Maximize — Details of iterative maximization

Description

All Stata commands maximize likelihood functions using `moptimize()` and `optimize()`; see *Methods and formulas* below. Commands use the Newton–Raphson method with step halving and special fix-ups when they encounter nonconcave regions of the likelihood. For details, see [M-5] **moptimize( )** and [M-5] **optimize( )**. For more information about programming maximum likelihood estimators in ado-files and Mata, see [R] **ml** and Pitblado, Poi, and Gould (2024).

Syntax

mle_cmd . . . [, *options*]

options	Description
`difficult`	use a different stepping algorithm in nonconcave regions
`technique(`*algorithm_spec*`)`	maximization technique
`iterate(#)`	perform maximum of # iterations; default is `iterate(300)`
[`no`]`log`	display an iteration log of the log likelihood; typically, the default
`trace`	display current parameter vector in iteration log
`gradient`	display current gradient vector in iteration log
`showstep`	report steps within an iteration in iteration log
`hessian`	display current negative Hessian matrix in iteration log
`showtolerance`	report the calculated result that is compared to the effective convergence criterion
`tolerance(#)`	tolerance for the coefficient vector; see *Options* for the defaults
`ltolerance(#)`	tolerance for the log likelihood; see *Options* for the defaults
`nrtolerance(#)`	tolerance for the scaled gradient; see *Options* for the defaults
`qtolerance(#)`	when specified with algorithms `bhhh`, `dfp`, or `bfgs`, the **q** − **H** matrix is used as the final check for convergence rather than `nrtolerance()` and the **H** matrix; seldom used
`nonrtolerance`	ignore the `nrtolerance()` option
`from(`*init_specs*`)`	initial values for the coefficients

algorithm_spec is

algorithm [# [*algorithm* [#]]. . .]

algorithm is { `nr` | `bhhh` | `dfp` | `bfgs` }

init_specs is one of

matname [, `skip` `copy`]

{ [*eqname* :] *name* = # | / *eqname* = # } [. . .]

[# . . .], `copy`

Maximization options

`difficult` specifies that the likelihood function is likely to be difficult to maximize because of nonconcave regions. When the message "not concave" appears repeatedly, `ml`'s standard stepping algorithm may not be working well. `difficult` specifies that a different stepping algorithm be used in nonconcave regions. There is no guarantee that `difficult` will work better than the default; sometimes it is better and sometimes it is worse. You should use the `difficult` option only when the default stepper declares convergence and the last iteration is "not concave" or when the default stepper is repeatedly issuing "not concave" messages and producing only tiny improvements in the log likelihood.

`technique(`*algorithm_spec*`)` specifies how the likelihood function is to be maximized. The following algorithms are allowed. For details, see Pitblado, Poi, and Gould (2024).

`technique(nr)` specifies Stata's modified Newton–Raphson (NR) algorithm.

`technique(bhhh)` specifies the Berndt–Hall–Hall–Hausman (BHHH) algorithm.

`technique(dfp)` specifies the Davidon–Fletcher–Powell (DFP) algorithm.

`technique(bfgs)` specifies the Broyden–Fletcher–Goldfarb–Shanno (BFGS) algorithm.

The default is `technique(nr)`.

You can switch between algorithms by specifying more than one in the `technique()` option. By default, an algorithm is used for five iterations before switching to the next algorithm. To specify a different number of iterations, include the number after the technique in the option. For example, specifying `technique(bhhh 10 nr 1000)` requests that `ml` perform 10 iterations with the BHHH algorithm followed by 1000 iterations with the NR algorithm, and then switch back to BHHH for 10 iterations, and so on. The process continues until convergence or until the maximum number of iterations is reached.

