Title

stata.com

gmm — Generalized method of moments estimation

Description

gmm performs generalized method of moments (GMM) estimation. With the interactive version of the command, you enter the residual equation for each moment condition directly into the dialog box or on the command line by using substitutable expressions. The moment-evaluator program version gives you greater flexibility in exchange for increased complexity; with this version, you write a program in an ado-file that calculates the moments based on a vector of parameters passed to it.

gmm can fit both single- and multiple-equation models. It allows moment conditions of the form $E\{\mathbf{z}_i u_i(\beta)\} = \mathbf{0}$, where $\mathbf{z}_i$ is a vector of instruments, and $u_i(\boldsymbol{\beta})$ is an error term, as well as more general moment conditions of the form $E\{\mathbf{h}_i(\mathbf{z}_i;\boldsymbol{\beta})\} = \mathbf{0}$. gmm works with cross-sectional, time-series, and longitudinal (panel) data.

Menu

Statistics > Endogenous covariates > Generalized method of moments estimation

Syntax

Interactive version

```
gmm ([reqname1:]rexp1) ([reqname2:]rexp2)...[if] [in] [weight] [, options]
```

Moment-evaluator program version

```
gmm moment_prog [if] [in] [weight], { equations(namelist) | nequations(#) }
    { parameters(namelist) | nparameters(#) } [options] [program_options]
```

reqname$_j$ is the jth residual equation name,
rexp$_j$ is the substitutable expression for the jth residual equation, and
moment_prog is a moment-evaluator program.

options	Description
Model	
`derivative(`[*reqname* | #]`/`*name* `=` *dexp*$_{jk}$`)`	specify derivative of *reqname* (or #) with respect to parameter *name*; can be specified more than once (interactive version only)
* `twostep`	use two-step GMM estimator; the default
* `onestep`	use one-step GMM estimator
* `igmm`	use iterative GMM estimator
`variables(`*varlist*`)`	specify variables in model
`nocommonesample`	do not restrict estimation sample to be the same for all equations
Instruments	
`instruments(`[*reqlist*`:`]*varlist*[`, noconstant`]`)`	specify instruments; can be specified more than once
`xtinstruments(`[*reqlist*`:`]*varlist*`, lags(`#$_1$`/`#$_2$`))`	specify panel-style instruments; can be specified more than once
Weight matrix	
`wmatrix(`*wmtype*[`, independent`]`)`	specify weight matrix; *wmtype* may be `robust`, `cluster` *clustvar*, `hac` *kernel* [*lags*], or `unadjusted`
`center`	center moments in weight-matrix computation
`winitial(`*iwtype*[`, independent`]`)`	specify initial weight matrix; *iwtype* may be `unadjusted`, `identity`, `xt` *xtspec*, or the name of a Stata matrix

SE/Robust

`vce(`*vcetype*`[, independent])`	*vcetype* may be `robust`, `cluster` *clustvar*, `bootstrap`, `jackknife`, `hac` *kernel lags*, or `unadjusted`
`quickderivatives`	use alternative method of computing numerical derivatives for VCE

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

Optimization

`from(`*initial_values*`)`	specify initial values for parameters
‡ `igmmiterate(#)`	specify maximum number of iterations for iterated GMM estimator
‡ `igmmeps(#)`	specify # for iterated GMM parameter convergence criterion; default is `igmmeps(1e-6)`
‡ `igmmweps(#)`	specify # for iterated GMM weight-matrix convergence criterion; default is `igmmweps(1e-6)`
optimization_options	control the optimization process; seldom used
`coeflegend`	display legend instead of statistics

*You can specify at most one of these options.

‡These options may be specified only when `igmm` is specified.

program_options	Description

Model

evaluator_options	additional options to be passed to the moment-evaluator program
* `hasderivatives`	moment-evaluator program can calculate parameter-level derivatives
* `haslfderivatives`	moment-evaluator program can calculate linear-form derivatives
† `equations(`*namelist*`)`	specify residual equation names
† `nequations(#)`	specify number of residual equations
‡ `parameters(`*namelist*`)`	specify parameter names
‡ `nparameters(#)`	specify number of parameters

*You may not specify both `hasderivatives` and `haslfderivatives`.

†You must specify `equations(`*namelist*`)` or `nequations(#)`; you may specify both.

‡You must specify `parameters(`*namelist*`)` or `nparameters(#)`; you may specify both.

rexp$_j$ and *dexp*$_{jk}$ 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`, and `statsby` 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.

rexp$_j$ and *dexp*$_{jk}$ are substitutable expressions, that is, 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 `{`*lcname*`:`*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

`derivative(`[*reqname* | #]`/`*name* `=` *dexp*$_{jk}$`)` specifies the derivative of residual equation *reqname* or # with respect to parameter *name*. If *reqname* or # is not specified, `gmm` assumes that the derivative applies to the first residual equation.

For a moment condition of the form $E\{\mathbf{z}_{ji}u_{ji}(\boldsymbol{\beta})\} = \mathbf{0}$, `derivative(`*j*`/`$\beta_k$ `=` *dexp*$_{jk}$`)` is to contain a substitutable expression for $\partial u_{ji}/\partial\beta_k$. If you specified `m` as the *reqname*, then for a moment condition of the form $E\{\mathbf{z}_{mi}u_{mi}(\boldsymbol{\beta})\} = \mathbf{0}$, you can specify `derivative(m/`$\beta_k$ `=` *dexp*$_{mk}$`)`, where m is the index of `m`.

dexp$_{jk}$ uses the same substitutable expression syntax as is used to specify residual equations. If you declare a linear combination in a residual equation, you provide the derivative for the linear combination; `gmm` 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, `gmm` calculates derivatives numerically. You must either specify no derivatives or specify a derivative for each of the k parameters that appears in each of the j residual equations unless the derivative is identically zero. You cannot specify some analytic derivatives and have `gmm` compute the rest numerically.

`twostep`, `onestep`, and `igmm` specify which estimator is to be used. You can specify at most one of these options. `twostep` is the default.

`twostep` requests the two-step GMM estimator. `gmm` obtains parameter estimates based on the initial weight matrix, computes a new weight matrix based on those estimates, and then reestimates the parameters based on that weight matrix.

`onestep` requests the one-step GMM estimator. The parameters are estimated based on an initial weight matrix, and no updating of the weight matrix is performed except when calculating the appropriate variance–covariance (VCE) matrix.

`igmm` requests the iterative GMM estimator. `gmm` obtains parameter estimates based on the initial weight matrix, computes a new weight matrix based on those estimates, reestimates the parameters based on that weight matrix, computes a new weight matrix, and so on, to convergence. Convergence is declared when the relative change in the parameter vector is less than `igmmeps()`, the relative change in the weight matrix is less than `igmmweps()`, or `igmmiterate()` iterations have been

completed. Hall (2005, sec. 2.4 and 3.6) mentions that there may be gains to finite-sample efficiency from using the iterative estimator.

`variables(`*varlist*`)` specifies the variables in the model. `gmm` ignores observations for which any of these variables has a missing value. If you do not specify `variables()`, then `gmm` assumes all the observations are valid and issues an error message if any residual equations evaluate to missing for any observations at the initial value of the parameter vector.

`nocommonesample` requests that `gmm` not restrict the estimation sample to be the same for all equations. By default, `gmm` will restrict the estimation sample to observations that are available for all equations in the model, mirroring the behavior of other multiple-equation estimators such as `nlsur`, `sureg`, or `reg3`. For certain models, however, different equations can have different numbers of observations. For these models, you should specify `nocommonesample`. See *Dynamic panel-data models* below for one application of this option. You cannot specify weights if you specify `nocommonesample`.

Instruments

`instruments(`[*reqlist*`:`]*varlist*[`, noconstant`]`)` specifies a list of instrumental variables to be used. If you specify a single residual equation, then you do not need to specify the equations to which the instruments apply; you can omit the *reqlist* and simply specify `instruments(`*varlist*`)`. By default, a constant term is included in *varlist*; to omit the constant term, use the `noconstant` suboption: `instruments(`*varlist*`, noconstant)`.

If your model has multiple moment conditions of the form

$$E\left\{\begin{array}{c}\mathbf{z}_{1i}u_{1i}(\boldsymbol{\beta})\\ \cdots\\ \mathbf{z}_{qi}u_{qi}(\boldsymbol{\beta})\end{array}\right\}=\mathbf{0}$$

then you can specify multiple corresponding residual equations. Then, specify the *reqname* or an *reqlist* to indicate the residual equations for which the list of variables is to be used as instruments if you do not want that list applied to all the residual equations. For example, you might type

```
gmm (main:rexp1) (rexp2) (rexp3), instruments(z1 z2) ///
  instruments(2: z3) instruments(main 3: z4)
```

Variables `z1` and `z2` will be used as instruments for all three equations, `z3` will be used as an instrument for the second equation, and `z4` will be used as an instrument for the first and third equations. Notice that we chose to supply a name for the first residual equation but not the second two, identifying each by its equation number.

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

`xtinstruments(`[*reqlist*`:`]*varlist*`, lags(`#$_1$`/`#$_2$`))` is for use with panel-data models in which the set of available instruments depends on the time period. As with `instruments()`, you can prefix the list of variables with residual equation names or numbers to target instruments to specific equations. Unlike with `instruments()`, a constant term is not included in *varlist*. You must `xtset` your data before using this option; see [XT] **xtset**.

If you specify

```
gmm ..., xtinstruments(x, lags(1/.)) ...
```

then for panel i and period t, `gmm` uses $x_{i,t-1}, x_{i,t-2}, \ldots, x_{i1}$ as instruments. More generally, specifying `xtinstruments(x, lags(`#$_1$`, `#$_2$`))` uses $x_{i,t-\#_1}, \ldots, x_{i,t-\#_2}$ as instruments; setting $\#_2$ = `.` requests all available lags. #$_1$ and #$_2$ must be zero or positive integers.

gmm automatically excludes observations for which no valid instruments are available. It does, however, include observations for which only a subset of the lags is available. For example, if you request that lags one through three be used, then gmm will include the observations for the second and third time periods even though fewer than three lags are available as instruments.

Weight matrix

wmatrix(*wmtype*[, independent]) specifies the type of weight matrix to be used in conjunction with the two-step and iterated GMM estimators.

Specifying wmatrix(robust) requests a weight matrix that is appropriate when the errors are independent but not necessarily identically distributed. wmatrix(robust) is the default.

Specifying wmatrix(cluster *clustvar*) requests a weight matrix that accounts for arbitrary correlation among observations within clusters identified by *clustvar*.

Specifying wmatrix(hac *kernel #*) requests a heteroskedasticity- and autocorrelation-consistent (HAC) weight matrix using the specified kernel (see below) with # lags. The bandwidth of a kernel is equal to the number of lags plus one.

Specifying wmatrix(hac *kernel* opt [#]) requests an HAC weight matrix using the specified kernel, and the lag order is selected using Newey and West's (1994) optimal lag-selection algorithm. # is an optional tuning parameter that affects the lag order selected; see the discussion in *Methods and formulas*.

Specifying wmatrix(hac *kernel*) requests an HAC weight matrix using the specified kernel and $N - 2$ lags, where N is the sample size.

There are three kernels available for HAC weight matrices, and you can request each one by using the name used by statisticians or the name perhaps more familiar to economists:

bartlett or nwest requests the Bartlett (Newey–West) kernel;

parzen or gallant requests the Parzen (Gallant) kernel; and

quadraticspectral or andrews requests the quadratic spectral (Andrews) kernel.

Specifying wmatrix(unadjusted) requests a weight matrix that is suitable when the errors are homoskedastic. In some applications, the GMM estimator so constructed is known as the (nonlinear) two-stage least-squares (2SLS) estimator.

Including the independent suboption creates a weight matrix that assumes moment conditions are independent. This suboption is often used to replicate other models that can be motivated outside the GMM framework, such as the estimation of a system of equations by system-wide 2SLS. This suboption has no effect if only one residual equation is specified.

wmatrix() has no effect if onestep is also specified.

center requests that the sample moments be centered (demeaned) when computing GMM weight matrices. By default, centering is not done.

winitial(*iwtype*[, independent]) specifies the weight matrix to use to obtain the first-step parameter estimates.

Specifying winitial(unadjusted) requests a weight matrix that assumes the moment conditions are independent and identically distributed. This matrix is of the form $(\mathbf{Z}'\mathbf{Z})^{-1}$, where $\mathbf{Z}$ represents all the instruments specified in the instruments() option. To avoid a singular weight matrix, you should specify at least $q - 1$ moment conditions of the form $E\{\mathbf{z}_{hi}u_{hi}(\boldsymbol{\beta})\} = \mathbf{0}$, where q is the number of moment conditions, or you should specify the independent suboption.

Including the independent suboption creates a weight matrix that assumes moment conditions are independent. Elements of the weight matrix corresponding to covariances between two moment conditions are set equal to zero. This suboption has no effect if only one residual equation is specified.

winitial(unadjusted) is the default.

winitial(identity) requests that the identity matrix be used.

winitial(xt *xtspec*) is for use with dynamic panel-data models in which one of the residual equations is specified in first-differences form. *xtspec* is a string consisting of the letters "L" and "D", the length of which is equal to the number of residual equations in the model. You specify "L" for a residual equation if that residual equation is written in levels, and you specify "D" for a residual equation if it is written in first differences; *xtspec* is not case sensitive. When you specify this option, you can specify at most one residual equation in levels and one residual equation in first differences. See the examples listed in *Dynamic panel-data models* under *Remarks and examples* below.

winitial(*matname*) requests that Stata matrix *matname* be used. You cannot specify the independent suboption if you specify winitial(*matname*).

SE/Robust

vce(*vcetype*[, independent]) specifies the type of standard error reported, which includes types 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***.

vce(unadjusted) specifies that an unadjusted (nonrobust) VCE matrix be used; this, along with the twostep option, results in the "optimal two-step GMM" estimates often discussed in textbooks.

The default *vcetype* is based on the *wmtype* specified in the wmatrix() option. If wmatrix() is specified but vce() is not, then *vcetype* is set equal to *wmtype*. To override this behavior and obtain an unadjusted (nonrobust) VCE matrix, specify vce(unadjusted).

Specifying vce(bootstrap) or vce(jackknife) results in standard errors based on the bootstrap or jackknife, respectively. See [R] ***vce_option***, [R] **bootstrap**, and [R] **jackknife** for more information on these VCEs.

The syntax for *vcetype*s other than bootstrap and jackknife is identical to those for wmatrix().

quickderivatives requests that an alternative method be used to compute the numerical derivatives for the VCE. This option has no effect if you specify the derivatives(), hasderivatives, or haslfderivatives option.

The VCE depends on a matrix of partial derivatives that gmm must compute numerically unless you supply analytic derivatives. This Jacobian matrix will be especially large if your model has many instruments, residual equations, or parameters.

By default, gmm computes each element of the Jacobian matrix individually, searching for an optimal step size each time. Although this procedure results in accurate derivatives, it is computationally taxing: gmm may have to evaluate the moments of your model five or more times for each element of the Jacobian matrix.

When you specify the quickderivatives option, gmm computes all derivatives corresponding to a parameter at once, using a fixed step size proportional to the parameter's value. This method requires just two evaluations of the model's moments to compute an entire column of the Jacobian matrix and therefore has the most impact when you specify many instruments or residual equations.

Most of the time, the two methods produce virtually identical results, but the `quickderivatives` method may fail if a residual equation is highly nonlinear or if instruments differ by orders of magnitude. In the rare case where you specify `quickderivatives` and obtain suspiciously large or small standard errors, try refitting your model without this 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, `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**.

