Title

stata.com

mlexp — Maximum likelihood estimation of user-specified expressions

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

Description

`mlexp` performs maximum likelihood estimation of models that satisfy the linear-form restrictions, that is, models for which you can write the log likelihood for an individual observation and for which the overall log likelihood is 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`.

Quick start

Linear regression of `y` on `x1` and `x2`

```
mlexp (ln(normalden(y, {xb: x1 x2 _cons}, {sigma})))
```

Same as above

```
mlexp (ln(normalden(y, {b0} + {b1}*x1 + {b2}*x2, {sigma})))
```

As above, and set initial values for `b2` and `sigma`

```
mlexp (ln(normalden(y, {b0} + {b1}*x1 + {b2=0.5}*x2, {sigma=3})))
```

Constrain `sigma` and the coefficient on `x1` to be positive

```
mlexp (ln(normalden(y, {b0} + exp({lnb1})*x1 + {b2}*x2, exp({lnsigma}))))
nlcom (b1: exp(_b[lnb1:_cons])) (sigma: exp(_b[lnsigma:_cons]))
```

Omit observations with missing values for `y`, `x1`, or `x2`

```
mlexp (ln(normalden(y, {xb: x1 x2 _cons}, {sigma}))), variables(y x1 x2)
```

Menu

Statistics > Other > Maximum likelihood estimation of expression

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
`constraints(`*numlist*`)`	apply specified linear constraints
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 columns and column formats, row spacing, line width, display of omitted variables and base and empty cells, and factor-variable labeling
Maximization	
maximize_options	control the maximization process; seldom used
`debug`	display debug output
`collinear`	keep collinear variables
`coeflegend`	display legend instead of statistics

lexp and *dexp* may contain factor variables and time-series operators; see **[U] 11.4.3 Factor variables** and **[U] 11.4.4 Time-series varlists**.

`bootstrap`, `by`, `collect`, `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**.

`vce()` and weights are not allowed with the `svy` prefix; see [SVY] **svy**.

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

`debug`, `collinear`, and `coeflegend` do not appear in the dialog box.

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

lexp and *dexp* are substitutable expressions, Stata expressions that also contain parameters to be estimated. The parameters are enclosed in curly braces and must satisfy the naming requirements for variables; `{beta}` is an example of a parameter. The notation {*lc*:*varlist*} is allowed for linear combinations of multiple covariates and their parameters. For example, `{xb: mpg price turn _cons}` defines a linear combination of the variables `mpg`, `price`, `turn`, and `_cons` (the constant term). See *Substitutable expressions* under *Remarks and examples* below.

Options

Model

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

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

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

or equivalently

```
matrix define initval = (1.23, 4.57)
mlexp ..., from(initval) ...
```

Initial values declared in the `from()` 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.

`constraints(`*numlist*`)`; see [R] **Estimation options**.

Derivatives

`derivative(`/*name* = *dexp*`)` specifies the derivative of the observation-level log-likelihood function with respect to parameter *name*. If you wish to specify analytic derivatives, you must specify `derivative()` for each parameter in your model.

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* for examples.

If you do not specify the `derivative()` option, `mlexp` calculates derivatives numerically.

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: `noci`, `nopvalues`, `noomitted`, `vsquish`, `noemptycells`, `baselevels`, `allbaselevels`, `nofvlabel`, `fvwrap(#)`, `fvwrapon(`*style*`)`, `cformat(%`*fmt*`)`, `pformat(%`*fmt*`)`, `sformat(%`*fmt*`)`, and `nolstretch`; 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 options are available with mlexp but are not shown in the dialog box:

debug specifies that differences between the numerically computed gradient and the gradient computed from your derivative expression are reported at each iteration. This option is only allowed with the derivative() option.

collinear, coeflegend; see [R] **Estimation options**.

Remarks and examples

stata.com

Remarks are presented under the following headings:

Introduction
Substitutable expressions
Parameter interpretation using margins
Parameter constraints
Specifying derivatives

Introduction

mlexp performs maximum likelihood estimation of models that satisfy the linear-form restrictions, that is, models for which you can write the log likelihood for a single observation and for which the overall log likelihood is 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 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. mlexp allows you to avoid the programming requirements of ml. 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

Substitutable expressions allow you to distinguish between variables and parameters. There are three rules to follow when defining substitutable expressions:

1. Parameters of the model are bound in curly braces: {b0}, {param}, etc. Parameter names must follow the same conventions as variable names; see [U] **11.3 Naming conventions**.
2. Initial values for parameters are given by including an equal sign and the initial value inside the curly braces: {b0=1}, {param=3.571}, etc.
3. Linear combinations of variables can be included using the notation {*lc*:*varlist*}: {xb: mpg price weight _cons}, {score: w x z}, etc. Parameters of linear combinations are initialized to zero.

