real values for the parameters. Rather than relying on luck, we should instead reparameterize our model so that we avoid having to directly estimate parameters that are restricted.

For example, consider the parameter $\lambda > 0$, and suppose we define the new parameter $\theta = \ln(\lambda)$ so that $\lambda = \exp(\theta)$. With this parameterization, for any real value of θ that `mlexp` might try to use when evaluating the log-likelihood function, λ is guaranteed to be positive.

▷ Example 3: Classical normal regression

In the classical normal variant of linear regression (Goldberger 1991, chap. 19), we assume not only that $y_i = \mathbf{x}_i'\boldsymbol{\beta} + \epsilon_i$ but also that $\mathbf{x}_i$ is nonstochastic and that ϵ_i is distributed independently and identically normal with mean zero and variance σ^2. This is equivalent to assuming that

$$y_i|\mathbf{x}_i \sim N(\mathbf{x}_i'\boldsymbol{\beta}, \sigma^2)$$

Using the properties of the normal distribution, we could write the log likelihood for the ith observation as

$$\ln\ell_i = \ln\left\{\frac{1}{\sigma}\phi\left(\frac{y_i - \mathbf{x}_i'\boldsymbol{\beta}}{\sigma}\right)\right\}$$

where $\phi(\cdot)$ is the standard normal density function. In Stata, we use the three-argument version of the `normalden()` function and directly specify the conditional mean ($\mathbf{x}_i'\boldsymbol{\beta}$) and standard deviation ($\boldsymbol{\sigma}$) as the additional arguments.

The normal density function is defined only for $\sigma > 0$, so instead of estimating σ directly, we will instead estimate the unconstrained parameter θ and let $\sigma = \exp(\theta)$. For any real value of θ, this transformation ensures that $\sigma > 0$.

Using `auto.dta`, say that we want to fit the classical normal regression

$$\texttt{mpg}_i = \beta_0 + \beta_1 \texttt{weight}_i + \epsilon_i$$

We type

```
. use http://www.stata-press.com/data/r13/auto
(1978 Automobile Data)

. mlexp (ln(normalden(mpg, {b0} + {b1}*weight, exp({theta}))))
initial:       log likelihood =     -<inf>  (could not be evaluated)
feasible:      log likelihood = -882.02886
rescale:       log likelihood = -882.02886
rescale eq:    log likelihood = -274.09391
Iteration 0:   log likelihood = -274.09391  (not concave)
  (output omitted)
Iteration 13:  log likelihood = -195.38869

Maximum likelihood estimation

Log likelihood = -195.38869                     Number of obs   =         74
```

	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
/b0	39.44028	1.592043	24.77	0.000	36.31993	42.56063
/b1	-.0060087	.0005108	-11.76	0.000	-.0070099	-.0050075
/theta	1.221449	.0821995	14.86	0.000	1.060341	1.382557

To recover our estimate of σ, we can use `nlcom`:

```
. nlcom (sigma: exp(_b[/theta]))
       sigma:  exp(_b[/theta])
```

	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
sigma	3.392099	.2788288	12.17	0.000	2.845605	3.938594

◁

In the previous example, we named the unconstrained parameter θ `theta`. In actual practice, however, we would generally name that parameter `lnsigma` to indicate that it is $\ln(\sigma)$.

Two other parameter restrictions often appear in maximum likelihood estimation. Consider the correlation coefficient ρ. In general, it must be true that $-1 < \rho < 1$. Define the parameter $\eta = \tanh^{-1}(\rho)$, where $\tanh^{-1}(\cdot)$ is the hyperbolic arctangent function. Then $\rho = \tanh(\eta)$, and by the properties of the hyperbolic tangent function, for any real value of η, we will have $-1 < \rho < 1$. Stata has the built-in function `tanh()`, so recovering ρ from η is easy. In practice, instead of naming the unconstrained parameter `eta` in our likelihood expression, we would name it `atanhrho` to remind us that it is the hyperbolic arctangent of ρ.

Other parameters, such as those that represent probabilities or ratios of variances, are often restricted to be between 0 and 1. Say that we have the restriction $0 < \kappa < 1$. Consider the parameter $\psi = \ln\{\kappa/(1-\kappa)\}$. Then $\kappa = e^{\psi}/(1+e^{\psi})$ and for any real value of ψ, we have $0 < \kappa < 1$. The formula for κ in terms of ψ is known as the inverse logit transformation and is available as `invlogit()` in Stata. Thus in our likelihood expression, we would code `invlogit({logitk})` to map the unconstrained parameter `logitk` into the $(0,1)$ interval.