Optimization

`from(`*initial_values*`)` specifies the initial values to begin the estimation. You can specify a parameter name, its initial value, another parameter name, its initial value, and so on, 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

```
gmm ..., from(alpha 1.23 delta 4.57) ...
```

or equivalently

```
matrix define initval = (1.23, 4.57)
gmm ..., 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, `gmm` exits with an error message. If you specify a matrix, the values must be in the same order in which the parameters are declared in your model.

`igmmiterate(#)`, `igmmeps(#)`, and `igmmweps(#)` control the iterative process for the iterative GMM estimator. These options can be specified only if you also specify `igmm`.

`igmmiterate(#)` specifies the maximum number of iterations to perform with the iterative GMM estimator. The default is the number set using `set maxiter`, which is 300 by default.

`igmmeps(#)` specifies the convergence criterion used for successive parameter estimates when the iterative GMM estimator is used. The default is `igmmeps(1e-6)`. Convergence is declared when the relative difference between successive parameter estimates is less than `igmmeps()` and the relative difference between successive estimates of the weight matrix is less than `igmmweps()`.

`igmmweps(#)` specifies the convergence criterion used for successive estimates of the weight matrix when the iterative GMM estimator is used. The default is `igmmweps(1e-6)`. Convergence is declared when the relative difference between successive parameter estimates is less than `igmmeps()` and the relative difference between successive estimates of the weight matrix is less than `igmmweps()`.

optimization_options: `technique()`, `conv_maxiter()`, `conv_ptol()`, `conv_vtol()`, `conv_nrtol()`, `tracelevel()`. `technique()` specifies the optimization technique to use; `gn` (the default), `nr`, `dfp`, and `bfgs` are allowed. `conv_maxiter()` specifies the maximum number

of iterations; conv_ptol(), conv_vtol(), and conv_nrtol() specify the convergence criteria for the parameters, gradient, and scaled Hessian, respectively. tracelevel() allows you to obtain additional details during the iterative process. See [M-5] **optimize( )**.

The following options pertain only to the moment-evaluator program version of gmm.

Model

evaluator_options refer to any options allowed by your *moment_prog*.

hasderivatives and haslfderivatives indicate that you have written your moment-evaluator program to compute derivatives. You may specify one or the other but not both. If you do not specify either of these options, gmm computes the derivatives numerically.

hasderivatives indicates that your moment-evaluator program computes parameter-level derivatives.

haslfderivatives indicates that your moment-evaluator program computes equation-level derivatives and is useful only when you specify the parameters of your model using the {*lcname*:*varlist*} syntax of the parameters() option.

See *Details of moment-evaluator programs* below for more information.

equations(*namelist*) specifies the names of the residual equations in the model. If you specify both equations() and nequations(), the number of names in the former must match the number specified in the latter.

nequations(*#*) specifies the number of residual equations in the model. If you do not specify names with the equations() option, gmm numbers the residual equations 1, 2, 3, ... If you specify both equations() and nequations(), the number of names in the former must match the number specified in the latter.

parameters(*namelist*) specifies the names of the parameters in the model. The names of the parameters must comply with the naming conventions of Stata's variables; see [U] **11.3 Naming conventions**.

Alternatively, you can use parameter equation notation to specify linear combinations of parameters. Each linear combination is of the form {*lcname*:*varlist*}, where *varlist* is one or more variable names. Specify the system variable _cons in *varlist* to include a constant term. Distinguish between {*lcname*:*varlist*}, in which *lcname* identifies the linear combination, and (*reqname*:*rexp*), in which *reqname* identifies the residual equation. When you use linear-combination syntax, gmm prepends each element of the parameter vector passed to your evaluator program with *lcname*: to generate unique names.

If you specify both parameters() and nparameters(), the number of names in the former must match the number specified in the latter.

nparameters(*#*) specifies the number of parameters in the model. If you do not specify names with the parameters() option, gmm names them b1, b2, ..., b*#*. If you specify both parameters() and nparameters(), the number of names in the former must match the number specified in the latter.

The following option is available with gmm 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

The GMM estimator is a workhorse of modern econometrics and is discussed in all the leading textbooks, including Cameron and Trivedi (2005, 2022), Davidson and MacKinnon (1993), Greene (2018, 500–534), Ruud (2000), Hayashi (2000), Wooldridge (2010), Hamilton (1994), and Baum (2006). An excellent treatise on GMM with a focus on time-series applications is Hall (2005). The collection of papers by Mátyás (1999) provides both theoretical and applied aspects of GMM. Here we give a brief introduction to the methodology and emphasize how the various options of `gmm` are used.

The starting point for the GMM estimator is the analogy principle, which says we can estimate a parameter by replacing a population moment condition with its sample analogue. For example, the mean of an independent and identically distributed (i.i.d.) population is defined as the value μ such that the first (central) population moment is zero; that is, μ solves $E(y-\mu)=0$, where y is a random draw from the population. The analogy principle tells us that to obtain an estimate, $\widehat{\mu}$, of μ, we replace the population-expectations operator with its sample analogue (Manski 1988; Wooldridge 2010),

$$E(y-\mu)=0 \;\longrightarrow\; \frac{1}{N}\sum_{i=1}^{N}(y_i-\widehat{\mu})=0 \;\longrightarrow\; \widehat{\mu}=\frac{1}{N}\sum_{i=1}^{N}y_i$$

where N denotes sample size, and y_i represents the ith observation of y in our dataset. The estimator $\widehat{\mu}$ is known as the method of moments (MM) estimator because we started with a population moment condition and then applied the analogy principle to obtain an estimator that depends on the observed data.

Ordinary least-squares (OLS) regression can also be viewed as an MM estimator. In the model

$$y=\mathbf{x}'\boldsymbol{\beta}+u$$

we assume that u has mean zero conditional on $\mathbf{x}$: $E(u|\mathbf{x})=0$. This conditional expectation implies the unconditional expectation $E(\mathbf{x}u)=\mathbf{0}$ because, with the law of iterated expectations,

$$E(\mathbf{x}u)=E_{\mathbf{x}}\left\{E(\mathbf{x}u|\mathbf{x})\right\}=E_{\mathbf{x}}\left\{\mathbf{x}\,E(u|\mathbf{x})\right\}=\mathbf{0}$$

(Using the law of iterated expectations to derive unconditional expectations based on conditional expectations, perhaps motivated by subject theory, is extremely common in GMM estimation.) Continuing, we see that

$$E(\mathbf{x}u)=E\left\{\mathbf{x}(y-\mathbf{x}'\boldsymbol{\beta})\right\}=\mathbf{0}$$

Applying the analogy principle, we obtain

$$E\left\{\mathbf{x}(y-\mathbf{x}'\beta)\right\} \longrightarrow \frac{1}{N}\sum_{i=1}^{N}\mathbf{x}_i(y_i-\mathbf{x}_i'\boldsymbol{\beta})=\mathbf{0}$$

so that

$$\widehat{\boldsymbol{\beta}}=\left(\sum\nolimits_i \mathbf{x}_i\mathbf{x}_i'\right)^{-1}\sum\nolimits_i \mathbf{x}_i y_i$$

which is just the more familiar formula $\widehat{\boldsymbol{\beta}}=(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{y}$ written with summation notation.

In both of the previous examples, the number of parameters we were estimating equaled the number of moment conditions. In the first example, we estimated one parameter, μ, and had one moment condition $E(y-\mu)=0$. In the second example, the parameter vector $\boldsymbol{\beta}$ had k elements, as did the vector of regressors $\mathbf{x}$, yielding k moment conditions. Ignoring peculiar cases, we see that a model of m equations in m unknowns has a unique solution; and because the residual equations in these examples were linear, we could solve for the parameters analytically. If the moment conditions had been nonlinear, we would have had to use numerical techniques to solve for the parameters, but that is not a significant limitation with modern computers.

What if we have more moment conditions than parameters? Say we have q moment conditions and k parameters. A model of $q>k$ equations in k unknowns does not have a unique solution. Any size-k subset of the moment conditions would yield a consistent parameter estimate, though the parameter estimate would in general be different depending on which k moment conditions we used.

For concreteness, let's return to our regression model,

$$y=\mathbf{x}'\boldsymbol{\beta}+u$$

Now, however, we no longer wish to assume that $E(\mathbf{x}u)=\mathbf{0}$; we suspect that the error term u affects one or more elements of $\mathbf{x}$. Thus, we can no longer use the OLS estimator. Suppose we have a vector $\mathbf{z}$ with the properties that $E(\mathbf{z}u)=\mathbf{0}$, that the rank of $E(\mathbf{z}'\mathbf{z})$ equals q, and that the rank of $E(\mathbf{z}'\mathbf{x})=k$. The first assumption simply states that $\mathbf{z}$ is not correlated with the error term. The second assumption rules out perfect collinearity among the elements of $\mathbf{z}$. The third assumption, known as the rank condition in econometrics, ensures that $\mathbf{z}$ is sufficiently correlated with $\mathbf{x}$ and that the estimator is feasible. If some elements of $\mathbf{x}$ are not correlated with u, then they should also appear in $\mathbf{z}$.

If $q<k$, then the rank of $E(\mathbf{z}'\mathbf{x})<k$, which violates the rank condition.

If $q=k$, then we can use the simpler MM estimator we already discussed; we would obtain what is sometimes called the simple instrumental-variables estimator $\widehat{\boldsymbol{\beta}}=(\sum_i \mathbf{z}_i\mathbf{x}_i')^{-1}\sum_i \mathbf{z}_i y_i$. The rank condition ensures that $\sum_i \mathbf{z}_i\mathbf{x}_i'$ is invertible, at least in the population.

If $q>k$, the GMM estimator chooses the value, $\widehat{\boldsymbol{\beta}}$, that minimizes a quadratic function of the moment conditions. We could define

$$\widehat{\boldsymbol{\beta}}\equiv \arg\min_{\boldsymbol{\beta}}\left\{\frac{1}{N}\sum\nolimits_i \mathbf{z}_i u_i(\boldsymbol{\beta})\right\}'\left\{\frac{1}{N}\sum\nolimits_i \mathbf{z}_i u_i(\boldsymbol{\beta})\right\}$$

where for our linear regression example $u_i(\boldsymbol{\beta})=y_i-\mathbf{x}_i'\boldsymbol{\beta}$. This estimator tries to make the moment conditions as close to zero as possible. This simple estimator, however, applies equal weight to each of the moment conditions; and as we will see later, we can obtain more efficient estimators by choosing to weight some moment conditions more highly than others.

Consider the quadratic function

$$Q(\boldsymbol{\beta}) = \left\{ \frac{1}{N} \sum\nolimits_i \mathbf{z}_i u_i(\boldsymbol{\beta}) \right\}' \mathbf{W} \left\{ \frac{1}{N} \sum\nolimits_i \mathbf{z}_i u_i(\boldsymbol{\beta}) \right\}$$

where $\mathbf{W}$ is a symmetric positive-definite matrix known as a weight matrix. Then we define the GMM estimator as

$$\widehat{\boldsymbol{\beta}} \equiv \arg\min_{\boldsymbol{\beta}} \; Q(\boldsymbol{\beta}) \tag{1}$$

Continuing with our regression model example, if we choose

$$\mathbf{W} = \left(\frac{1}{N} \sum\nolimits_i \mathbf{z}_i \mathbf{z}_i' \right)^{-1}$$

then we obtain

$$\widehat{\boldsymbol{\beta}} = \left\{ \left(\frac{1}{N} \sum\nolimits_i \mathbf{x}_i \mathbf{z}_i' \right) \left(\frac{1}{N} \sum\nolimits_i \mathbf{z}_i \mathbf{z}_i' \right)^{-1} \left(\frac{1}{N} \sum\nolimits_i \mathbf{z}_i \mathbf{x}_i' \right) \right\}^{-1} \times \left(\frac{1}{N} \sum\nolimits_i \mathbf{x}_i \mathbf{z}_i' \right) \left(\frac{1}{N} \sum\nolimits_i \mathbf{z}_i \mathbf{z}_i' \right)^{-1} \left(\frac{1}{N} \sum\nolimits_i \mathbf{z}_i y_i \right)$$

which is the well-known two-stage least-squares (2SLS) estimator. Our choice of weight matrix here was based on the assumption that u was homoskedastic. A feature of GMM estimation is that by selecting different weight matrices, we can obtain estimators that can tolerate heteroskedasticity, clustering, autocorrelation, and other features of u. See [R] **ivregress** for more information about the 2SLS and linear GMM estimators.

Returning to the case where the model is "just identified", meaning that $q = k$, if we apply the GMM estimator, we will obtain the same estimate, $\widehat{\boldsymbol{\beta}}$, regardless of our choice of $\mathbf{W}$. Because $q = k$, if a unique solution exists, it will set all the sample moment conditions jointly to zero, so $\mathbf{W}$ has no impact on the value of $\boldsymbol{\beta}$ that minimizes the objective function.

We will highlight other features of the GMM estimator and the `gmm` command as we proceed through examples. First, though, we discuss how to specify moment conditions by using substitutable expressions.

Substitutable expressions

To use the interactive version of `gmm`, you define the moment conditions by using substitutable expressions. Your moment conditions are of the form $E\{z_i' u_i(\boldsymbol{\beta})\} = \mathbf{0}$, where $u_i(\boldsymbol{\beta})$ is a residual expression that depends on the parameter vector $\boldsymbol{\beta}$ as well as variables in your dataset, though we suppress expressing the variables for notational simplicity.

`gmm` requires you to write a substitutable expression for $u_i(\boldsymbol{\beta})$. This substitutable expression is the right-hand side of the model written in terms of u, or in the language of Stata syntax, a "residual equation". For example, suppose you want to fit the function $y = f(\mathbf{x}; \boldsymbol{\beta}) + u$. In this example, $u_i(\boldsymbol{\beta}) = u = y - f(\mathbf{x}; \boldsymbol{\beta})$, so you would type

`gmm (y -` *expression for* $f(\mathbf{x};\boldsymbol{\beta})$`), ...`

Note that we are not restricted to models with additive error terms.

In general, 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.

   You can also specify initial values by using the from() option. Initial values specified in from() override whatever initial values are given within the substitutable expression. If you do not specify an initial value for a parameter, it is initialized to 0.
3. Linear combinations of variables can be included using the notation {*lcname*:*varlist*}: {xb: mpg price weight}, {score: w x z}, etc. Parameters of linear combinations are initialized to 0.

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

The notation {xb:x1 x2 x3} tells gmm that you want a linear combination of the variables x1, x2, and x3. We named this linear combination xb, so gmm will name the three parameters xb:x1, xb:x2, and xb:x3, which corresponds to the three variables x1, x2, and x3 in the xb equation. Specify _cons to include a constant term in a linear combination. Factor variables and time-series operators are allowed; see [U] 11.4.3 Factor variables and [U] 11.4.4 Time-series varlists.

Once we have declared the variables in the linear combination xb, we can refer to the linear combination in our substitutable expression by using the notation xb:. The colon is not optional; it tells gmm that you are referring to a previously declared linear combination, not an individual parameter. This shorthand notation is also handy when specifying derivatives, as we will show later.

▷ Example 1: OLS regression

In *Introduction*, we stated that OLS is an MM estimator. Say that we want to fit the model

$$\texttt{mpg} = \beta_0 + \beta_1\texttt{weight} + \beta_2\texttt{length} + u$$

where u is an i.i.d. error term. Recall that the moment condition for OLS regression is $E(\mathbf{x}u) = \mathbf{0}$, where $\mathbf{x}$, the list of instruments, is the same as the list of regressors in the model. Writing this in the form required for a gmm substitutable expression, we have

$$u = \texttt{mpg} - \beta_0 - \beta_1\texttt{weight} - \beta_2\texttt{length}$$

The right-hand side of the equation is the substitutable expression that we will provide to gmm. We give β_0, β_1, and β_2 the parameter names b0, b1, and b2 and enclose them in curly braces. Because linear combinations declared in substitutable expressions do not include a constant term by default, we include our own (b0). We specify the regressors, weight and length, with their respective parameters and also in the instruments() option. gmm includes a constant term in the instrument list by default, so we do not need to add an additional term there. Thus, our command is

```
. use https://www.stata-press.com/data/r17/auto
(1978 automobile data)
. gmm (mpg - {b1}*weight - {b2}*length - {b0}), instruments(weight length)
Step 1
Iteration 0:   GMM criterion Q(b) =   475.4138
Iteration 1:   GMM criterion Q(b) =  2.696e-20
Iteration 2:   GMM criterion Q(b) =  3.329e-27
Step 2
Iteration 0:   GMM criterion Q(b) =  5.109e-28
Iteration 1:   GMM criterion Q(b) =  7.237e-32
note: model is exactly identified.
GMM estimation
Number of parameters =   3
Number of moments    =   3
Initial weight matrix: Unadjusted                 Number of obs   =         74
GMM weight matrix:     Robust
```

	Coefficient	Robust std. err.	z	P>|z|	[95% conf.	interval]
/b1	-.0038515	.0019472	-1.98	0.048	-.0076678	-.0000351
/b2	-.0795935	.0677528	-1.17	0.240	-.2123866	.0531996
/b0	47.88487	7.50599	6.38	0.000	33.1734	62.59634