Substitutable expressions can include any mathematical expression involving scalars and variables. See [U] **13.2 Operators** and [U] **13.3 Functions** for more information on expressions.

▷ Example 1: The gamma density function

In an extract of the March 2014 Current Population Survey downloaded from Integrated Public Use Microdata Series (IPUMS), `wage` contains the wage and salary income earned in tens of thousands of dollars per year for working individuals; see King et al. (2010). We want to model `wage` using the two-parameter gamma distribution with shape parameter α and rate parameter β. The density function for $y > 0$ is

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

so that the log likelihood for the ith observation is

$$\ln(\ell_i) = \alpha \ln(\beta) - \ln\{\Gamma(\alpha)\} + (\alpha - 1) \ln(y_i) - \beta y_i$$

To fit `wage` to the two-parameter gamma distribution, we let `a` be the parameter name that conforms to Stata naming conventions for α and `b` be the parameter name for β. We enclose both in `{}` in our substitutable expression for the log-likelihood function:

```
. use https://www.stata-press.com/data/r17/cpswage
. mlexp ({a}*ln({b}) - lngamma({a}) + ({a}-1)*ln(wage)  - {b}*wage)
initial:       log likelihood =     -<inf>  (could not be evaluated)
feasible:      log likelihood = -239367.36
rescale:       log likelihood = -228037.02
rescale eq:    log likelihood = -163560.64
Iteration 0:   log likelihood = -163560.64
Iteration 1:   log likelihood = -162820.41
Iteration 2:   log likelihood = -162808.55
Iteration 3:   log likelihood = -162808.55
Maximum likelihood estimation
Log likelihood = -162808.55                             Number of obs = 64,748
```

	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
/a	1.097287	.0054134	202.70	0.000	1.086677	1.107897
/b	.2406718	.0014917	161.34	0.000	.2377482	.2435954

Because we did not specify initial values, `mlexp` initialized α and β to be 0. When both parameters are 0, 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)`. When this occurs, `mlexp` uses a search routine to find alternative initial values that do allow the log-likelihood function to be calculated.

We now initialize `a` to 1 and `b` to 0.1 the first time that we type them within the substitutable expression:

```
. mlexp ({a=1}*ln({b=.1}) - lngamma({a}) + ({a}-1)*ln(wage)  - {b}*wage)
initial:       log likelihood = -178608.12
rescale:       log likelihood = -178608.12
rescale eq:    log likelihood = -173389.94
Iteration 0:   log likelihood = -173389.94
Iteration 1:   log likelihood = -163081.69
Iteration 2:   log likelihood = -162813.54
Iteration 3:   log likelihood = -162808.55
Iteration 4:   log likelihood = -162808.55
Maximum likelihood estimation
Log likelihood = -162808.55                     Number of obs = 64,748
```

	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
/a	1.097287	.0054134	202.70	0.000	1.086677	1.107897
/b	.2406719	.0014917	161.34	0.000	.2377483	.2435955

Even when `mlexp` can find alternative initial values, specifying your own values facilitates optimization. ◁

By default, initial values of parameters are set to zero, and `mlexp` performs an iterative search for optimum starting values before beginning maximization. If `mlexp` cannot find initial parameter values for which it can calculate the log-likelihood function, it will exit with an error message. Restricting the sample or specifying starting values solves this problem.

Use the `variables()` option, an *if* qualifier, or an *in* qualifier to restrict the sample. You can also specify initial values by using the `from()` option or within the substitutable expression by including an equal sign and the initial value after the parameter. If you specify initial values by using `from()`, they override whatever initial values are given within the substitutable expression. 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.

▷ Example 2: Linear combinations of covariates

We frequently want to model the parameters as linear combinations of variables. Continuing example 1, we see that the mean of the two-parameter gamma distribution is $\mathbf{E}(y) = \alpha/\beta$. By letting $\alpha = a_1\texttt{age}+a_0$, we model the mean of `wage` conditional on `age` as $\mathbf{E}(\texttt{wage}|\texttt{age}) = (a_1\texttt{age}+a_0)/\beta$. Below, we specify `{a: age _cons}` to model `a` as a linear combination of `age` and a constant term.

```
. mlexp ({a:age _cons}*ln({b=.1})-lngamma({a:})+({a:}-1)*ln(wage)-{b}*wage)
initial:       log likelihood =     -<inf>  (could not be evaluated)
feasible:      log likelihood = -239367.36
rescale:       log likelihood = -228037.02
rescale eq:    log likelihood = -163560.64
Iteration 0:   log likelihood = -163560.64
Iteration 1:   log likelihood = -160123.65
Iteration 2:   log likelihood = -159838.42
Iteration 3:   log likelihood = -159838.38
Iteration 4:   log likelihood = -159838.38
Maximum likelihood estimation
Log likelihood = -159838.38                     Number of obs = 64,748
```