Specifying derivatives

By default, `mlexp` calculates derivatives of the log-likelihood function numerically using a sophisticated algorithm that produces accurate results. However, `mlexp` will fit your model more quickly (and even more accurately) if you specify analytic derivatives.

You specify derivatives by using substitutable expressions in much the same way as you specify the log-likelihood function. If you specify a linear combination in your log-likelihood function, then you supply a derivative with respect to that linear combination; `mlexp` then uses the chain rule to obtain the derivatives with respect to the individual parameters.

We will illustrate how to specify derivatives using the probit model for dichotomous outcomes. The log likelihood for the probit model is often written as

$$\ln\ell_i = \begin{cases} \ln\Phi(\mathbf{x}_i'\boldsymbol{\beta}) & y_i = 1 \\ \ln\Phi(-\mathbf{x}_i'\boldsymbol{\beta}) & y_i = 0 \end{cases}$$

using the fact that $1-\Phi(\mathbf{x}_i'\boldsymbol{\beta}) = \Phi(-\mathbf{x}_i'\boldsymbol{\beta})$, where $\Phi(\cdot)$ is the cumulative standard normal distribution function. If we use the trick suggested by Greene (2012, 691, fn. 7), we can simplify the log-likelihood function, making the derivative calculation easier. Let $q_i = 2y_i - 1$. Then we can write the log-likelihood function as

$$\ln\ell_i = \ln\Phi(q_i\mathbf{x}_i'\boldsymbol{\beta}) \tag{1}$$

and the first derivative as

$$\frac{\partial \ln\ell_i}{\partial\boldsymbol{\beta}} = \frac{q_i\phi(q_i\mathbf{x}_i'\boldsymbol{\beta})}{\Phi(q_i\mathbf{x}_i'\boldsymbol{\beta})}\mathbf{x}_i \tag{2}$$

▷ Example 4: Probit with one regressor

Say that we want to fit a probit model of `foreign` on `mpg` and a constant term. We have two parameters, so we will need to specify two derivatives; $\mathbf{x}_i$ consists of the ith observation on `mpg` and a `1` as the constant term. Because the term $q_i\mathbf{x}_i'\boldsymbol{\beta}$ will appear several times in our command, we create a macro to store it. We type

```
. use http://www.stata-press.com/data/r13/auto
(1978 Automobile Data)
. generate q = 2*foreign - 1
. global qxb "q*({b1}*mpg + {b0})"
. mlexp (ln(normal($qxb))), derivative(/b1 = q*normalden($qxb)/normal($qxb)*mpg)
> deriv(/b0 = q*normalden($qxb)/normal($qxb))
initial:       log likelihood = -51.292891
alternative:   log likelihood = -2017.3105
rescale:       log likelihood = -47.888213
rescale eq:    log likelihood = -46.343247
Iteration 0:   log likelihood = -46.343247
Iteration 1:   log likelihood = -39.268764
Iteration 2:   log likelihood = -39.258972
Iteration 3:   log likelihood = -39.258972
Maximum likelihood estimation
Log likelihood = -39.258972                     Number of obs   =         74
```

	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
/b1	.0960601	.0301523	3.19	0.001	.0369627	.1551575
/b0	-2.635268	.6841462	-3.85	0.000	-3.97617	-1.294366

◁

When you specify a linear combination of variables, you specify the derivative with respect to the linear combination. That way, if you change the variables that comprise the linear combination, you do not need to change the derivative at all. To see why this is the case, consider the function $f(\mathbf{x}_i'\boldsymbol{\beta})$, where $\mathbf{x}_i'\beta$ is a linear combination. Then, using the chain rule,

$$\frac{\partial f(\mathbf{x}_i'\boldsymbol{\beta})}{\partial \beta_j} = \frac{\partial f(\mathbf{x}_i'\boldsymbol{\beta})}{\partial \mathbf{x}_i'\boldsymbol{\beta}} \times \frac{\partial \mathbf{x}_i'\boldsymbol{\beta}}{\partial \beta_j} = \frac{\partial f(\mathbf{x}_i'\boldsymbol{\beta})}{\partial \mathbf{x}_i'\boldsymbol{\beta}} \times x_{ij}$$

Once the derivative with respect to the linear combination is known, `mlexp` can then multiply it by each of the variables in the linear combination to get the full set of derivatives with respect to the parameters needed to maximize the likelihood function. Moreover, the derivative with respect to the linear combination does not depend on the variables within the linear combination, so even if you change the variables in it, you will not need to modify the specification of the corresponding `derivative()` option.