```
Instruments for equation 1: weight length _cons
```

Because the number of moments equals the number of parameters we are estimating, the model is said to be "just identified" or "exactly identified". Therefore, the choice of weight matrix has no impact on the solution to (1), and the criterion function $Q(\boldsymbol{\beta})$ achieves its minimum value at 0.

The OLS estimator is a one-step GMM estimator, but we did not bother to specify the `onestep` option, because the model is just identified. Doing a second step of GMM estimation affects neither the point estimates nor the standard errors, so to keep the syntax as simple as possible, we did not include the `onestep` option. The first step of estimation resulted in $Q(\boldsymbol{\beta}) = 0$ as expected, and the second step of estimation did not change the minimized value of $Q(\boldsymbol{\beta})$. (4×10^{-27} and 3×10^{-31} are both 0 for all practical purposes.)

When you do not specify either the `wmatrix()` or the `vce()` option, `gmm` reports heteroskedasticity-robust standard errors. The parameter estimates reported here match those that we would obtain from the command

```
. regress mpg weight length, vce(robust)
```

The standard errors reported by that `regress` command would be larger than those reported by `gmm` by a factor of $\text{sqrt}(74/71)$ because `regress` makes a small-sample adjustment to the estimated variance matrix while `gmm` does not. Likewise, if we had specified the `vce(unadjusted)` option with our `gmm` command, then our standard errors would differ by a factor of $\text{sqrt}(74/71)$ from those reported by `regress` without the `vce(robust)` option.

We could have submitted our substitutable expression using the notation for linear combinations of parameters. If we select `xb` as the name of our parameter equation, we could type

```
. gmm (mpg - {xb: weight length _cons}), instruments(weight length)
```

and obtain identical results. With this syntax, instead of having parameters `b1`, `b2`, and `b0`, we would have parameters `xb:weight`, `xb:length`, and `xb:_cons`. Note that `_cons` allows you to include a constant in a linear combination, so this time, we do not have to specify a separate parameter from our *varlist*.

◁

Factor variables and time-series–operated variables are allowed in the linear combinations. For example,

```
. regress mpg i.foreign i.foreign#c.weight, vce(robust)
```

produces the same results as

```
. gmm (mpg - {xb:i.foreign i.foreign#c.weight _cons}),
> instruments(i.foreign i.foreign#c.weight)
```

See [U] **11.4.3 Factor variables** and [U] **11.4.4 Time-series varlists** for an introduction to factor variables and time-series operators. See example 4 for an example of factor-variable syntax with `gmm`. See example 16 for an example using time-series–operated variables.

▷ Example 2: Instrumental-variables regression

In *Introduction*, we mentioned that 2SLS can be viewed as a GMM estimator. In example 1 of [R] **ivregress**, we fit by a 2SLS model of rental rates (`rent`) as a function of the value of owner-occupied housing (`hsngval`) and the percentage of the population living in urban areas (`pcturban`):

$$\texttt{rent} = \beta_0 + \beta_1\texttt{hsngval} + \beta_2\texttt{pcturban} + u$$

We argued that random shocks that affect rental rates likely also affect housing values, so we treated `hsngval` as an endogenous variable. As additional instruments, we used family income, `faminc`, and three regional dummies (`reg2`–`reg4`).

To replicate the results of `ivregress 2sls` by using `gmm`, we type

```
. use https://www.stata-press.com/data/r17/hsng2, clear
(1980 Census housing data)
. gmm (rent - {xb:hsngval pcturban _cons}),
> instruments(pcturban faminc reg2-reg4) vce(unadjusted) onestep
Step 1
Iteration 0:   GMM criterion Q(b) =   56115.03
Iteration 1:   GMM criterion Q(b) =  110.91583
Iteration 2:   GMM criterion Q(b) =  110.91583
GMM estimation
Number of parameters =   3
Number of moments    =   6
Initial weight matrix: Unadjusted                 Number of obs   =         50
```

	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
hsngval	.0022398	.0003284	6.82	0.000	.0015961	.0028836
pcturban	.081516	.2987652	0.27	0.785	-.5040531	.6670851
_cons	120.7065	15.22839	7.93	0.000	90.85942	150.5536

```
Instruments for equation 1: pcturban faminc reg2 reg3 reg4 _cons
```

We specified `vce(unadjusted)` so that we would obtain an unadjusted VCE matrix and our standard errors would match those reported in [R] **ivregress**.

Note how we specified the `instruments()` option. In *Introduction*, we mentioned that the moment conditions for the 2SLS estimator are $E(\mathbf{z}u) = \mathbf{0}$, and we mentioned that if some elements of $\mathbf{x}$ (the regressors) are not endogenous, then they should also appear in $\mathbf{z}$. In this model, we assume the regressor `pcturban` is exogenous, so we included it in the list of instrumental variables. Commands like `ivregress`, `ivprobit`, and `ivtobit` accept standard *varlist*s, so they can deduce the exogenous regressors in the model. Because `gmm` accepts arbitrary functions in the form of substitutable expressions, it has no way of discerning the exogenous variables of the model on its own.

Also notice that we specified the `onestep` option. The 2SLS estimator is a one-step GMM estimator that is based on a weight matrix that assumes the error terms are i.i.d. Unlike the previous example, this example had more instruments than parameters, so the minimized value of $Q(\boldsymbol{\beta})$ is nonzero. We discuss the weight matrix and its relationship to two-step estimation next.

◁

The weight matrix and two-step estimation

Recall our definition of the GMM estimator given in (1). The estimator, $\widehat{\boldsymbol{\beta}}$, depends on the choice of the weight matrix, $\mathbf{W}$. Under relatively mild assumptions, our estimator, $\widehat{\boldsymbol{\beta}}$, is consistent regardless of the choice of $\mathbf{W}$, so how are we to decide what $\mathbf{W}$ to use? The most common solution is to use the two-step estimator, which we now describe.

A key result in Hansen's (1982) seminal paper is that if we denote by $\mathbf{S}$ the covariance matrix of the moment conditions, then the optimal (in a way we make precise later) GMM estimator is the one that uses a weight matrix equal to the inverse of the moment covariance matrix. That is, if we let $\mathbf{S} = \text{Cov}(\mathbf{z}u)$, then we want to use $\mathbf{W} = \mathbf{S}^{-1}$. But how do we obtain $\mathbf{S}$ in the first place?

If we assume that the errors are i.i.d., then

$$\text{Cov}(\mathbf{z}u) = E(u^2\mathbf{z}\mathbf{z}') = \sigma^2 E(\mathbf{z}\mathbf{z}')$$

where σ^2 is the variance of u. Because σ^2 is a positive scalar, we can ignore it when solving (1). Thus, we compute

$$\widehat{\mathbf{W}}_1 = \left(\frac{1}{N}\sum\nolimits_i \mathbf{z}_i\mathbf{z}_i'\right)^{-1} \tag{2}$$

which does not depend on any unknown model parameters. (Notice that $\widehat{\mathbf{W}}_1$ is the same weight matrix used in 2SLS.) Given $\widehat{\mathbf{W}}_1$, we can solve (1) to obtain an initial estimate, say, $\widehat{\boldsymbol{\beta}}_1$.

Our estimate, $\widehat{\boldsymbol{\beta}}_1$, is consistent, so by Slutsky's theorem, the sample residuals $\widehat{u}$ computed at this value of $\boldsymbol{\beta}$ will also be consistent. Using virtually the same arguments used to justify the Huber/Eicker/White heteroskedasticity-robust VCE, if we assume that the residuals are independent though not identically distributed, we can estimate $\mathbf{S}$ as

$$\widehat{\mathbf{S}} = \frac{1}{N}\sum_i \widehat{u}_i^2\mathbf{z}_i\mathbf{z}_i'$$

Then, in the second step, we re-solve (1), using $\widehat{\mathbf{W}}_2 = \widehat{\mathbf{S}}^{-1}$, which yields the two-step GMM estimate $\widehat{\boldsymbol{\beta}}_2$. If the residuals exhibit clustering, you can specify `wmatrix(cluster` *varname*`)` so that `gmm` computes a weight matrix that does not assume the u_i's are independent within clusters identified by *varname*. You can specify `wmatrix(hac ...)` to obtain weight matrices that are suitable for when the u_i's exhibit autocorrelation as well as heteroskedasticity.

We could take the point estimates from the second round of estimation and use them to compute yet another weight matrix, $\widehat{\mathbf{W}}_3$, say, re-solve (1) yet again, and so on, stopping when the parameters or weight matrix do not change much from one iteration to the next. This procedure is known as the iterative GMM estimator and is obtained with the `igmm` option. Asymptotically, the two-step and iterative GMM estimators have the same distribution. However, Hall (2005, 90) suggests that the iterative estimator may have better finite-sample properties.

Instead of computing $\widehat{\mathbf{W}}_1$ as in (2), we could simply choose $\widehat{\mathbf{W}}_1 = \mathbf{I}$, the identity matrix. The initial estimate, $\widehat{\boldsymbol{\beta}}_1$, would still be consistent. You can request this behavior by specifying the `winitial(identity)` option. However, if you specify all of your moment conditions of the form $E(\mathbf{z}u) = \mathbf{0}$, we recommend using the default `winitial(unadjusted)` instead; the rescaling of the moment conditions implied by using a homoskedastic initial weight matrix makes the numerical routines used to solve (1) more stable.

If you fit a model with more than one of the moment conditions of the form $E\{h(\mathbf{z};\boldsymbol{\beta})\} = \mathbf{0}$, then you must use `winitial(identity)` or `winitial(unadjusted, independent)`. With moment conditions of that form, you do not specify a list of instruments, and `gmm` cannot evaluate (2)—the matrix expression in parentheses would necessarily be singular, so it cannot be inverted.

▷ Example 3: Two-step linear GMM estimator

From the previous discussion and the comments in *Introduction*, we see that the linear 2SLS estimator is a one-step GMM estimator where we use the weight matrix defined in (2) that assumes the errors are i.i.d. If we use the 2SLS estimate of $\boldsymbol{\beta}$ to obtain the sample residuals, compute a new weight matrix based on those residuals, and then do a second step of GMM estimation, we obtain the linear two-step GMM estimator as implemented by `ivregress gmm`.

In example 3 of [R] **ivregress**, we fit the model of rental rates as discussed in example 2 above. We now allow the residuals to be heteroskedastic, though we will maintain our assumption that they are independent. We type

```
. gmm (rent - {xb:hsngval pcturban _cons}), instruments(pcturban faminc reg2-reg4)
Step 1
Iteration 0:   GMM criterion Q(b) =   56115.03
Iteration 1:   GMM criterion Q(b) =  110.91583
Iteration 2:   GMM criterion Q(b) =  110.91583
Step 2
Iteration 0:   GMM criterion Q(b) =   .2406087
Iteration 1:   GMM criterion Q(b) =  .13672801
Iteration 2:   GMM criterion Q(b) =  .13672801
GMM estimation
Number of parameters =   3
Number of moments    =   6
Initial weight matrix: Unadjusted                 Number of obs   =         50
GMM weight matrix:     Robust
```

	Coefficient	Robust std. err.	z	P>|z|	[95% conf.	interval]
hsngval	.0014643	.0004473	3.27	0.001	.0005877	.002341
pcturban	.7615482	.2895105	2.63	0.009	.1941181	1.328978
_cons	112.1227	10.80234	10.38	0.000	90.95052	133.2949