	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
a						
age	.0194396	.0002689	72.31	0.000	.0189126	.0199665
_cons	.4108355	.0096791	42.45	0.000	.3918647	.4298062
/b	.2667954	.0016423	162.45	0.000	.2635765	.2700142

The notation `{a: age _cons}` indicates to `mlexp` that you want a linear combination of the variable `age` and a constant term. We named the linear combination `a`, so `mlexp` names the parameters `a:age` and `a:_cons`, respectively.

Once you have declared a linear combination, you can subsequently refer to the linear combination by specifying its name and a colon inside curly braces, as we did in 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.

◁

▷ Example 3: Linear combinations of factor variables

Factor variables and time-series operated variables can be included in a *varlist* defining a linear combination. Continuing example 2, we want to allow different intercepts for males and females in α, and we want different β parameters for males and females. We implement this model below by including `ibn.female` in each linear combination using factor-variable notation.

```
. mlexp ({a:age ibn.female}*ln({b:ibn.female}) - lngamma({a:}) +
> ({a:}-1)*ln(wage)  - {b:}*wage)
initial:       log likelihood =     -<inf>  (could not be evaluated)
feasible:      log likelihood = -239367.36
rescale:       log likelihood = -228037.02
rescale eq:    log likelihood = -163560.64
Iteration 0:   log likelihood = -163560.64
Iteration 1:   log likelihood = -158999.51
Iteration 2:   log likelihood = -158135.16
Iteration 3:   log likelihood = -158130.94
Iteration 4:   log likelihood = -158130.93
Maximum likelihood estimation
Log likelihood = -158130.93                           Number of obs = 64,748
```

	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
a						
age	.020543	.0002781	73.88	0.000	.019998	.0210881
female						
Male	.3917619	.0111697	35.07	0.000	.3698696	.4136542
Female	.4475977	.0116285	38.49	0.000	.4248063	.4703892
b						
female						
Male	.2279862	.0018677	122.07	0.000	.2243256	.2316468
Female	.3600628	.0030156	119.40	0.000	.3541523	.3659733

◁

Parameter interpretation using margins

The `margins` command can be used after `mlexp` to estimate the effect of a covariate from a set of parameter estimates. The estimated covariate effects can be conditional on the other covariates or population-averaged effects that average out the other covariates.

▷ Example 4

Continuing example 3, we use `margins` to estimate the average wage if everyone in the sample were male and if everyone in the sample were female. The `expression()` option is used to specify the formula for the mean. Wages, measured in tens of thousands of dollars, have a gamma distribution, so the mean is a ratio of the α and β parameters. The linear prediction is specified in the expression with `xb()`.

```
. margins i.female, expression(xb(a)/xb(b))
Predictive margins                                    Number of obs = 64,748
Model VCE: OIM
Expression: xb(a)/xb(b)
```

	Margin	Delta-method std. err.	z	P>|z|	[95% conf.	interval]
female						
Male	5.452281	.0267573	203.77	0.000	5.399838	5.504724
Female	3.607372	.0178778	201.78	0.000	3.572332	3.642412

When everyone is male, the estimated average wage is 5.45; when everyone is female, the estimated average wage is 3.61.

◁

Parameter constraints

▷ Example 5

In examples 1, 2, and 3, we were lucky. The two-parameter gamma density function is defined only when both α and β are positive. However, `mlexp` does not know this; when maximizing the log-likelihood function, it will consider all real values for the parameters. Also recall from above, `mlexp` will exit with an error message if it cannot find parameter values that produce nonmissing values for the likelihood for each sample observation.