`iterate(#)` specifies the maximum number of iterations. When the number of iterations equals `iterate()`, the optimizer stops and presents the current results. If convergence is declared before this threshold is reached, it will stop when convergence is declared. Specifying `iterate(0)` is useful for viewing results evaluated at the initial value of the coefficient vector. Specifying `iterate(0)` and `from()` together allows you to view results evaluated at a specified coefficient vector; however, not all commands allow the `from()` option. The default value of `iterate(#)` for both estimators programmed internally and estimators programmed with `ml` is the number set using `set maxiter`, which is 300 by default.

`log` and `nolog` specify whether an iteration log showing the progress of the log likelihood is to be displayed. For most commands, the log is displayed by default, and `nolog` suppresses it; see `set iterlog` in [R] ***set iter***. For a few commands (such as the `svy` maximum likelihood estimators), you must specify `log` to see the log.

`trace` adds to the iteration log a display of the current parameter vector.

`gradient` adds to the iteration log a display of the current gradient vector.

`showstep` adds to the iteration log a report on the steps within an iteration. This option was added so that developers at StataCorp could view the stepping when they were improving the `ml` optimizer code. At this point, it mainly provides entertainment.

`hessian` adds to the iteration log a display of the current negative Hessian matrix.

showtolerance adds to the iteration log the calculated value that is compared with the effective convergence criterion at the end of each iteration. Until convergence is achieved, the smallest calculated value is reported.

shownrtolerance is a synonym of showtolerance.

Below, we describe the three convergence tolerances. Convergence is declared when the nrtolerance() criterion is met and either the tolerance() or the ltolerance() criterion is also met.

tolerance(#) specifies the tolerance for the coefficient vector. When the relative change in the coefficient vector from one iteration to the next is less than or equal to tolerance(), the tolerance() convergence criterion is satisfied.

tolerance(1e-4) is the default for estimators programmed with ml.

tolerance(1e-6) is the default.

ltolerance(#) specifies the tolerance for the log likelihood. When the relative change in the log likelihood from one iteration to the next is less than or equal to ltolerance(), the ltolerance() convergence is satisfied.

ltolerance(0) is the default for estimators programmed with ml.

ltolerance(1e-7) is the default.

nrtolerance(#) specifies the tolerance for the scaled gradient. Convergence is declared when $\mathbf{g}\mathbf{H}^{-1}\mathbf{g}' < \texttt{nrtolerance()}$. The default is nrtolerance(1e-5).

qtolerance(#) when specified with algorithms bhhh, dfp, or bfgs uses the $\mathbf{q} - \mathbf{H}$ matrix as the final check for convergence rather than nrtolerance() and the $\mathbf{H}$ matrix.

Beginning with Stata 12, by default, Stata now computes the $\mathbf{H}$ matrix when the $\mathbf{q}-\mathbf{H}$ matrix passes the convergence tolerance, and Stata requires that $\mathbf{H}$ be concave and pass the nrtolerance() criterion before concluding convergence has occurred.

qtolerance() provides a way for the user to obtain Stata's earlier behavior.

nonrtolerance specifies that the default nrtolerance() criterion be turned off.

---

from() specifies initial values for the coefficients. Not all estimators in Stata support this option. You can specify the initial values in one of three ways: by specifying the name of a vector containing the initial values (for example, from(b0), where b0 is a properly labeled vector); by specifying coefficient names with the values (for example, from(age=2.1 /sigma=7.4)); or by specifying a list of values (for example, from(2.17.4, copy)). from() is intended for use when doing bootstraps (see [R] **bootstrap**) and in other special situations (for example, with iterate(0)). Even when the values specified in from() are close to the values that maximize the likelihood, only a few iterations may be saved. Poor values in from() may lead to convergence problems.

skip specifies that any parameters found in the specified initialization vector that are not also found in the model be ignored. The default action is to issue an error message.

copy specifies that the list of values or the initialization vector be copied into the initial-value vector by position rather than by name.

Remarks and examples

Only in rare circumstances would you ever need to specify any of these options, except `nolog`. The `nolog` option is useful for reducing the amount of output appearing in log files; also see `set iterlog` in [R] ***set iter***.