```
Instruments for equation 1: pcturban faminc reg2 reg3 reg4 _cons
```

By default, `gmm` computes a heteroskedasticity-robust weight matrix before the second step of estimation, though we could have specified `wmatrix(robust)` if we wanted to be explicit. Because we did not specify the `vce()` option, `gmm` used a heteroskedasticity-robust one. Our results match those in example 3 of [R] **ivregress**. Moreover, the only substantive difference between this example and example 2 is that here we did not specify the `onestep` option, so we obtain the two-step estimates.

◁

Obtaining standard errors

This section is a bit more theoretical and can be skipped on first reading. However, the information is sufficiently important that you should return to this section at some point.

So far in our discussion, we have focused on point estimation without much mention of how we obtain the standard errors of the estimates. We also mentioned that if we choose $\mathbf{W}$ to be the inverse of the covariance matrix of the moment conditions, then we obtain the "optimal" GMM estimator. We elaborate those points now.

Using mostly standard statistical arguments, we can show that for the GMM estimator defined in (1), the variance of $\widehat{\boldsymbol{\beta}}$ is given by

$$\text{Var}(\widehat{\boldsymbol{\beta}}) = \frac{1}{N}\left\{\overline{\mathbf{G}}(\widehat{\boldsymbol{\beta}})'\mathbf{W}\overline{\mathbf{G}}(\widehat{\boldsymbol{\beta}})\right\}^{-1}\overline{\mathbf{G}}(\widehat{\boldsymbol{\beta}})'\mathbf{W}\mathbf{S}\mathbf{W}\overline{\mathbf{G}}(\widehat{\boldsymbol{\beta}})\left\{\overline{\mathbf{G}}(\widehat{\boldsymbol{\beta}})'\mathbf{W}\overline{\mathbf{G}}(\widehat{\boldsymbol{\beta}})\right\}^{-1} \tag{3}$$

where

$$\overline{\mathbf{G}}(\widehat{\boldsymbol{\beta}}) = \frac{1}{N}\sum\nolimits_i \mathbf{z}_i \left.\frac{\partial u_i}{\partial \boldsymbol{\beta}}\right|_{\boldsymbol{\beta}=\widehat{\boldsymbol{\beta}}} \qquad \text{or} \qquad \overline{\mathbf{G}}(\widehat{\boldsymbol{\beta}}) = \frac{1}{N}\sum\nolimits_i \left.\frac{\partial \mathbf{h}_i}{\partial \boldsymbol{\beta}}\right|_{\boldsymbol{\beta}=\widehat{\boldsymbol{\beta}}}$$

as the case may be and $\mathbf{S} = E(\mathbf{z}uu'\mathbf{z}')$.

Assuming the `vce(unadjusted)` option is not specified, `gmm` reports standard errors based on the robust variance matrix defined in (3). For the two-step estimator, $\mathbf{W}$ is the weight matrix requested with the `wmatrix()` option, and it is calculated based on the residuals obtained after the first estimation step. The second-step point estimates and residuals are obtained, and $\mathbf{S}$ is calculated based on the specification of the `vce()` option. For the iterated estimator, $\mathbf{W}$ is calculated based on the second-to-last round of estimation, while $\mathbf{S}$ is based on the residuals obtained after the last round of estimation. Computation of the covariance matrix for the one-step estimator is, perhaps surprisingly, more involved; we discuss the covariance matrix with the one-step estimator in the technical note at the end of this section.

If the model is exactly identified, the matrix $\overline{\mathbf{G}}(\widehat{\boldsymbol{\beta}})$ is square, and (3) simplifies to the following:

$$\text{Var}(\widehat{\boldsymbol{\beta}}) = \frac{1}{N}\overline{\mathbf{G}}(\widehat{\boldsymbol{\beta}})^{-1}\mathbf{S}(\overline{\mathbf{G}}(\widehat{\boldsymbol{\beta}})')^{-1}$$

If we choose the weight matrix to be the inverse of the covariance matrix of the moment conditions so that $\mathbf{W} = \mathbf{S}^{-1}$, then (3) simplifies substantially:

$$\text{Var}(\widehat{\boldsymbol{\beta}}) = \frac{1}{N}\left\{\overline{\mathbf{G}}(\widehat{\boldsymbol{\beta}})'\mathbf{W}\overline{\mathbf{G}}(\widehat{\boldsymbol{\beta}})\right\}^{-1} \tag{4}$$

The GMM estimator constructed using this choice of weight matrix along with the covariance matrix in (4) is known as the "optimal" GMM estimator. One can show that if in fact $\mathbf{W} = \mathbf{S}^{-1}$, then the variance in (4) is smaller than the variance in (3) of any other GMM estimator based on the same moment conditions but with a different choice of weight matrix. Thus, the optimal GMM estimator is also known as the efficient GMM estimator because it has the smallest variance of any estimator based on the given moment conditions.

To obtain standard errors from `gmm` based on the optimal GMM estimator, you specify the `vce(unadjusted)` option. We call that VCE unadjusted because we do not recompute the residuals after estimation to obtain the matrix $\mathbf{S}$ required in (3) or allow for the fact that those residuals may not be i.i.d. Some statistical packages by default report standard errors based on (4) and offer standard errors based on (3) only as an option or not at all. While the optimal GMM estimator is theoretically appealing, Cameron and Trivedi (2005, 177) suggest that in finite samples, it need not perform better than the GMM estimator that uses (3) to obtain standard errors.

❑ Technical note

Computing the covariance matrix of the parameters after using the one-step estimator is actually a bit more complex than after using the two-step or iterative estimator. We can illustrate most of the intricacies by using linear regression with moment conditions of the form $E\{\mathbf{x}(y - \mathbf{x}'\boldsymbol{\beta})\} = \mathbf{0}$.

If you specify `winitial(unadjusted)` and `vce(unadjusted)`, then the initial weight matrix will be computed as

$$\widehat{\mathbf{W}}_1 = \left(\frac{1}{N}\sum_i \mathbf{x}_i\mathbf{x}_i'\right)^{-1} \tag{5}$$

Moreover, for linear regression, we can show that

$$\overline{\mathbf{G}}(\widehat{\boldsymbol{\beta}}) = \frac{1}{N}\sum_i \mathbf{x}_i\mathbf{x}_i'$$

so that (4) becomes

$$\begin{aligned}
\mathrm{Var}(\widehat{\boldsymbol{\beta}}) &= \frac{1}{N}\left\{\left(\frac{1}{N}\sum_i \mathbf{x}_i\mathbf{x}_i'\right)\left(\frac{1}{N}\sum_i \mathbf{x}_i\mathbf{x}_i'\right)^{-1}\left(\frac{1}{N}\sum_i \mathbf{x}_i\mathbf{x}_i'\right)\right\}^{-1} \\
&= \left(\sum_i \mathbf{x}_i\mathbf{x}_i'\right)^{-1} \\
&= (\mathbf{X}'\mathbf{X})^{-1}
\end{aligned} \tag{6}$$

However, we know that the nonrobust covariance matrix for the OLS estimator is actually $\widehat{\sigma}^2(\mathbf{X}'\mathbf{X})^{-1}$. What is missing from (6) is the scalar $\widehat{\sigma}^2$, the estimated variance of the residuals. When you use the one-step estimator and specify `winitial(unadjusted)`, the weight matrix (5) does not include the $\widehat{\sigma}^2$ term because `gmm` does not have a consistent estimate of $\boldsymbol{\beta}$ from which it can then estimate σ^2. The point estimates are still correct because multiplying the weight matrix by a scalar factor does not affect the solution to the minimization problem.

To circumvent this issue, if you specify `winitial(unadjusted)` and `vce(unadjusted)`, `gmm` uses the estimated $\widehat{\boldsymbol{\beta}}$ (which is consistent) to obtain a new unadjusted weight matrix that does include the term $\widehat{\sigma}^2$ so that evaluating (4) will yield correct standard errors.

If you use the two-step or iterated GMM estimator, this extra effort is not needed to obtain standard errors because the first-step (and subsequent steps') estimate of $\boldsymbol{\beta}$ is consistent and can be used to estimate σ^2 or some other weight matrix based on the `wmatrix()` option. Straightforward algebra shows that this extra effort is also not needed if you request any type of adjusted (robust) covariance matrix with the one-step estimator.

A similar issue arises when you specify `winitial(identity)` and `vce(unadjusted)` with the one-step estimator. Again the solution is to compute an unadjusted weight matrix after obtaining $\widehat{\beta}$ so that (4) provides the correct standard errors.

We have illustrated the problem and solution using a single-equation linear model. However, the problem arises whenever you use the one-step estimator with an unadjusted VCE, regardless of the number of equations, and `gmm` handles all the details automatically. Computation of Hansen's J statistic presents an identical issue, and `gmm` takes care of that as well.

If you supply your own initial weight matrix by using `winitial(`*matname*`)`, then the standard errors (as well as the J statistic reported by `estat overid`) are based on that weight matrix. You should verify that the weight matrix you provide will yield appropriate statistics.

❑

Factor-variable coefficients in multiple residual functions

The long example in this section uses `gmm` to replicate the results produced by `regress` with factor variables and `margins`. It illustrates how to refer to the coefficients on factor variables in linear combinations in subsequent residual functions. The example also shows how to use `gmm` to address the two-step estimation problem, or the inconsistency of standard errors produced by two-step estimators that depend on previously estimated parameters.

▷ Example 4: Means of linear combinations of factor variables

The mean of a variable when everyone in a population receives a given treatment level is known as a potential-outcome mean. We use `regress` and `margins` to estimate the potential-outcome means of a mother's smoking behavior while pregnant on the birthweight of her baby after controlling for the mother's age and an indicator for whether the mother had a prenatal visit in the first trimester. We use an extract of data from Cattaneo (2010) in which `bweight` is the baby's birthweight in grams, `mbsmoke` is a binary variable indicating whether a mother smoked while pregnant, `mage` is the mother's age, and `prenatal1` is a binary variable indicating whether the mother had a prenatal visit in the first trimester.

We use `regress` to estimate the regression coefficients.

```
. use https://www.stata-press.com/data/r17/cattaneo2
(Excerpt from Cattaneo (2010) Journal of Econometrics 155: 138-154)
. regress bweight ibn.mbsmoke ibn.mbsmoke#(c.mage i.prenatal1),
> noconstant vce(robust)
Linear regression                               Number of obs     =      4,642
                                                F(6, 4636)        =   27751.75
                                                Prob > F          =     0.0000
                                                R-squared         =     0.9726
                                                Root MSE          =     565.08
```

bweight	Coefficient	Robust std. err.	t	P>|t|	[95% conf.	interval]
mbsmoke						
Nonsmoker	3073.201	48.68899	63.12	0.000	2977.748	3168.655
Smoker	3217.973	93.637	34.37	0.000	3034.4	3401.546
mbsmoke# c.mage						
Nonsmoker	9.737189	1.825552	5.33	0.000	6.158237	13.31614
Smoker	-4.962403	3.852613	-1.29	0.198	-12.51536	2.590552
mbsmoke# prenatal1						
Nonsmoker # Yes	95.11727	26.82039	3.55	0.000	42.53654	147.698
Smoker#Yes	64.61752	39.72317	1.63	0.104	-13.25879	142.4938

We used factor variables to interact `mbsmoke` with the other covariates to allow for separate coefficients for smoking and nonsmoking mothers.

The postestimation command margins uses the estimated regression coefficients to estimate the potential-outcome means of bweight, first assuming that no mother smoked and then assuming that all mothers smoked.

```
. margins i.mbsmoke, vce(unconditional)
Predictive margins                                      Number of obs = 4,642
Expression: Linear prediction, predict()
```

	Unconditional					
	Margin	std. err.	t	P>|t|	[95% conf.	interval]
mbsmoke						
Nonsmoker	3407.506	9.346894	364.56	0.000	3389.181	3425.83
Smoker	3138.23	21.20463	148.00	0.000	3096.659	3179.801

Note that the standard errors for the estimated means account for the estimation error in the estimated coefficients used to compute them.

Before using gmm to simultaneously estimate the regression coefficients and the potential-outcome means, we demonstrate the equivalence of point estimates from gmm and regress and illustrate the two-step estimation problem. First, we use gmm to estimate just the regression coefficients. Note that we specify the factor variables in the linear combination xb: and in the instrument list.

```
. gmm (eq1: bweight - {xb:ibn.mbsmoke ibn.mbsmoke#(c.mage i.prenatal1)}),
> instruments(eq1: ibn.mbsmoke ibn.mbsmoke#(c.mage i.prenatal1), noconstant)
> coeflegend onestep
Step 1
Iteration 0:   GMM criterion Q(b) =   11316945
Iteration 1:   GMM criterion Q(b) =  7.143e-19
Iteration 2:   GMM criterion Q(b) =  2.051e-26
note: model is exactly identified.
GMM estimation
Number of parameters =   6
Number of moments    =   6
Initial weight matrix: Unadjusted                 Number of obs   =      4,642
```

	Coefficient	Legend
mbsmoke		
Nonsmoker	3073.201	_b[0bn.mbsmoke]
Smoker	3217.973	_b[1.mbsmoke]
mbsmoke# c.mage		
Nonsmoker	9.737189	_b[0bn.mbsmoke#c.mage]
Smoker	-4.962403	_b[1.mbsmoke#c.mage]
mbsmoke# prenatal1		
Nonsmoker # Yes	95.11727	_b[0bn.mbsmoke#1.prenatal1]
Smoker#Yes	64.61752	_b[1.mbsmoke#1.prenatal1]

```
Instruments for equation eq1: 0.mbsmoke 1.mbsmoke 0.mbsmoke#c.mage
    1.mbsmoke#c.mage 0o.mbsmoke#0b.prenatal1 0.mbsmoke#1.prenatal1
    1o.mbsmoke#0b.prenatal1 1.mbsmoke#1.prenatal1
```

We specified the coeflegend option to learn the names of the coefficients on the factor variables in the linear combination. As expected, the point estimates are the same as those reported by regress.

Next, we illustrate the effect of the two-step estimation problem if we calculate the potential-outcome means by hand. We can calculate the mean when no mothers smoke by accessing these coefficients and then estimate the standard errors in the estimated means:

```
. generate mean0 = _b[xb:0.mbsmoke] + _b[xb:0.mbsmoke#c.mage]*mage
> + _b[xb:0.mbsmoke#1.prenatal1]*prenatal1
. mean mean0
Mean estimation                         Number of obs = 4,642
```

	Mean	Std. err.	[95% conf.	interval]
mean0	3407.506	1.085503	3405.378	3409.634

The estimated potential-outcome mean for no mothers smoking is the same as that reported by `margins`, but the standard error is much smaller because `mean` ignores the estimation error in the coefficients. This underscores the importance of accounting for the estimation error when estimating the standard errors.

Now, we use `gmm` to estimate the coefficients and the potential-outcome means simultaneously.

```
. gmm (eq1: bweight - {xb:ibn.mbsmoke ibn.mbsmoke#(c.mage i.prenatal1)})
>     (eq2: {xb:0.mbsmoke} + {xb:0bn.mbsmoke#c.mage}*mage
>         + {xb:0bn.mbsmoke#1.prenatal1}*1.prenatal1 - {m0})
>     (eq3: {xb:1.mbsmoke} + {xb:1.mbsmoke#c.mage}*mage
>         + {xb:1.mbsmoke#1.prenatal1}*1.prenatal1 - {m1}),
>     instruments(eq1: ibn.mbsmoke ibn.mbsmoke#(c.mage i.prenatal1),
>        noconstant)
>     instruments(eq2 eq3:) winitial(identity) onestep
Step 1
Iteration 0:   GMM criterion Q(b) =  5.819e+09
Iteration 1:   GMM criterion Q(b) =  3.108e-13
Iteration 2:   GMM criterion Q(b) =  1.010e-22
note: model is exactly identified.
GMM estimation
Number of parameters =   8
Number of moments    =   8
Initial weight matrix: Identity                  Number of obs   =      4,642
```

	Coefficient	Robust std. err.	z	P>|z|	[95% conf.	interval]
mbsmoke						
Nonsmoker	3073.201	48.65751	63.16	0.000	2977.834	3168.568
Smoker	3217.973	93.57647	34.39	0.000	3034.566	3401.379
mbsmoke# c.mage						
Nonsmoker	9.737189	1.824372	5.34	0.000	6.161485	13.31289
Smoker	-4.962403	3.850123	-1.29	0.197	-12.5085	2.583699
mbsmoke# prenatal1						
Nonsmoker # Yes	95.11727	26.80305	3.55	0.000	42.58425	147.6503
Smoker#Yes	64.61752	39.69749	1.63	0.104	-13.18813	142.4232
/m0	3407.506	9.340852	364.80	0.000	3389.198	3425.813
/m1	3138.23	21.19093	148.09	0.000	3096.696	3179.763

```
Instruments for equation eq1: 0.mbsmoke 1.mbsmoke 0.mbsmoke#c.mage
    1.mbsmoke#c.mage 0o.mbsmoke#0b.prenatal1 0.mbsmoke#1.prenatal1
    1o.mbsmoke#0b.prenatal1 1.mbsmoke#1.prenatal1
Instruments for equation eq2: _cons
Instruments for equation eq3: _cons
```

This output has five noteworthy features.

1. We specify three different residual equations. The first, `eq1:`, defines the moment conditions for the regression using the covariates as instruments; `eq2:` is the moment condition for the potential outcome when no mothers smoke; and `eq3:` is the moment condition for the potential outcome when all mothers smoke.
2. In `eq2:` and `eq3:`, we refer to the individual coefficients on the factor variables in the linear combination `xb:` by enclosing their names in curly braces.
3. The option `instruments()` is repeatable. We first specify that the covariates in the regression are the instruments for the residual equation `eq1:` and then specify that only the unit variable, also known as the constant, is an instrument for each of `eq2:` and `eq3:`.
4. The point estimates and the standard errors match those reported by `regress` and `margins`, after accounting for the small-sample adjustment performed by `regress`.
5. Although the point estimates match, the standard errors reported by `gmm` are much larger than those reported by `mean` because `gmm` takes into account that the regression coefficients are estimated.

◁

Parameter interpretation using margins

In the last section, we demonstrated how to use `gmm` to estimate potential-outcome means in a linear regression model jointly with the coefficients of the models. However, you can also estimate the potential-outcome mean, or any other predictive margins, by using the `margins` command after `gmm`. Using `margins` after `gmm` can allow more flexibility in the predictive margins that we estimate and is also more convenient. See *Obtaining margins of responses* in [R] **margins** for more information about predictive margins.

▷ Example 5: Predicting treatment effects after estimation

In example 4, we used `gmm` to estimate potential-outcome means of a mother's smoking behavior on her baby's birthweight (in grams) after controlling for age and whether she had a prenatal visit in the first trimester. Here we demonstrate how to use `margins` after `gmm` to estimate the potential-outcome means.

First, we load the data and reestimate the regression coefficients.

```
. use https://www.stata-press.com/data/r17/cattaneo2
(Excerpt from Cattaneo (2010) Journal of Econometrics 155: 138-154)
. gmm (eq1: bweight - {xb:ibn.mbsmoke ibn.mbsmoke#(c.mage i.prenatal1)}),
> instruments(eq1: ibn.mbsmoke ibn.mbsmoke#(c.mage i.prenatal1), noconstant)
> onestep
Step 1
Iteration 0:   GMM criterion Q(b) =   11316945
Iteration 1:   GMM criterion Q(b) =  7.143e-19
Iteration 2:   GMM criterion Q(b) =  2.051e-26
note: model is exactly identified.
GMM estimation
Number of parameters =   6
Number of moments    =   6
Initial weight matrix: Unadjusted                 Number of obs   =      4,642
```

	Coefficient	Robust std. err.	z	P>|z|	[95% conf.	interval]
mbsmoke						
Nonsmoker	3073.201	48.65752	63.16	0.000	2977.834	3168.568
Smoker	3217.973	93.57647	34.39	0.000	3034.566	3401.379
mbsmoke# c.mage						
Nonsmoker	9.737189	1.824372	5.34	0.000	6.161484	13.31289
Smoker	-4.962403	3.850123	-1.29	0.197	-12.5085	2.583699
mbsmoke# prenatal1						
Nonsmoker # Yes	95.11727	26.80305	3.55	0.000	42.58425	147.6503
Smoker#Yes	64.61752	39.69749	1.63	0.104	-13.18813	142.4232

```
Instruments for equation eq1: 0.mbsmoke 1.mbsmoke 0.mbsmoke#c.mage
    1.mbsmoke#c.mage 0o.mbsmoke#0b.prenatal1 0.mbsmoke#1.prenatal1
    1o.mbsmoke#0b.prenatal1 1.mbsmoke#1.prenatal1
```

Now, we use `margins` to estimate the potential-outcome means. We specify `vce(unconditional)` to obtain standard errors for the potential-outcome means of the population rather than the sample. When we specify this option, the standard errors for the estimated means will account for the estimation error in the estimated coefficients from `gmm`.