We could reparameterize our model so that we avoid having to directly estimate parameters that are restricted. For example, consider the parameter $\alpha > 0$, and suppose we define the new parameter $\theta = \ln(\alpha)$ so that $\alpha = \exp(\theta)$. With this parameterization, for any real value of θ that `mlexp` might try to use when evaluating the log-likelihood function, α is guaranteed to be positive. Below we apply this parameterization to the model in example 1.

```
. mlexp (exp({lna})*{lnb} - lngamma(exp({lna}))
> + (exp({lna})-1)*ln(wage) - exp({lnb})*wage)

initial:       log likelihood = -295203.41
alternative:   log likelihood = -249215.67
rescale:       log likelihood = -230376.58
rescale eq:    log likelihood = -166588.81
Iteration 0:   log likelihood = -166588.81
Iteration 1:   log likelihood = -162845.95
Iteration 2:   log likelihood = -162808.56
Iteration 3:   log likelihood = -162808.55
Iteration 4:   log likelihood = -162808.55

Maximum likelihood estimation

Log likelihood = -162808.55                     Number of obs = 64,748

------------------------------------------------------------------------------
             |      Coefficient  Std. err.      z    P>|z|     [95% conf. interval]
-------------+----------------------------------------------------------------
        /lna |      .0928409   .0049334    18.82   0.000     .0831716    .1025103
        /lnb |     -1.424321   .0061979  -229.81   0.000    -1.436469   -1.412173
------------------------------------------------------------------------------
```

We can use `nlcom` to obtain the back-transformed parameter estimates, but first we need to know how to refer to the parameters estimated by `mlexp`. We can replay the results and request the coefficient legend.

```
. mlexp, coeflegend

Maximum likelihood estimation

Log likelihood = -162808.55                     Number of obs = 64,748

------------------------------------------------------------------------------
             |      Coefficient  Legend
-------------+----------------------------------------------------------------
        /lna |      .0928409  _b[lna:_cons]
        /lnb |     -1.424321  _b[lnb:_cons]
------------------------------------------------------------------------------
```

Now, we see that we can refer to _b[lna:_cons] and _b[lnb:_cons] in nlcom.

```
. nlcom (a: exp(_b[lna:_cons])) (b: exp(_b[lnb:_cons]))
          a: exp(_b[lna:_cons])
          b: exp(_b[lnb:_cons])
```

	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
a	1.097287	.0054134	202.70	0.000	1.086677	1.107897
b	.2406718	.0014917	161.34	0.000	.2377482	.2435954

The optimal value of the log-likelihood function and the back-transformed parameter estimates match those reported in example 1.

However, when you impose nonlinear constraints on linear combinations of covariates, you change the model. These nonlinear constraints change the model because they are not invertible functions of the original parameters. For example, we can use the exponential function to ensure $\alpha > 0$ and $\beta > 0$ in the model from example 3.

```
. mlexp (exp({a:age ibn.female})*{b:ibn.female} - lngamma(exp({a:}))
> + (exp({a:})-1)*ln(wage) - exp({b:})*wage)

initial:       log likelihood = -295203.41
alternative:   log likelihood = -249215.67
rescale:       log likelihood = -230376.58
rescale eq:    log likelihood = -166588.81
Iteration 0:   log likelihood = -166588.81
Iteration 1:   log likelihood = -160273.12
Iteration 2:   log likelihood = -158868.39
Iteration 3:   log likelihood = -158864.02
Iteration 4:   log likelihood = -158864.02

Maximum likelihood estimation

Log likelihood = -158864.02                     Number of obs = 64,748
```

	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
a						
age	.0126713	.0001785	70.98	0.000	.0123214	.0130212
female						
Male	-.3470203	.009978	-34.78	0.000	-.3665769	-.3274638
Female	-.3177398	.0103376	-30.74	0.000	-.338001	-.2974786
b						
female						
Male	-1.502222	.0085431	-175.84	0.000	-1.518966	-1.485478
Female	-1.059358	.0087602	-120.93	0.000	-1.076528	-1.042189

The value of the log-likelihood function is now −158,864.02, which is less than the value −158,130.93, which was reported in example 3. The constrained parameterization that keeps $\alpha > 0$ and $\beta > 0$ has changed the model, and the optimal value of the log-likelihood function in the constrained model is less than the optimal value in the unconstrained model.

Because neither the unconstrained nor the constrained parameterization generates missing predicted values for α or β, we can choose between them. However, if any of the predicted values for α or β produce missing values for the log-likelihood function using the unconstrained parameterization, we could not compute the log-likelihood function for the unconstrained parameterization, and we would have to use a constrained parameterization.

◁

Specifying derivatives

By default, `mlexp` calculates derivatives of the log-likelihood function numerically using an 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 function 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 (2018, 742, fn. 16), 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

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

and the first derivative as

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

▷ Example 6: Probit with a linear combination

Now, let's fit a probit model of the indicator for whether an individual is below the official poverty level `offpov` on `age`, `female`, and a constant. We could specify the parameters and independent variables individually, but we will use a linear combination instead. First, note that

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

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 in the linear combination, you do not need to change the derivative. 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, we see that

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

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.