The following is an example of an iteration log:

```
Iteration 0:  Log likelihood = -3791.0251
Iteration 1:  Log likelihood =  -3761.738
Iteration 2:  Log likelihood = -3758.0632  (not concave)
Iteration 3:  Log likelihood = -3758.0447
Iteration 4:  Log likelihood = -3757.5861
Iteration 5:  Log likelihood =  -3757.474
Iteration 6:  Log likelihood = -3757.4613
Iteration 7:  Log likelihood = -3757.4606
Iteration 8:  Log likelihood = -3757.4606
  (table of results omitted$ $)
```

At iteration 8, the model converged. The message "not concave" at the second iteration is notable. This example was produced using the `heckman` command; its likelihood is not globally concave, so it is not surprising that this message sometimes appears. The other message that is occasionally seen is "backed up". Neither of these messages should be of any concern unless they appear at the final iteration.

If a "not concave" message appears at the last step, there are two possibilities. One is that the result is valid, but there is collinearity in the model that the command did not otherwise catch. Stata checks for obvious collinearity among the independent variables before performing the maximization, but strange collinearities or near collinearities can sometimes arise between coefficients and ancillary parameters. The second, more likely cause for a "not concave" message at the final step is that the optimizer entered a flat region of the likelihood and prematurely declared convergence.

If a "backed up" message appears at the last step, there are also two possibilities. One is that Stata found a perfect maximum and could not step to a better point; if this is the case, all is fine, but this is a highly unlikely occurrence. The second is that the optimizer worked itself into a bad concave spot where the computed gradient and Hessian gave a bad direction for stepping.

If either of these messages appears at the last step, perform the maximization again with the `gradient` option. If the gradient goes to zero, the optimizer has found a maximum that may not be unique but is a maximum. From the standpoint of maximum likelihood estimation, this is a valid result. If the gradient is not zero, it is not a valid result, and you should try tightening up the convergence criterion, or try `ltol(0) tol(1e-7)` to see if the optimizer can work its way out of the bad region.

If you get repeated "not concave" steps with little progress being made at each step, try specifying the `difficult` option. Sometimes `difficult` works wonderfully, reducing the number of iterations and producing convergence at a good (that is, concave) point. Other times, `difficult` works poorly, taking much longer to converge than the default stepper.

Stored results

Maximum likelihood estimators store the following in e():

Scalars

e(N)	number of observations	always stored
e(k)	number of parameters	always stored
e(k_eq)	number of equations in e(b)	usually stored
e(k_eq_model)	number of equations in overall model test	usually stored
e(k_dv)	number of dependent variables	usually stored
e(df_m)	model degrees of freedom	always stored
e(r2_p)	pseudo-R^2	sometimes stored
e(ll)	log likelihood	always stored
e(ll_0)	log likelihood, constant-only model	stored when constant-only model is fit
e(N_clust)	number of clusters	stored when vce(cluster *clustvar*) is specified; see **[U] 20.22 Obtaining robust variance estimates**
e(chi2)	χ^2	usually stored
e(p)	p-value for model test	usually stored
e(rank)	rank of e(V)	always stored
e(rank0)	rank of e(V) for constant-only model	stored when constant-only model is fit
e(ic)	number of iterations	usually stored
e(rc)	return code	usually stored
e(converged)	1 if converged, 0 otherwise	usually stored