```
. margins i.mbsmoke, vce(unconditional)
Predictive margins                                Number of obs = 4,642
Expression: Linear prediction, predict()
```

	Margin	Unconditional std. err.	z	P>|z|	[95% conf.	interval]
mbsmoke						
Nonsmoker	3407.506	9.341858	364.76	0.000	3389.196	3425.815
Smoker	3138.23	21.19321	148.08	0.000	3096.692	3179.768

Our point estimates of the potential-outcome means exactly match those that appear as `\m0` and `\m1` in example 4. However, the standard errors are slightly higher because `gmm` and `margins` use different values in the denominator for the formula for the robust covariance matrix. `gmm` uses N while `margins`

uses $N-1$, so the standard errors differ by a factor of $\sqrt{\{N/(N-1)\}} = \sqrt{4{,}642/4{,}641} \approx 1.0002$. More details about the calculation of the standard errors are provided in *Marginal predictions with unconditional standard errors* in the *Methods and formulas*.

In addition to potential-outcome means, we can use `margins` to estimate the average treatment effect (ATE) of the mother's smoking behavior on birthweight. We use the contrast operator `r.` to instruct `margins` to difference the potential-outcome means and estimate a treatment effect. We specify the `contrast(nowald)` option to suppress the Wald test that `margins` displays by default for contrasts.

```
. margins r.mbsmoke, vce(unconditional) contrast(nowald)
Contrasts of predictive margins                         Number of obs = 4,642
Expression: Linear prediction, predict()
```

	Contrast	Unconditional std. err.	[95% conf.	interval]
mbsmoke (Smoker vs Nonsmoker)	-269.2759	23.16069	-314.67	-223.8818

The ATE of -269.28 is interpreted as the difference between the average birthweight if all mothers in the population smoked and the average birthweight if all mothers in the population did not smoke. The average birthweight if all mothers were to smoke would be 269.28 grams less than if they did not smoke.

◁

Exponential (Poisson) regression models

Exponential regression models are frequently encountered in applied work. For example, they can be used as alternatives to linear regression models on log-transformed dependent variables, obviating the need for post-hoc transformations to obtain predicted values in the original metric of the dependent variable. When the dependent variable represents a discrete count variable, exponential regression models are also known as Poisson regression models; see Cameron and Trivedi (2013).

For now, we consider models of the form

$$y = \exp(\mathbf{x}'\boldsymbol{\beta}) + u \tag{7}$$

where u is a zero-mean additive error term so that $E(y) = \exp(\mathbf{x}'\boldsymbol{\beta})$. Because the error term is additive, if $\mathbf{x}$ represents strictly exogenous regressors, then we have the population moment condition

$$E[\mathbf{x}\{y - \exp(\mathbf{x}'\boldsymbol{\beta})\}] = \mathbf{0} \tag{8}$$

Moreover, because the number of parameters in the model is equal to the number of instruments, there is no point to using the two-step GMM estimator.

▷ Example 6: Exponential regression

Cameron and Trivedi (2022, 584) fit a model of the number of doctor visits based on whether the patient has private insurance, whether the patient has a chronic disease, gender, and income. Here we fit that model by using `gmm`. To allow for potential excess dispersion, we will obtain a robust VCE matrix, which is the default for `gmm` anyway. We type

```
. use https://www.stata-press.com/data/r17/docvisits
. gmm (docvis - exp({xb:private chronic female income _cons})),
> instruments(private chronic female income) onestep
Step 1
Iteration 0:   GMM criterion Q(b) =  16.853973
Iteration 1:   GMM criterion Q(b) =  2.2706472
Iteration 2:   GMM criterion Q(b) =  .19088097
Iteration 3:   GMM criterion Q(b) =  .00041101
Iteration 4:   GMM criterion Q(b) =  3.939e-09
Iteration 5:   GMM criterion Q(b) =  6.572e-19
note: model is exactly identified.
GMM estimation
Number of parameters =   5
Number of moments    =   5
Initial weight matrix: Unadjusted                 Number of obs   =      4,412
```

	Coefficient	Robust std. err.	z	P>|z|	[95% conf.	interval]
private	.7986654	.1089891	7.33	0.000	.5850507	1.01228
chronic	1.091865	.0559888	19.50	0.000	.9821291	1.201601
female	.4925481	.0585298	8.42	0.000	.3778317	.6072644
income	.003557	.0010824	3.29	0.001	.0014356	.0056784
_cons	-.2297263	.1108607	-2.07	0.038	-.4470093	-.0124434

```
Instruments for equation 1: private chronic female income _cons
```

Our point estimates agree with those reported by Cameron and Trivedi (2022) to at least six significant digits; the small discrepancies are attributable to different optimization techniques and convergence criteria being used by `gmm` and `poisson`. The standard errors differ by a factor of $\text{sqrt}(4412/4411)$ because `gmm` uses N in the denominator of the formula for the robust covariance matrix, while the robust covariance matrix estimator used by `poisson` uses $N - 1$.

◁

❑ Technical note

That the GMM and maximum likelihood estimators of the exponential regression model coincide is not a general property of these two classes of estimators. The maximum likelihood estimator solves the score equations

$$\frac{1}{N}\sum_{i=1}^{N}\frac{\partial \ln \ell_i}{\partial \boldsymbol{\beta}} = \mathbf{0}$$

where ℓ_i is the likelihood for the ith observation. These score equations can be viewed as the sample analogues of the population moment conditions

$$E\left(\frac{\partial \ln \ell_i}{\partial \boldsymbol{\beta}}\right) = \mathbf{0}$$

establishing that maximum likelihood estimators represent a subset of the class of GMM estimators.

For the Poisson model,

$$\ln \ell_i = -\exp(\mathbf{x}_i'\boldsymbol{\beta}) + y_i\mathbf{x}_i'\boldsymbol{\beta} - \ln y_i!$$

so the score equations are

$$\frac{1}{N}\sum_{i=1}^{N}\mathbf{x}_i\left\{y_i-\exp(\mathbf{x}_i'\boldsymbol{\beta})\right\}=\mathbf{0}$$

which are just the sample moment conditions implied by (8) that we used in the previous example. That is why our results using `gmm` match Cameron and Trivedi's (2022) results using `poisson`.

On the other hand, an intuitive set of moment conditions to consider for GMM estimation of a probit model is

$$E[\mathbf{x}\{y-\Phi(\mathbf{x}'\boldsymbol{\beta})\}]=\mathbf{0}$$

where $\Phi()$ is the standard normal cumulative distribution function. Differentiating the likelihood function for the maximum-likelihood probit estimator, we can show that the corresponding score equations are

$$\frac{1}{N}\sum_{i=1}^{N}\left[\mathbf{x}_i\left\{y_i\frac{\phi(\mathbf{x}_i'\boldsymbol{\beta})}{\Phi(\mathbf{x}_i'\boldsymbol{\beta})}-(1-y_i)\frac{\phi(\mathbf{x}_i'\boldsymbol{\beta})}{1-\Phi(\mathbf{x}_i'\boldsymbol{\beta})}\right\}\right]=\mathbf{0}$$

where $\phi()$ is the standard normal density function. These two moment conditions are not equivalent, so the maximum likelihood and GMM probit estimators are distinct.

❑

▷ Example 7: Comparison of GMM and maximum likelihood

Using the automobile dataset, we fit a probit model of `foreign` on `gear_ratio`, `length`, and `headroom` using first the score equations and then the intuitive set of GMM equations. We type

```
. use https://www.stata-press.com/data/r17/auto
(1978 automobile data)
. gmm (foreign*normalden({xb:gear_ratio length headroom _cons})/
> normal({xb:}) - (1-foreign)*normalden({xb:})/(1-normal({xb:}))),
> instruments(gear_ratio length headroom) onestep
  (output omitted)
. estimates store ml
. gmm (foreign - normal({xb:gear_ratio length headroom _cons})),
> instruments(gear_ratio length headroom) onestep
  (output omitted)
. estimates store gmm
. estimates table ml gmm, b se
```

Variable	ml	gmm
gear_ratio	2.9586277	2.8489213
	.64042341	.63570246
length	-.02148933	-.02056033
	.01382043	.01396954
headroom	.01136927	.02240761
	.27278528	.2849891
_cons	-6.0222289	-5.8595615
	3.5594588	3.5188028

Legend: b/se

The coefficients on gear_ratio and length are close for the two estimators. The GMM estimate of the coefficient on headroom is twice that of the maximum likelihood estimate, though the relatively large standard errors imply that this difference is not significant. You can verify that the coefficients in the column marked "ml" match those you would obtain with probit. We have not discussed the differences among standard errors based on the various GMM and maximum-likelihood covariance matrix estimators to avoid tedious algebra. However, you can verify that the robust covariance matrix after one-step GMM estimation differs by only a finite-sample adjustment factor of $(N/N-1)$ from the robust covariance matrix reported by probit. Both the maximum likelihood and GMM probit estimators require the normality assumption, and the maximum likelihood estimator is efficient if that normality assumption is correct; therefore, in this example, there is no reason to prefer the GMM estimator.

◁

We can modify (8) easily to allow for endogenous regressors. Suppose that x_j is endogenous in the sense that $E(u|x_j) \neq 0$. Then, (8) is no longer a valid moment condition. However, suppose we have some variables other than $\mathbf{x}$ such that $E(u|\mathbf{z}) = \mathbf{0}$. We can instead use the moment conditions

$$E(\mathbf{z}u) = E[\mathbf{z}\{y - \exp(\mathbf{x}'\boldsymbol{\beta})\}] = \mathbf{0}$$

As usual, if some elements of $\mathbf{x}$ are exogenous, then they should appear in $\mathbf{z}$ as well.

▷ Example 8: Exponential regression with endogenous regressors

Returning to the model discussed in example 6, we treat income as endogenous; unobservable factors that determine a person's income may also affect the number of times a person visits a doctor. We use a person's age and race as instruments. These are valid instruments if we believe that age and race influence a person's income but do not have a direct impact on the number of doctor visits. (Whether this belief is justified is another matter; we test that belief in [R] **gmm postestimation**.) Because we have more instruments (seven) than parameters (five), we have an overidentified model. Therefore, the choice of weight matrix does matter. We will use the default two-step GMM estimator. In the first step, we will use a weight matrix that assumes the errors are i.i.d. In the second step, we will use a weight matrix that assumes heteroskedasticity. When you specify twostep, these are the defaults for the first- and second-step weight matrices, so we do not have to use the winitial() or wmatrix() options. We will again obtain a robust VCE, which is also the default. We type

```
. use https://www.stata-press.com/data/r17/docvisits
. gmm (docvis - exp({xb:private chronic female income _cons})),
> instruments(private chronic female age black hispanic) twostep
Step 1
Iteration 0:   GMM criterion Q(b) =  16.910173
Iteration 1:   GMM criterion Q(b) =  .82276104
Iteration 2:   GMM criterion Q(b) =  .21832032
Iteration 3:   GMM criterion Q(b) =  .12685935
Iteration 4:   GMM criterion Q(b) =  .12672369
Iteration 5:   GMM criterion Q(b) =  .12672365
Step 2
Iteration 0:   GMM criterion Q(b) =  .00234641
Iteration 1:   GMM criterion Q(b) =  .00215957
Iteration 2:   GMM criterion Q(b) =  .00215911
Iteration 3:   GMM criterion Q(b) =  .00215911
```

```
GMM estimation

Number of parameters =   5
Number of moments    =   7
Initial weight matrix: Unadjusted                 Number of obs   =      4,412
GMM weight matrix:     Robust
```

	Coefficient	Robust std. err.	z	P>|z|	[95% conf.	interval]
private	.535335	.1599039	3.35	0.001	.2219291	.8487409
chronic	1.090126	.0617659	17.65	0.000	.9690668	1.211185
female	.6636579	.0959884	6.91	0.000	.4755241	.8517918
income	.0142855	.0027162	5.26	0.000	.0089618	.0196092
_cons	-.5983477	.138433	-4.32	0.000	-.8696713	-.327024

```
Instruments for equation 1: private chronic female age black hispanic _cons
```

Once we control for the endogeneity of income, we find that its coefficient has quadrupled in size. Additionally, access to private insurance has less of an impact on the number of doctor visits and gender has more of an impact.

◁

❑ Technical note

Although you may be tempted to try, you cannot, as you can in a Poisson model, replace $\mathbf{x}$ in the moment conditions for the probit (or logit) model with a vector of instruments, $\mathbf{z}$, if you have endogenous regressors. See Wilde (2008).

❑

Mullahy (1997) considers a slightly more complicated version of the exponential regression model that incorporates nonadditive unobserved heterogeneity. His model can be written as

$$y_i = \exp(\mathbf{x}_i'\boldsymbol{\beta})\eta_i + \epsilon_i$$

where $\eta_i > 0$ is an unobserved heterogeneity term that may be correlated with $\mathbf{x}_i$. One result from his article is that instead of using the additive moment condition (8), we can use the multiplicative moment condition

$$E\left\{\mathbf{z}\frac{y - \exp(\mathbf{x}'\boldsymbol{\beta})}{\exp(\mathbf{x}'\boldsymbol{\beta})}\right\} = E[\mathbf{z}\{y\exp(-\mathbf{x}'\boldsymbol{\beta}) - 1\}] = \mathbf{0}$$

Windmeijer and Santos Silva (1997) discuss the use of additive versus multiplicative moment conditions with endogenous regressors and note that a set of instruments that satisfies the additive moment conditions will not also satisfy the multiplicative moment conditions. They remark that the decision about which to use is an empirical issue that can at least partially be settled by using the test of overidentifying restrictions that is implemented by `estat overid` after `gmm` to see whether the instruments for a given model are valid. See [R] **gmm postestimation** for information on the test of overidentifying restrictions.

Specifying derivatives

By default, gmm calculates derivatives numerically, and the method used produces accurate results for the vast majority of applications. However, if you refit the same model repeatedly or else have the derivatives available, then gmm will run more quickly if you supply it with analytic derivatives.

When you use the interactive version of gmm, you specify derivatives using substitutable expressions in much the same way you specify the residual equations. There are three rules you must follow:

1. As with the substitutable expressions that define residual equations, you bind parameters of the model in curly braces: {b0}, {param}, etc.
2. You must specify a derivative for each parameter that appears in each residual equation. If a parameter does not appear in a residual equation, then you do not specify a derivative for that parameter in that residual equation.
3. If you declare a linear combination in an equation, then you specify a derivative with respect to that linear combination. gmm applies the chain rule to obtain the derivatives with respect to the individual parameters encompassed by that linear combination.

We illustrate with several examples.

▷ Example 9: Derivatives for a single-equation model

Consider a simple exponential regression model with one exogenous regressor and a constant term. We have

$$u_i = y_i - \exp(\beta_0 + \beta_1 x_i)$$

Now,

$$\frac{\partial u_i}{\partial \beta_0} = -\exp(\beta_0 + \beta_1 x_i) \qquad \text{and} \qquad \frac{\partial u_i}{\partial \beta_1} = -x_i \exp(\beta_0 + \beta_1 x_i)$$

In Stata, we type

```
. gmm (docvis - exp({b0} + {b1}*income)), instruments(income)
> deriv(/b0 = -1*exp({b0} + {b1}*income))
> deriv(/b1 = -1*income*exp({b0}+{b1}*income)) onestep
Step 1
Iteration 0:   GMM criterion Q(b) =  9.1548611
Iteration 1:   GMM criterion Q(b) =  3.5146131
Iteration 2:   GMM criterion Q(b) =  .01344695
Iteration 3:   GMM criterion Q(b) =  3.690e-06
Iteration 4:   GMM criterion Q(b) =  4.606e-13
Iteration 5:   GMM criterion Q(b) =  1.502e-26
note: model is exactly identified.
GMM estimation
Number of parameters =   2
Number of moments    =   2
Initial weight matrix: Unadjusted                 Number of obs   =      4,412
```

	Coefficient	Robust std. err.	z	P>|z|	[95% conf.	interval]
/b0	1.204888	.0462355	26.06	0.000	1.114268	1.295507
/b1	.0046702	.0009715	4.81	0.000	.0027662	.0065743