▷ Example 5: Probit with a linear combination

Now let's fit a probit model of `foreign` on `mpg` and `gear_ratio`. We could specify the parameters and independent variables individually, but we will use a linear combination instead. First, note that

$$\frac{\partial \ln \ell_i}{\partial \mathbf{x}_i'\boldsymbol{\beta}} = \frac{q_i \phi(q_i \mathbf{x}_i'\boldsymbol{\beta})}{\Phi(q_i \mathbf{x}_i'\boldsymbol{\beta})}$$

We type

```
. use http://www.stata-press.com/data/r13/auto, clear
(1978 Automobile Data)
. generate q = 2*foreign - 1
. global qxb "q*({xb:} + {b0})"
. mlexp (ln(normal(q*({xb:mpg gear_ratio}+{b0})))),
> deriv(/xb = q*normalden($qxb)/normal($qxb))
> deriv(/b0 = q*normalden($qxb)/normal($qxb))
initial:       log likelihood = -51.292891
alternative:   log likelihood = -2556.2172
rescale:       log likelihood = -47.865271
rescale eq:    log likelihood = -46.658776
Iteration 0:   log likelihood = -46.658776
Iteration 1:   log likelihood = -22.541058
Iteration 2:   log likelihood = -21.467371
Iteration 3:   log likelihood = -21.454446
Iteration 4:   log likelihood = -21.454436
Iteration 5:   log likelihood = -21.454436
Maximum likelihood estimation
Log likelihood = -21.454436                     Number of obs   =         74
```

	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
/xb_mpg	-.0282433	.0464514	-0.61	0.543	-.1192864	.0627998
/xb_gear_r~o	3.699635	.8368276	4.42	0.000	2.059483	5.339787
/b0	-11.57588	2.337239	-4.95	0.000	-16.15678	-6.994972

We first redefined our global macro `$qxb` to contain the linear combination `xb` and a constant term `b0`. More importantly, we did not specify the variables in the linear combination just yet. Instead, we will use `$qxb` after we explicitly declare the variables in `xb` when we specify our model. To avoid making mistakes, you can declare the variables in a linear combination only once when you set up your model. If we had declared the variables when we defined `$qxb`, we would have received an error because, upon substituting for `$qxb`, we would have declared the variables multiple times in our call to `mlexp`.

◁

Stored results

mlexp stores the following in e():

Scalars

e(N)	number of observations
e(k)	number of parameters
e(k_aux)	number of ancillary parameters
e(k_eq)	number of equations in e(b)
e(k_eq_model)	number of equations in overall model test
e(df_m)	model degrees of freedom
e(ll)	log likelihood
e(N_clust)	number of clusters
e(rank)	rank of e(V)
e(ic)	number of iterations
e(rc)	return code
e(converged)	1 if converged, 0 otherwise

Macros

e(cmd)	mlexp
e(cmdline)	command as typed
e(lexp)	likelihood expression
e(wtype)	weight type
e(wexp)	weight expression
e(usrtitle)	user-specified title
e(usrtitle2)	user-specified secondary title
e(vce)	*vcetype* specified in vce()
e(vcetype)	title used to label Std. Err.
e(params)	names of parameters
e(hasderiv)	yes, if derivative() is specified
e(d_j)	derivative expression for parameter j
e(rhs)	contents of variables()
e(opt)	type of optimization
e(ml_method)	type of ml method
e(technique)	maximization technique
e(singularHmethod)	m-marquardt or hybrid; method used when Hessian is singular[1]
e(crittype)	optimization criterion[1]
e(properties)	b V
e(estat_cmd)	program used to implement estat
e(predict)	program used to implement predict
e(marginsnotok)	predictions disallowed by margins
e(marginsprop)	signals to the margins command

Matrices

e(b)	coefficient vector
e(ilog)	iteration log (up to 20 iterations)
e(init)	initial values
e(gradient)	gradient vector
e(V)	variance–covariance matrix of the estimators
e(V_modelbased)	model-based variance

Functions

e(sample)	marks estimation sample

1. Type ereturn list, all to view these results; see [P] **return**.

Methods and formulas

Optimization is carried out using `moptimize()`; see [M-5] **moptimize( )**.

References

Goldberger, A. S. 1991. *A Course in Econometrics*. Cambridge, MA: Harvard University Press.

Gould, W. W., J. S. Pitblado, and B. P. Poi. 2010. *Maximum Likelihood Estimation with Stata*. 4th ed. College Station, TX: Stata Press.

Greene, W. H. 2012. *Econometric Analysis*. 7th ed. Upper Saddle River, NJ: Prentice Hall.

Also see

[R] **mlexp postestimation** — Postestimation tools for mlexp

[R] **gmm** — Generalized method of moments estimation

[R] **maximize** — Details of iterative maximization

[R] **ml** — Maximum likelihood estimation

[R] **nl** — Nonlinear least-squares estimation

[R] **nlsur** — Estimation of nonlinear systems of equations