Macros

e(cmd)	name of command	always stored
e(cmdline)	command as typed	always stored
e(depvar)	names of dependent variables	always stored
e(wtype)	weight type	stored when weights are specified or implied
e(wexp)	weight expression	stored when weights are specified or implied
e(title)	title in estimation output	usually stored by commands using ml
e(clustvar)	name of cluster variable	stored when vce(cluster *clustvar*) is specified; see **[U] 20.22 Obtaining robust variance estimates**
e(chi2type)	Wald or LR; type of model χ^2 test	usually stored
e(vce)	*vcetype* specified in vce()	stored when command allows vce()
e(vcetype)	title used to label Std. err.	sometimes stored
e(opt)	type of optimization	always stored
e(which)	max or min; whether optimizer is to perform maximization or minimization	always stored
e(ml_method)	type of ml method	always stored by commands using ml
e(user)	name of likelihood-evaluator program	always stored
e(technique)	from technique() option	sometimes stored
e(singularHmethod)	m-marquardt or hybrid; method used when Hessian is singular	sometimes stored[1]
e(crittype)	optimization criterion	always stored[1]
e(properties)	estimator properties	always stored
e(predict)	program used to implement predict	usually stored

Macros, *continued*		
e(deriv_useminbound)	off or on; whether minimum bounds are used for step sizes in numerical derivative computations	always stored by ml model
Matrices		
e(b)	coefficient vector	always stored
e(Cns)	constraints matrix	sometimes stored
e(ilog)	iteration log (up to 20 iterations)	usually stored
e(gradient)	gradient vector	usually stored
e(V)	variance–covariance matrix of the estimators	always stored
e(V_modelbased)	model-based variance	only stored when e(V) is robust, cluster–robust, bootstrap, or jackknife variance
e(deriv_minbound)	minimum values for step sizes	only stored when e(deriv_useminbound) is on
Functions		
e(sample)	marks estimation sample	always stored

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

See *Stored results* in the manual entry for any maximum likelihood estimator for a list of returned results.

Methods and formulas

Optimization is currently performed by moptimize() and optimize(), with the former implemented in terms of the latter; see [M-5] **moptimize( )** and [M-5] **optimize( )**. Some estimators use moptimize() and optimize() directly, and others use the ml ado-file interface to moptimize().

Prior to Stata 11, Stata had three separate optimization engines: an internal one used by estimation commands implemented in C code; ml implemented in ado-code separately from moptimize() and used by most estimators; and moptimize() and optimize() used by a few recently written estimators. These days, the internal optimizer and the old version of ml are used only under version control. In addition, arch and arima (see [TS] **arch** and [TS] **arima**) are currently implemented using the old ml.

Let L_1 be the log likelihood of the full model (that is, the log-likelihood value shown on the output), and let L_0 be the log likelihood of the "constant-only" model. The likelihood-ratio χ^2 model test is defined as $2(L_1 - L_0)$. The pseudo-R^2 (McFadden 1974) is defined as $1 - L_1/L_0$. This is simply the log likelihood on a scale where 0 corresponds to the "constant-only" model and 1 corresponds to perfect prediction for a discrete model (in which case the overall log likelihood is 0).

Some maximum likelihood routines can report coefficients in an exponentiated form, for example, odds ratios in logistic. Let b be the unexponentiated coefficient, s its standard error, and b_0 and b_1 the reported confidence interval for b. In exponentiated form, the point estimate is e^b, the standard error $e^b s$, and the confidence interval e^{b_0} and e^{b_1}. The displayed Z (or t) statistics and p-values are the same as those for the unexponentiated results. This is justified because $e^b = 1$ and $b = 0$ are equivalent hypotheses, and normality is more likely to hold in the b metric.

References

McFadden, D. L. 1974. "Conditional logit analysis of qualitative choice behavior". In *Frontiers in Econometrics*, edited by P. Zarembka, 105–142. New York: Academic Press.

Pitblado, J. S., B. P. Poi, and W. W. Gould. 2024. *Maximum Likelihood Estimation with Stata*. 5th ed. College Station, TX: Stata Press.

Also see

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

[R] ***set iter*** — Control iteration settings

[SVY] **ml for svy** — Maximum pseudolikelihood estimation for survey data

[M-5] **moptimize( )** — Model optimization

[M-5] **optimize( )** — Function optimization