```
Instruments for equation 1: income _cons
```

Notice how we specified the `derivative()` option for each parameter. We simply specified a slash, the name of the parameter, an equal sign, then a substitutable expression that represents the derivative. Because our model has only one residual equation, we do not need to specify equation numbers in the `derivative()` options.

◁

When you specify a linear combination of variables, your derivative should be with respect to the entire linear combination. For example, say we have the residual equation

$$u_i = y - \exp(\mathbf{x}_i'\boldsymbol{\beta} + \beta_0)$$

for which we would type

```
. gmm (y - exp({xb: x1 x2 x3} + {b0}) ...
```

Then, in addition to the derivative $\partial u_i/\partial\beta_0$, we are to compute and specify

$$\frac{\partial u_i}{\partial \mathbf{x}_i'\boldsymbol{\beta}} = -\exp(\mathbf{x}_i'\boldsymbol{\beta} + \beta_0)$$

Using the chain rule, $\partial u_i/\partial\beta_j = \partial u_i/\partial(\mathbf{x}_i'\boldsymbol{\beta}) \times \partial(\mathbf{x}_i'\boldsymbol{\beta})/\partial\beta_j = -x_{ij}\exp(\mathbf{x}_i'\boldsymbol{\beta}+\beta_0)$. Stata does this last calculation automatically. It knows the variables in the linear combination, so all it needs is the derivative of the residual equation with respect to the linear combination. This allows you to change the variables in your linear combination without having to change the derivatives.

▷ Example 10: Derivatives with a linear combination

We refit the model described in the example illustrating exponential regression with endogenous regressors, now providing analytic derivatives. We type

```
. gmm (docvis - exp({xb:private chronic female income _cons})),
> instruments(private chronic female age black hispanic)
> derivative(/xb = -1*exp({xb:}))
Step 1
Iteration 0:   GMM criterion Q(b) =  16.910173
Iteration 1:   GMM criterion Q(b) =  .82270871
Iteration 2:   GMM criterion Q(b) =  .21831995
Iteration 3:   GMM criterion Q(b) =  .12685934
Iteration 4:   GMM criterion Q(b) =  .12672369
Iteration 5:   GMM criterion Q(b) =  .12672365
Step 2
Iteration 0:   GMM criterion Q(b) =  .00234641
Iteration 1:   GMM criterion Q(b) =  .00215957
Iteration 2:   GMM criterion Q(b) =  .00215911
Iteration 3:   GMM criterion Q(b) =  .00215911
GMM estimation
Number of parameters =   5
Number of moments    =   7
Initial weight matrix: Unadjusted                 Number of obs   =      4,412
GMM weight matrix:     Robust
```

	Coefficient	Robust std. err.	z	P>|z|	[95% conf.	interval]
private	.535335	.159904	3.35	0.001	.221929	.848741
chronic	1.090126	.0617659	17.65	0.000	.9690668	1.211185
female	.6636579	.0959885	6.91	0.000	.475524	.8517918
income	.0142855	.0027162	5.26	0.000	.0089618	.0196092
_cons	-.5983477	.138433	-4.32	0.000	-.8696714	-.327024

```
Instruments for equation 1: private chronic female age black hispanic _cons
```

In the first `derivative()` option, we specified the name of the linear combination, `xb`, instead of an individual parameter's name. We already declared the variables of our linear combination in the substitutable expression for the residual equation, so in our substitutable expressions for the derivatives, we can use the shorthand notation `{xb:}` to refer to it.

Our point estimates are identical to those we obtained earlier. The standard errors and confidence intervals differ by only trivial amounts.

◁

Exponential regression models with panel data

In addition to supporting cross-sectional and time-series data, `gmm` also works with panel-data models. Here we illustrate `gmm`'s panel-data capabilities by expanding our discussion of exponential regression models to allow for panel data. This also provides us the opportunity to demonstrate the moment-evaluator program version of `gmm`. Our discussion is based on Blundell, Griffith, and Windmeijer (2002). Also see Wooldridge (1999) for further discussion of nonlinear panel-data models.

First, we expand (7) for panel data. With individual heterogeneity term η_i, we have

$$E(y_{it}|\mathbf{x}_{it}, \eta_i) = \exp(\mathbf{x}'_{it}\boldsymbol{\beta} + \eta_i) = \mu_{it}\nu_i$$

where $\mu_{it} = \exp(\mathbf{x}'_{it}\boldsymbol{\beta})$ and $\nu_i = \exp(\eta_i)$. Note that there is no constant term in this model, because its effect cannot be disentangled from ν_i. With an additive idiosyncratic error term, we have the regression model

$$y_{it} = \mu_{it}\nu_i + \epsilon_{it}$$

We do not impose the assumption $E(\mathbf{x}_{it}\eta_i) = \mathbf{0}$, so η_i can be considered a fixed effect in the sense that it may be correlated with the regressors.

As discussed by Blundell, Griffith, and Windmeijer (2002), if $\mathbf{x}_{it}$ is strictly exogenous, meaning $E(\mathbf{x}_{it}\epsilon_{is}) = \mathbf{0}$ for all t and s, then we can estimate the parameters of the model by using the sample moment conditions

$$\sum_i \sum_t \mathbf{x}_{it} \left(y_{it} - \mu_{it} \frac{\overline{y}_i}{\overline{\mu}_i} \right) = \mathbf{0} \tag{9}$$

where $\overline{y}_i$ and $\overline{\mu}_i$ are the means of y_{it} and μ_{it} for panel i, respectively. Because $\overline{\mu}_i$ depends on the parameters of the model, it must be recomputed each time `gmm` needs to evaluate the residual equation. Therefore, we cannot use the substitutable expression version of `gmm`. Instead, we must use the moment-evaluator program version.

The moment-evaluator program version of `gmm` functions much like the function-evaluator program versions of `nl` and `nlsur`. The program you write is passed one or more variables to be filled in with the residuals evaluated at the parameter values specified in an option passed to your program. For the fixed-effects Poisson model with strictly exogenous regressors, our first crack at a function-evaluator program is

```
program gmm_poi
        version 17.0
        syntax varlist if, at(name)
        quietly {
                tempvar mu mubar ybar
                generate double `mu' = exp(x1*`at'[1,1] + x2*`at'[1,2]    ///
                                                  + x3*`at'[1,3]) `if'
                egen double `mubar' = mean(`mu') `if', by(id)
                egen double `ybar' = mean(y) `if', by(id)
                replace `varlist' = y - `mu'*`ybar'/`mubar' `if'
        }
end
```

You can save your program in an ado-file named *name*`.ado`, where *name* is the name you use for your program; here we would save the program in the ado-file `gmm_poi.ado`. Alternatively, if you are working from within a do-file, you can simply define the program before calling `gmm`. The `syntax` statement declares we are expecting to receive a *varlist* that will contain the names of variables whose values we are to replace with the values of the residual equations and an `if` expression that will mark the estimation sample; because our model has one residual equation, *varlist* will consist of one variable. `at()` is a required option to our program, and it will contain the name of a matrix containing the parameter values at which we are to evaluate the residual equation. All moment-evaluator programs must accept the *varlist*, `if` condition, and `at()` option.

The first part of our program computes μ_{it}. In the model we will fit shortly, we have three regressors, named `x1`, `x2`, and `x3`. The `` `at' `` vector will have three elements, one for each of those variables. Notice that we included `` `if' `` at the end of each statement that affects variables to restrict the computations to the relevant estimation sample. The two `egen` statements compute $\overline{\mu}_i$ and $\overline{y}_i$; in the example dataset we will use shortly, the panel variable is named `id`, and for simplicity, we hardcoded that variable into our program as well. Finally, we compute the residual equation, which is the portion of (9) bound in parentheses.

▷ Example 11: Panel Poisson with strictly exogenous regressors

To fit our model, we type

```
. use https://www.stata-press.com/data/r17/poisson1
. gmm gmm_poi, nequations(1) parameters(b1 b2 b3)
> instruments(x1 x2 x3, noconstant) vce(cluster id) onestep
Step 1
Iteration 0:   GMM criterion Q(b) =   51.99142
Iteration 1:   GMM criterion Q(b) =  .04345191
Iteration 2:   GMM criterion Q(b) =  8.720e-06
Iteration 3:   GMM criterion Q(b) =  7.115e-13
Iteration 4:   GMM criterion Q(b) =  5.130e-27
note: model is exactly identified.
GMM estimation
Number of parameters =   3
Number of moments    =   3
Initial weight matrix: Unadjusted                 Number of obs   =        409
                                  (Std. err. adjusted for 45 clusters in id)
------------------------------------------------------------------------------
             |               Robust
             | Coefficient  std. err.      z    P>|z|     [95% conf. interval]
-------------+----------------------------------------------------------------
         /b1 |     1.94866   .1000265    19.48   0.000     1.752612    2.144709
         /b2 |    -2.966119  .0923592   -32.12   0.000    -3.14714    -2.785099
         /b3 |     1.008634  .1156561     8.72   0.000      .781952    1.235315
------------------------------------------------------------------------------
Instruments for equation 1: x1 x2 x3
```

All three of our regressors are strictly exogenous, so they can serve as their own regressors. There is no constant term in the model (it would be unidentified), so we exclude a constant term from our list of instruments. We have one residual equation as indicated by `nequations(1)`, and we have three parameters, named `b1`, `b2`, and `b3`. The order in which you declare parameters in the `parameters()` option determines the order in which they appear in the `'at'` vector in the moment-evaluator program. We specified `vce(cluster id)` to obtain standard errors that allow for correlation among observations within each panel.

◁

The program we just wrote is sufficient to fit the model to the `poisson1` dataset, but if we want to fit that model to other datasets, we need to change the variable names and perhaps account for having a different number of parameters as well. Despite those limitations, if you just want to fit a single model, that program is adequate.

Next, we take advantage of the ability to specify full equation names in the `parameters()` option and rewrite our evaluator program so that we can more easily change the variables in our model. This approach is particularly useful if some of the residual equations are linear in the parameters because then we can use `matrix score` (see [P] **matrix score**) to evaluate those moments.

Our new evaluator program is

```
program gmm_poieq
        version 17.0
        syntax varlist if, at(name)
        quietly {
                tempvar mu mubar ybar
                matrix score double `mu' = `at' `if', eq(#1)
                replace `mu' = exp(`mu')
                egen double `mubar' = mean(`mu') `if', by(id)
                egen double `ybar' = mean(y) `if', by(id)
                replace `varlist' = y - `mu'*`ybar'/`mubar' `if'
        }
end
```

Rather than using `generate` to compute the temporary variable `` `mu' ``, we used `matrix score` to obtain the linear combination $\mathbf{x}_{it}'\boldsymbol{\beta}$ and then called `replace` to compute $\exp(\mathbf{x}_{it}'\boldsymbol{\beta})$.

▷ Example 12: Panel Poisson using matrix score

To fit our model, we type

```
. use https://www.stata-press.com/data/r17/poisson1
. gmm gmm_poieq, nequations(1) parameters({y:x1 x2 x3})
> instruments(x1 x2 x3, noconstant) vce(cluster id) onestep
Step 1
Iteration 0:   GMM criterion Q(b) =   51.99142
Iteration 1:   GMM criterion Q(b) =  .04345191
Iteration 2:   GMM criterion Q(b) =  8.720e-06
Iteration 3:   GMM criterion Q(b) =  7.115e-13
Iteration 4:   GMM criterion Q(b) =  5.130e-27
note: model is exactly identified.
GMM estimation
Number of parameters =   3
Number of moments    =   3
Initial weight matrix: Unadjusted                 Number of obs   =        409
                                  (Std. err. adjusted for 45 clusters in id)
------------------------------------------------------------------------------
             |               Robust
             | Coefficient  std. err.      z    P>|z|     [95% conf. interval]
-------------+----------------------------------------------------------------
          x1 |     1.94866   .1000265    19.48   0.000     1.752612    2.144709
          x2 |  -2.966119   .0923592   -32.12   0.000    -3.14714    -2.785099
          x3 |   1.008634   .1156561     8.72   0.000      .781952    1.235315
------------------------------------------------------------------------------
Instruments for equation 1: x1 x2 x3
```

Instead of specifying simple parameter names in the `parameters()` option, we specified a linear combination name and the variables associated with that combination. We named our linear combination `y`, but you could use any valid Stata name. When we use this syntax, the rows of the coefficient table are grouped by the equation names.

Say we wanted to refit our model using just `x1` and `x3` as regressors. We do not need to make any changes to `gmm_poieq`. We just change the specification of the `parameters()` option:

```
. gmm gmm_poieq, nequations(1) parameters({y:x1 y:x3})
> instruments(x1 x3, noconstant) vce(cluster id) onestep
```

In this evaluator program, we have still hardcoded the name of the dependent variable. The next two examples include methods to tackle that shortcoming.

◁

❑ Technical note

Say we specify the `parameters()` option like this:

```
. gmm ..., parameters({y1:x1 x2 _cons} {y2:_cons} {y3:x1 _cons})
```

Then, the `‘at’` vector passed to our program will have the following column names attached to it:

```
‘at’[1,6]
      y1:    y1:    y1:    y2:    y3:    y3:
      x1     x2  _cons  _cons     x1  _cons
```

Typing

```
. matrix score double eq1 = ‘at’, eq(#1)
```

is equivalent to typing

```
. generate double eq1 = x1*‘at’[1,1] + x2*‘at’[1,2] + ‘at’[1,3]
```

with one important difference. If we change some of the variables in the `parameters()` option when we call `gmm`, `matrix score` will compute the correct linear combination. If we were to use the `generate` statement instead, then every time we wanted to change the variables in our model, we would have to modify that statement as well.

The command

```
. matrix score double alpha = ‘at’, eq(#2) scalar
```

is equivalent to

```
. scalar alpha = ‘at’[1,4]
```

Thus, even if you specify linear combination and variable names in the `parameters()` option, you can still have scalar parameters in your model.

❑

When past values of the idiosyncratic error term affect the value of a regressor, we say that regressor is predetermined. When one or more regressors are predetermined, sample moment condition (8) is no longer valid. However, Chamberlain (1992) shows that a simple alternative is to consider moment conditions of the form

$$\sum_i \sum_{t=2}^{T} \mathbf{x}_{i,t-1} \left(y_{i,t-1} - \mu_{i,t-1} \frac{y_{it}}{\mu_{it}} \right) = \mathbf{0} \tag{10}$$

Also see Wooldridge (1997) and Windmeijer (2000) for other moment conditions that can be used with predetermined regressors.

▷ Example 13: Panel Poisson with predetermined regressors

Here we refit the previous model, treating all the regressors as predetermined and using the moment conditions in (10). Our moment-evaluator program is

```
program gmm_poipre
        version 17.0
        syntax varlist if, at(name) mylhs(varlist)
        quietly {
                tempvar mu
                matrix score double `mu' = `at' `if', eq(#1)
                replace `mu' = exp(`mu')
                replace `varlist' = L.`mylhs' - L.`mu'*`mylhs'/`mu' `if'
        }
end
```

To compute the residual equation, we used lag-operator notation so that Stata properly handles gaps in our panel dataset. We also made our program accept an additional option that we will use to pass in the dependent variable. When we specify this option in our `gmm` statement, it will get passed to our evaluator program because `gmm` will not recognize the option as one of its own. Equation (10) shows that we are to use the first lags of the regressors as instruments, so we type

```
. gmm gmm_poipre, mylhs(y) nequations(1)  vce(cluster id) onestep
> parameters({y:x1 x2 x3}) instructments(L.(x1 x2 x3), noconstant)
note: 45 missing values returned for equation 1 at initial values.