We type

```
. generate int q = 2*offpov - 1
. mlexp (ln(normal(q*({xb:age i.female _cons})))),
> deriv(/xb = q*normalden(q*{xb:})/normal(q*{xb:}))
initial:       log likelihood = -44879.894
alternative:   log likelihood = -27407.718
rescale:       log likelihood = -17888.147
Iteration 0:   log likelihood = -17888.147
Iteration 1:   log likelihood =  -15805.22
Iteration 2:   log likelihood = -15427.598
Iteration 3:   log likelihood =  -15427.04
Iteration 4:   log likelihood =  -15427.04
Maximum likelihood estimation
Log likelihood = -15427.04                              Number of obs = 64,748
```

	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
xb						
age	-.0179139	.0005852	-30.61	0.000	-.0190608	-.016767
female						
Female	.1514014	.0154565	9.80	0.000	.1211072	.1816955
_cons	-.8773462	.0244956	-35.82	0.000	-.9253567	-.8293357

After defining `q`, we specified the log-likelihood function using `q` and the linear combination `xb`. We also use `xb` in specifying the derivative.

◁

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(clustvar)	name of cluster variable
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(marginsok)	predictions allowed by margins
e(marginsnotok)	predictions disallowed by margins
e(marginsprop)	signals to the margins command
e(asbalanced)	factor variables fvset as asbalanced
e(asobserved)	factor variables fvset as asobserved

Matrices

e(b)	coefficient vector
e(Cns)	constraints matrix
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**.

In addition to the above, the following is stored in `r()`:

Matrices

`r(table)`	matrix containing the coefficients with their standard errors, test statistics, p-values, and confidence intervals

Note that results stored in `r()` are updated when the command is replayed and will be replaced when any r-class command is run after the estimation command.

Methods and formulas

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

References

Drukker, D. M. 2015a. Efficiency comparisons by Monte Carlo simulation. *The Stata Blog: Not Elsewhere Classified.* http://blog.stata.com/2015/10/13/efficiency-comparisons-by-monte-carlo-simulation/.

——. 2015b. Maximum likelihood estimation by mlexp: A chi-squared example. *The Stata Blog: Not Elsewhere Classified.* http://blog.stata.com/2015/10/08/maximum-likelihood-estimation-by-mlexp-a-chi-squared-example/.

——. 2015c. Understanding the generalized method of moments (GMM): A simple example. *The Stata Blog: Not Elsewhere Classified.* http://blog.stata.com/2015/12/03/understanding-the-generalized-method-of-moments-gmm-a-simple-example/.

Greene, W. H. 2018. *Econometric Analysis*. 8th ed. New York: Pearson.

King, M., S. Ruggles, J. T. Alexander, S. Flood, K. Genadek, M. B. Schroeder, B. Trampe, and R. Vick. 2010. Integrated Public Use Microdata Series, Current Population Survey: Version 3.0 [Machine-readable database]. Minneapolis, MN: Minnesota Population Center [producer and distributor].

Lindsey, C. 2015a. Probit model with sample selection by mlexp. *The Stata Blog: Not Elsewhere Classified.* http://blog.stata.com/2015/10/22/probit-model-with-sample-selection-by-mlexp/.

——. 2015b. Using mlexp to estimate endogenous treatment effects in a heteroskedastic probit model. *The Stata Blog: Not Elsewhere Classified.* http://blog.stata.com/2015/12/10/using-mlexp-to-estimate-endogenous-treatment-effects-in-a-heteroskedastic-probit-model/.

——. 2015c. Using mlexp to estimate endogenous treatment effects in a probit model. *The Stata Blog: Not Elsewhere Classified.* http://blog.stata.com/2015/11/05/using-mlexp-to-estimate-endogenous-treatment-effects-in-a-probit-model/.

Lindsey, C., and E. Pinzon. 2016. Multiple equation models: Estimation and marginal effects using mlexp. *The Stata Blog: Not Elsewhere Classified.* http://blog.stata.com/2016/06/14/multiple-equation-models-estimation-and-marginal-effects-using-mlexp/.

Rajbhandari, A. 2015. Estimating parameters by maximum likelihood and method of moments using mlexp and gmm. *The Stata Blog: Not Elsewhere Classified.* http://blog.stata.com/2015/10/15/estimating-parameters-by-maximum-likelihood-and-method-of-moments-using-mlexp-and-gmm/.

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

[U] **20 Estimation and postestimation commands**