Step 1
Iteration 0:   GMM criterion Q(b) =  76.652367
Iteration 1:   GMM criterion Q(b) =  1.9118192
Iteration 2:   GMM criterion Q(b) =  .06634724
Iteration 3:   GMM criterion Q(b) =   .0000233
Iteration 4:   GMM criterion Q(b) =  2.998e-12
Iteration 5:   GMM criterion Q(b) =  4.834e-26

note: model is exactly identified.

GMM estimation

Number of parameters =   3
Number of moments    =   3
Initial weight matrix: Unadjusted                 Number of obs   =        364
                                  (Std. err. adjusted for 45 clusters in id)
------------------------------------------------------------------------------
             |               Robust
             | Coefficient  std. err.      z    P>|z|     [95% conf. interval]
-------------+----------------------------------------------------------------
          x1 |    2.088246   .2513626     8.31   0.000     1.595584    2.580907
          x2 |   -2.905504   .2133908   -13.62   0.000    -3.323742   -2.487266
          x3 |    1.121081    .201654     5.56   0.000     .7258459    1.516315
------------------------------------------------------------------------------
Instruments for equation 1: L.x1 L.x2 L.x3
```

Here, like earlier with strictly exogenous regressors, the number of instruments equals the number of parameters, so there is no gain to using the two-step or iterated estimator. However, if you do have more instruments than parameters, you will most likely want to use one of those other estimators instead.

The note at the top of the output is given because we have 45 panels in our dataset. Our residual equation includes lagged terms and therefore cannot be evaluated for the first time period within each panel. Notes like this can be ignored once you know why they occurred. If you receive a note that you were not expecting, you should first investigate the cause of the note before trusting the results.

Instead of making our program accept the `mylhs()` option, we could have used Stata's `coleq` macro function to determine the dependent variable based on the column names attached to the `'at'` vector; see [P] **macro**. Then, we could refit our model with a different dependent variable by changing the *lcname* used in the `parameters()` option. In the next example, we take this approach.

◁

In the previous example, we used $\mathbf{x}_{i,t-1}$ as instruments. A more efficient GMM estimator would also use $\mathbf{x}_{i,t-2}, \mathbf{x}_{i,t-3}, \ldots, \mathbf{x}_{i,1}$ as instruments in period t as well. `gmm`'s `xtinstruments()` option allows you to specify instrument lists that grow as t increases. Later, we discuss the `xtinstruments()` option in detail in the context of linear dynamic panel-data models.

When a regressor is contemporaneously correlated with the idiosyncratic error term, we say that regressor is endogenous. Windmeijer (2000) shows that we can use the moment condition

$$\sum_i \sum_{t=3}^{T} \mathbf{x}_{i,t-2} \left(\frac{y_{it}}{\mu_{it}} - \frac{y_{i,t-1}}{\mu_{i,t-1}} \right)$$

Here we use the second lag of the endogenous regressor as an instrument. If a variable is strictly exogenous, it can of course serve as its own instrument.

▷ Example 14: Panel Poisson with endogenous regressors

Here we refit the model, treating `x3` as endogenous and `x1` and `x2` as strictly exogenous. Our moment-evaluator program is

```
program gmm_poiend
        version 17.0
        syntax varlist if, at(name)
        quietly {
                tempvar mu
                matrix score double `mu' = `at' `if', eq(#1)
                replace `mu' = exp(`mu')
                local mylhs : coleq `at'
                local mylhs : word 1 of `mylhs'
                replace `varlist' = `mylhs'/`mu' - L.`mylhs'/L.`mu' `if'
        }
end
```

Now, we call gmm using x1, x2, and L2.x3 as instruments:

```
. use https://www.stata-press.com/data/r17/poisson2
. gmm gmm_poiend, nequations(1) vce(cluster id) onestep
> parameters(y:x1 y:x2 y:x3) instruments(x1 x2 L2.x3, noconstant)
note: 500 missing values returned for equation 1 at initial values.
Step 1
Iteration 0:   GMM criterion Q(b) =  61.832288
Iteration 1:   GMM criterion Q(b) =  .03402584
Iteration 2:   GMM criterion Q(b) =  .01101288
Iteration 3:   GMM criterion Q(b) =  6.339e-06
Iteration 4:   GMM criterion Q(b) =  1.620e-12
Iteration 5:   GMM criterion Q(b) =  1.312e-25
note: model is exactly identified.
GMM estimation
Number of parameters =   3
Number of moments    =   3
Initial weight matrix: Unadjusted                 Number of obs   =      3,766
                                  (Std. err. adjusted for 500 clusters in id)
------------------------------------------------------------------------------
             |               Robust
             | Coefficient  std. err.      z    P>|z|     [95% conf. interval]
-------------+----------------------------------------------------------------
          x1 |      1.8141   .2688318     6.75   0.000       1.2872    2.341001
          x2 |   -2.982671   .1086666   -27.45   0.000    -3.195653   -2.769688
          x3 |    4.126518   6.369334     0.65   0.517    -8.357147    16.61018
------------------------------------------------------------------------------
Instruments for equation 1: x1 x2 L2.x3
```

The note at the top of the output is given because that we have 500 panels in our dataset. As in the previous example, our residual equation includes lagged terms and therefore cannot be evaluated for the first time period within each panel. Instead of using just $\mathbf{x}_{i,t-2}$ as an instrument, we could use all further lags of $\mathbf{x}_{it}$ as instruments as well.

◁

Rational-expectations models

Macroeconomic models typically assume that agents' expectations about the future are formed rationally. By rational expectations, we mean that agents use all information available when forming their forecasts, so the forecast error is uncorrelated with the information available when the forecast was made. Say that at time t, people make a forecast, $\widehat{y}_{t+1}$, of variable y in the next period. If Ω_t denotes all available information at time t, then rational expectations implies that $E\{(\widehat{y}_{t+1} - y_{t+1})|\Omega_t\} = 0$. If Ω_t denotes observable variables such as interest rates or prices, then this conditional expectation can serve as the basis of a moment condition for GMM estimation.

▷ Example 15: Fitting a Euler equation

In a well-known article, Hansen and Singleton (1982) consider a model of portfolio decision making and discuss parameter estimation using GMM. We will consider a simple example with one asset in which the agent can invest. A consumer wants to maximize the present value of his or her lifetime utility derived from buying a good. On the one hand, the consumer is impatient, so he or she would rather buy today than wait until tomorrow. On the other hand, by buying less today, the consumer can invest more money, earning more interest that can be used to buy more of the good tomorrow. Thus, there is a tradeoff between having cake today or sacrificing a bit today to have more cake tomorrow.

If we assume a specific form for the agent's utility function, known as the constant relative-risk aversion utility function, we can show that the Euler equation is

$$E\left[\mathbf{z}_t\left\{1-\beta(1+r_{t+1})(c_{t+1}/c_t)^{-\gamma}\right\}\right]=\mathbf{0}$$

where β and γ are the parameters to estimate, r_t is the return to the financial asset, and c_t is consumption in period t. β measures the agent's discount factor. If β is near 1, the agent is patient and is more willing to forgo consumption this period. If β is close to 0, the agent is less patient and prefers to consume more now. The parameter γ characterizes the agent's utility function. If $\gamma=0$, the utility function is linear. As γ tends toward 1, the utility function tends toward $u=\log(c)$.

We have data on three-month Treasury bills (r_t) and consumption expenditures (c_t). As instruments, we will use lagged rates of return and past growth rates of consumption. We will use the two-step estimator and a weight matrix that allows for heteroskedasticity and autocorrelation up to four lags with the Bartlett kernel. In Stata, we type

```
. use https://www.stata-press.com/data/r17/cr
. generate cgrowth = c / L.c
(1 missing value generated)
. gmm (1 - {b=1}*(1+F.r)*(F.c/c)^(-1*{gamma=1})),
> inst(L.r L2.r cgrowth L.cgrowth) wmat(hac nw 4) twostep
note: 1 missing value returned for equation 1 at initial values.
Step 1
Iteration 0:   GMM criterion Q(b) =  .00226482
Iteration 1:   GMM criterion Q(b) =  .00054369
Iteration 2:   GMM criterion Q(b) =  .00053904
Iteration 3:   GMM criterion Q(b) =  .00053904
Step 2
Iteration 0:   GMM criterion Q(b) =   .0600729
Iteration 1:   GMM criterion Q(b) =   .0596369
Iteration 2:   GMM criterion Q(b) =   .0596369
GMM estimation
Number of parameters =   2
Number of moments    =   5
Initial weight matrix: Unadjusted                 Number of obs   =        239
GMM weight matrix:     HAC Bartlett 4
```

	Coefficient	HAC std. err.	z	P>|z|	[95% conf.	interval]
/b	.9204617	.0134646	68.36	0.000	.8940716	.9468518
/gamma	-4.222361	1.473895	-2.86	0.004	-7.111143	-1.333579

```
HAC standard errors based on Bartlett kernel with 4 lags.
Instruments for equation 1: L.r L2.r cgrowth L.cgrowth _cons
```

The note at the top of the output is given because the forward operator in our substitutable expression says that residuals cannot be computed for the last observation. In addition, two observations are omitted because the `L2.r` instrument has missing values in the first two time periods. Therefore, of the 242 observations in our dataset, 239 are used to fit the model. Our estimate of β is near 1, in line with expectations and published results. However, our estimate of γ implies risk-loving behavior and therefore a poorly specified model.

Also notice our use of the forward operator to refer to the values of r and c one period ahead; time-series operators are allowed in substitutable expressions as long as you have previously `tsset` (see [TS] **tsset**) your data. See [U] **13.10 Time-series operators** for more information on time-series operators.

◁

System estimators

In many economic models, two or more variables are determined jointly through a system of simultaneous equations. Indeed, some of the earliest work in econometrics, including that of the Cowles Commission, was centered around estimation of the parameters of simultaneous equations. The 2SLS and IV estimators we have already discussed are used in some circumstances to estimate such parameters. Here we focus on the joint estimation of all the parameters of systems of equations, and we begin with the well-known three-stage least-squares (3SLS) estimator.

Recall that the 2SLS estimator is based on the moment conditions $E(\mathbf{z}u) = \mathbf{0}$. The 2SLS estimator can be used to estimate the parameters of one equation of a system of structural equations. Moreover, with the 2SLS estimator, we do not even need to specify the structural relationship among all the endogenous variables; we need to specify only the equation on which interest focuses and simply assume reduced-form relationships among the endogenous regressors of the equation of interest and the exogenous variables of the model. If we are willing to specify the complete system of structural equations, then assuming our model is correctly specified, by estimating all the equations jointly, we can obtain estimates that are more efficient than equation-by-equation 2SLS.

In [R] **reg3**, we fit a simple two-equation macroeconomic model,

$$\texttt{consump} = \beta_0 + \beta_1\texttt{wagepriv} + \beta_2\texttt{wagegovt} + \epsilon_1 \tag{11}$$

$$\texttt{wagepriv} = \beta_3 + \beta_4\texttt{consump} + \beta_5\texttt{govt} + \beta_6\texttt{capital1} + \epsilon_2 \tag{12}$$

where `consump` represents aggregate consumption; `wagepriv` and `wagegovt` are total wages paid by the private and government sectors, respectively; `govt` is government spending; and `capital1` is the previous period's capital stock. We are not willing to assume that ϵ_1 and ϵ_2 are independent, so we must treat both `consump` and `wagepriv` as endogenous. Suppose that a random shock makes ϵ_2 positive. Then by (12), `wagepriv` will be higher than it otherwise would. Moreover, ϵ_1 will be either higher or lower, depending on the correlation between it and ϵ_2. The shock to ϵ_2 has made both `wagepriv` and ϵ_1 move, which implies that in (11), `wagepriv` is an endogenous regressor. A similar argument shows that `consump` is an endogenous regressor in the second equation. In our model, `wagegovt`, `govt`, and `capital1` are all exogenous variables.

Let $\mathbf{z}_1$ and $\mathbf{z}_2$ denote the instruments for the first and second equations, respectively; we will discuss what comprises them shortly. We have two sets of moment conditions:

$$E\left\{\begin{array}{c}\mathbf{z}_1(\texttt{consump} - \beta_0 - \beta_1\texttt{wagepriv} - \beta_2\texttt{wagegovt}) \\ \mathbf{z}_2(\texttt{wagepriv} - \beta_3 - \beta_4\texttt{consump} - \beta_5\texttt{govt} - \beta_6\texttt{capital1})\end{array}\right\} = \mathbf{0} \tag{13}$$

One of the defining characteristics of 3SLS is that the errors are homoskedastic conditional on the instrumental variables. Using this assumption, we have

$$E\left[\left\{\begin{array}{c}\mathbf{z}_1\epsilon_1 \\ \mathbf{z}_2\epsilon_2\end{array}\right\}\left\{\begin{array}{cc}\mathbf{z}_1'\epsilon_1 & \mathbf{z}_2'\epsilon_2\end{array}\right\}\right] = \left\{\begin{array}{cc}\sigma_{11}E(\mathbf{z}_1\mathbf{z}_1') & \sigma_{12}E(\mathbf{z}_1\mathbf{z}_2') \\ \sigma_{21}E(\mathbf{z}_2\mathbf{z}_1') & \sigma_{22}E(\mathbf{z}_2\mathbf{z}_2')\end{array}\right\} \tag{14}$$

where $\sigma_{ij} = \mathrm{cov}(\epsilon_i, \epsilon_j)$. Let $\mathbf{\Sigma}$ denote the 2×2 matrix with typical element σ_{ij}.

The second defining characteristic of the 3SLS estimator is that it uses all the exogenous variables as instruments for all equations; here $\mathbf{z}_1 = \mathbf{z}_2 = (\texttt{wagegovt}, \texttt{govt}, \texttt{capital1}, 1)$, where the 1 indicates a constant term. From our discussion on the weight matrix and two-step estimation, we want to use the sample analogue of the matrix inverse of the right-hand side of (14) as our weight matrix.

To implement the 3SLS estimator, we apparently need to know $\mathbf{\Sigma}$ or at least have a consistent estimator of it. The solution is to fit (11) and (12) by 2SLS, use the sample residuals $\widehat{\epsilon_1}$ and $\widehat{\epsilon_2}$ to estimate $\mathbf{\Sigma}$, then estimate the parameters of (13) via GMM by using the weight matrix just discussed.

▷ Example 16: 3SLS estimation

3SLS is easier to do using gmm than it sounds. The 3SLS estimator is a two-step GMM estimator. In the first step, we do the equivalent of 2SLS on each equation, and then we compute a weight matrix based on (14). Finally, we perform a second step of GMM with this weight matrix.

In Stata, we type

```
. use https://www.stata-press.com/data/r17/klein, clear
. gmm (eq1: consump - {xb: wagepriv wagegovt _cons})
>     (eq2: wagepriv - {xc: consump govt capital1 _cons}),
>     instruments(eq1: wagegovt govt capital1)
>     instruments(eq2: wagegovt govt capital1)
>     winitial(unadjusted, independent) wmatrix(unadjusted) twostep
Step 1
Iteration 0:   GMM criterion Q(b) =  4195.4487
Iteration 1:   GMM criterion Q(b) =  .22175631
Iteration 2:   GMM criterion Q(b) =  .22175631  (backed up)
Step 2
Iteration 0:   GMM criterion Q(b) =  .09716589
Iteration 1:   GMM criterion Q(b) =  .07028208
Iteration 2:   GMM criterion Q(b) =  .07028208
GMM estimation
Number of parameters =   7
Number of moments    =   8
Initial weight matrix: Unadjusted                 Number of obs   =         22
GMM weight matrix:     Unadjusted
```

	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
xb						
wagepriv	.8012754	.1279329	6.26	0.000	.5505314	1.052019
wagegovt	1.029531	.3048424	3.38	0.001	.432051	1.627011
_cons	19.3559	3.583772	5.40	0.000	12.33184	26.37996
xc						
consump	.4026076	.2567312	1.57	0.117	-.1005764	.9057916
govt	1.177792	.5421253	2.17	0.030	.1152461	2.240338
capital1	-.0281145	.0572111	-0.49	0.623	-.1402462	.0840173
_cons	14.63026	10.26693	1.42	0.154	-5.492552	34.75306

```
Instruments for equation eq1: wagegovt govt capital1 _cons
Instruments for equation eq2: wagegovt govt capital1 _cons
```

The independent suboption of the winitial() option tells gmm to assume that the residuals are independent across moment conditions; this suboption sets $\sigma_{21} = \sigma_{12} = 0$ in (14). Assuming both homoskedasticity and cross-equation independence is equivalent to fitting the two equations of our model independently by 2SLS. The wmatrix() option controls how the weight matrix is computed on the basis of the first-step parameter estimates before the second step of estimation; here we request a weight matrix that assumes conditional homoskedasticity but that does not impose the cross-equation independence like the initial weight matrix we used. In this example, we also illustrated how to name residual equations and how equation names can be used in the instruments() option. Our results are identical to those in [R] **reg3**.

We could have specified our instruments with the syntax

```
instruments(wagegovt govt capital1)
```

because gmm uses the variables listed in the instruments() option for all equations unless you specify which equations the list of instruments is to be used with. However, we wanted to emphasize

that the same instruments are being used for both equations; in a moment, we will discuss an estimator that does not use the same instruments in all equations.

◁

In the previous example, if we omit the `twostep` option, the resulting coefficients will be equivalent to equation-by-equation 2SLS, which Wooldridge (2010, 216) calls the "system 2SLS estimator". Eliminating the `twostep` option makes the `wmatrix()` option irrelevant, so that option can be eliminated as well.

So far, we have developed the traditional 3SLS estimator. Wooldridge (2010, chap. 8) discusses the "GMM 3SLS" estimator, which extends the traditional 3SLS estimator by allowing for heteroskedasticity and different instruments for different equations.

Generalizing (14) to an arbitrary number of equations, we have

$$E\left(\mathbf{Z}'\boldsymbol{\epsilon}\boldsymbol{\epsilon}'\mathbf{Z}\right) = E\left(\mathbf{Z}'\boldsymbol{\Sigma}\mathbf{Z}\right) \tag{15}$$

where

$$\mathbf{Z} = \begin{bmatrix} \mathbf{z}_1 & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{z}_2 & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{z}_m \end{bmatrix}$$

and $\boldsymbol{\Sigma}$ is now $m \times m$. Equation (15) is the multivariate analogue of a homoskedasticity assumption; for each equation, the error variance is constant for all observations, as is the covariance between any two equations' errors.

We can relax this homoskedasticity assumption by considering different weight matrices. For example, if we continue to assume that observations are independent but not necessarily identically distributed, then by specifying `wmatrix(robust)`, we would obtain a weight matrix that allows for heteroskedasticity:

$$\widehat{W} = \frac{1}{N}\sum_i \mathbf{Z}_i'\widehat{\epsilon}_i\widehat{\epsilon}_i'\mathbf{Z}_i$$

This is the weight matrix in Wooldridge's (2010, 218) Procedure 8.1, "GMM with Optimal Weighting Matrix". By default, `gmm` would report standard errors based on his covariance matrix (8.27); specifying `vce(unadjusted)` would provide the optimal GMM standard errors. If you have multiple observations for each individual or firm in your dataset, you could specify `wmatrix(cluster` *id*`)`, where *id* identifies individuals or firms. This would allow arbitrary within-individual correlation, though it does not account for an individual-specific fixed or random effect. In both cases, we would continue to use `winitial(unadjusted, independent)` so that the first-step estimates are the system 2SLS estimates.

Wooldridge (2010, sec. 9.6) discusses instances where it is necessary to use different instruments in different equations. The GMM 3SLS estimator with different instruments in different equations but with conditional homoskedasticity is what Hayashi (2000, 275) calls the "full-information instrumental variables efficient" (FIVE) estimator. Implementing the FIVE estimator is easy with `gmm`. For example, say we have a two-equation system where `kids`, `age`, `income`, and `education` are all valid instruments for the first equation but where `education` is not a valid instrument for the second equation. Then, our syntax would take the form

```
gmm (rexp1) (rexp2), instruments(1:kids age income education)
  instruments(2:kids age income)
```

The following syntax is equivalent:

```
gmm (rexp1) (rexp2), instruments(kids age income)
  instruments(1:education)
```

Because we did not specify a list of equations in the second example's first `instruments()` option, those variables are used as instruments in both equations. You can use whichever syntax you prefer. The first requires a bit more typing but is arguably more transparent.

If all the regressors in the model are exogenous, then the traditional 3SLS estimator is the seemingly unrelated regression (SUR) estimator. Here you would specify all the regressors as instruments.

Dynamic panel-data models

Commands in Stata that work with panel data expect the data to be in the "long" format, meaning that each row of the dataset consists of one subobservation that is a member of a logical observation (represented by the panel identifier variable). See [D] **reshape** for a discussion of the long versus "wide" data forms. `gmm` is no exception in this respect when used with panel data. From a theoretical perspective, however, it is sometimes easier to view GMM estimators for panel data as system estimators in which we have N observations on a system of T equations, where N and T are the number of observations and panels, respectively, rather than a single-equation estimator with NT observations. Usually, each of the T equations will in fact be the same, though we will want to specify different instruments for each of these equations.

In a dynamic panel-data model, lagged values of the dependent variable are included as regressors. Here we consider a simple model with one lag of the dependent variable y as a regressor and a vector of strictly exogenous regressors, $\mathbf{x}_{it}$:

$$y_{it} = \rho y_{i,t-1} + \mathbf{x}_{it}'\boldsymbol{\beta} + u_i + \epsilon_{it} \tag{16}$$

u_i can be either a fixed- or a random-effect term in the sense that we do not require $\mathbf{x}_{it}$ to be independent of it. Even with the assumption that ϵ_{it} is i.i.d., the presence of both $y_{i,t-1}$ and u_i in (16) renders both the standard fixed- and random-effects estimators to be inconsistent because of the well-known Nickell (1981) bias. OLS regression of y_{it} on $y_{i,t-1}$ and $\mathbf{x}_{it}$ also produces inconsistent estimates because $y_{i,t-1}$ will be correlated with the error term.

❑ Technical note

Stata has the `xtabond`, `xtdpd`, and `xtdpdsys` commands (see [XT] **xtabond**, [XT] **xtdpd**, and [XT] **xtdpdsys**) to fit equations like (16); for everyday use, those commands are preferred because they offer features such as Windmeijer (2005) bias-corrected standard errors to account for the bias of traditional two-step GMM standard errors seen in dynamic panel-data models and, being linear estimators, only require you to specify variable names instead of complete equations. However, using `gmm` has several pedagogical advantages, including the ability to tie those model-specific commands into a more general framework, a clear illustration of how certain types of instrument matrices for panel-data models are formed, and demonstrations of several advanced features of `gmm`.

❑

First-differencing (16) removes the panel-specific u_i term:

$$y_{it} - y_{i,t-1} = \rho(y_{i,t-1} - y_{i,t-2}) + (\mathbf{x}_{it} - \mathbf{x}_{i,t-1})'\boldsymbol{\beta} + (\epsilon_{it} - \epsilon_{i,t-1}) \tag{17}$$

However, now $(y_{i,t-1} - y_{i,t-2})$ is correlated with $(\epsilon_{it} - \epsilon_{i,t-1})$. Thus, we need an instrument that is correlated with the former but not the latter. The lagged variables in (17) mean that the equation is not estimable for $t < 3$, so consider when $t = 3$. We have

$$y_{i3} - y_{i2} = \rho(y_{i2} - y_{i1}) + (\mathbf{x}_{i3} - \mathbf{x}_{i2})'\boldsymbol{\beta} + (\epsilon_{i3} - \epsilon_{i2}) \tag{18}$$

In the Arellano–Bond (1991) estimator, lagged levels of the dependent variable are used as instruments. With our assumption that the ϵ_{it} are i.i.d., (16) intimates that y_{i1} can serve as an instrumental variable when we fit (18).

Next, consider (17) when $t = 4$. We have

$$y_{i4} - y_{i3} = \rho(y_{i3} - y_{i2}) + (\mathbf{x}_{i4} - \mathbf{x}_{i3})'\boldsymbol{\beta} + (\epsilon_{i4} - \epsilon_{i3})$$

Now, (16) shows that both y_{i1} and y_{i2} are uncorrelated with the error term $(\epsilon_{i4} - \epsilon_{i3})$, so we have two instruments available. For $t = 5$, you can show that y_{i1}, y_{i2}, and y_{i3} can serve as instruments. As may now be apparent, one of the key features of these dynamic panel-data models is that the available instruments depend on the time period, t, as was the case for some of the panel Poisson models we considered earlier. Because the $\mathbf{x}_{it}$ are strictly exogenous by assumption, they can serve as their own instruments.

The initial weight matrix that is appropriate for the GMM dynamic panel-data estimator is slightly more involved than the unadjusted matrix that we have used in most of our previous examples and that assumes the errors are i.i.d. First, rewrite (17) for panel i as

$$\mathbf{y}_i - \mathbf{y}_i^L = \rho\,(\mathbf{y}_i^L - \mathbf{y}_i^{LL}) + (\mathbf{X}_i - \mathbf{X}_i^L)\boldsymbol{\beta} + (\boldsymbol{\epsilon}_i - \boldsymbol{\epsilon}_i^L)$$

where $\mathbf{y}_i = (y_{i3}, \ldots, y_{iT})$ and $\mathbf{y}_i^L = (y_{i2}, \ldots, y_{i,T-1})$, $\mathbf{y}_i^{LL} = (y_{i1}, \ldots, y_{i,T-2})$, and $\mathbf{X}_i$, $\mathbf{X}_i^L$, $\boldsymbol{\epsilon}_i$, and $\boldsymbol{\epsilon}_i^L$ are defined analogously. Let $\mathbf{Z}$ denote the full matrix of instruments for panel i, including the variables specified in both the `instruments()` and `xtinstruments()` options; the exact structure is detailed in *Methods and formulas*.

By assumption, ϵ_{it} is i.i.d., so the first difference $(\epsilon_{it} - \epsilon_{i,t-1})$ is necessarily autocorrelated with correlation -0.5. Therefore, we should not use a weight matrix that assumes the errors are independent. For dynamic panel-data models, we can show that the appropriate initial weight matrix is

$$\widehat{\mathbf{W}} = \left(\frac{1}{N}\sum\nolimits_i \mathbf{Z}_i'\mathbf{H}_D\mathbf{Z}_i\right)^{-1}$$

where

$$\mathbf{H}_D = \begin{bmatrix} 1 & -0.5 & 0 & \ldots & 0 & 0 \\ -0.5 & 1 & -0.5 & \ldots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \ldots & 1 & -0.5 \\ 0 & 0 & 0 & \ldots & -0.5 & 1 \end{bmatrix}$$

We can obtain this initial weight matrix by specifying `winitial(xt D)`. The letter `D` indicates that the equation we are estimating is specified in first differences.

▷ Example 17: Arellano–Bond estimator

Say we want to fit the model

$$\mathtt{n}_{it} = \rho\,\mathtt{n}_{i,t-1} + \beta_1\mathtt{w}_{it} + \beta_2\mathtt{w}_{i,t-1} + \beta_3\mathtt{k}_{it} + \beta_4\mathtt{k}_{i,t-1} + u_i + \epsilon_{it} \tag{19}$$

where we assume that $\mathtt{w}_{it}$ and $\mathtt{k}_{it}$ are strictly exogenous. First-differencing, our residual equation is

$$\begin{aligned}\epsilon^*_{it} = (\epsilon_{it} - \epsilon_{i,t-1}) =& \mathtt{n}_{it} - \mathtt{n}_{i,t-1} - \rho\,(\mathtt{n}_{i,t-1} - \mathtt{n}_{i,t-2}) - \beta_1(\mathtt{w}_{it} - \mathtt{w}_{i,t-1}) \\ &- \beta_2(\mathtt{w}_{i,t-1} - \mathtt{w}_{i,t-2}) - \beta_3(\mathtt{k}_{it} - \mathtt{k}_{i,t-1}) - \beta_4(\mathtt{k}_{i,t-1} - \mathtt{k}_{i,t-2})\end{aligned} \tag{20}$$

In Stata, we type

```
. use https://www.stata-press.com/data/r17/abdata
. gmm (D.n - {rho}*LD.n - {xb:D.w LD.w D.k LD.k}),
> xtinstruments(n, lags(2/.)) instruments(D.w LD.w D.k LD.k, noconstant)
> deriv(/rho = -1*LD.n) deriv(/xb = -1) winitial(xt D) onestep
Step 1
Iteration 0:   GMM criterion Q(b) =   .0011455
Iteration 1:   GMM criterion Q(b) =  .00009103
Iteration 2:   GMM criterion Q(b) =  .00009103
GMM estimation
Number of parameters =   5
Number of moments    =  32
Initial weight matrix: XT D                     Number of obs   =        751
                                  (Std. err. adjusted for 140 clusters in id)
------------------------------------------------------------------------------
             |               Robust
             | Coefficient  std. err.      z    P>|z|     [95% conf. interval]
-------------+----------------------------------------------------------------
rho          |
       _cons |   .8041712   .1199819     6.70   0.000     .5690111    1.039331
-------------+----------------------------------------------------------------
xb           |
           w |
         D1. |  -.5600476   .1619472    -3.46   0.001    -.8774583    -.242637
         LD. |   .3946699   .1092229     3.61   0.000     .1805969    .6087429
             |
           k |
         D1. |   .3520286   .0536546     6.56   0.000     .2468676    .4571897
         LD. |  -.2160435   .0679689    -3.18   0.001    -.3492601   -.0828269
------------------------------------------------------------------------------
Instruments for equation 1:
        XT-style: L(2/.).n
        Standard: D.w LD.w D.k LD.k
```

Because `w` and `k` are strictly exogenous, we specified their variants that appear in (20) in the `instruments()` option; because there is no constant term in the model, we specified `noconstant` to omit the constant from the instrument list.

We specified `xtinstruments(n, lags(2/.))` to tell `gmm` what instruments to use for the lagged dependent variable included as a regressor in (19). On the basis of our previous discussion, lags two and higher of $\mathtt{n}_{it}$ can serve as instruments. The `lags(2/.)` suboption tells `gmm` that the first available instrument for $\mathtt{n}_{it}$ is the lag-two value $\mathtt{n}_{i,t-2}$. The "`.`" tells `gmm` to use all further lags of $\mathtt{n}_{it}$ as instruments as well. The instrument matrices in dynamic panel-data models can become large if the dataset has many time periods per panel. In those cases, you could specify, for example, `lags(2/4)` to use just lags two through four instead of using all available lags.

Our results are identical to those we would obtain using `xtabond` with the syntax

```
xtabond n L(0/1).w L(0/1).k, lags(1) noconstant vce(robust)
```

If we had left off the vce(robust) option in our call to xtabond, we would have had to specify vce(unadjusted) in our call to gmm to obtain the same standard errors.

◁

❑ Technical note

gmm automatically excludes observations for which there are no valid observations for the panel-style instruments. However, it keeps in the estimation sample those observations for which fewer than the maximum number of instruments you requested are available. For example, if you specify the lags(2/4) suboption, you have requested three instruments, but gmm will keep observations even if only one or two instruments are available.

❑

▷ Example 18: Two-step Arellano–Bond estimator

Here we refit the model from example 17, using the two-step GMM estimator.

```
. gmm (D.n - {rho}*LD.n - {xb:D.w LD.w D.k LD.k}),
> xtinstruments(n, lags(2/.)) instruments(D.w LD.w D.k LD.k, noconstant)
> deriv(/rho = -1*LD.n) deriv(/xb = -1) winitial(xt D) wmatrix(robust)
> vce(unadjusted)
Step 1
Iteration 0:   GMM criterion Q(b) =   .0011455
Iteration 1:   GMM criterion Q(b) =  .00009103
Iteration 2:   GMM criterion Q(b) =  .00009103
Step 2
Iteration 0:   GMM criterion Q(b) =  .44107941
Iteration 1:   GMM criterion Q(b) =   .4236729
Iteration 2:   GMM criterion Q(b) =   .4236729  (backed up)
GMM estimation
Number of parameters =   5
Number of moments    =  32
Initial weight matrix: XT D                      Number of obs   =        751
GMM weight matrix:     Robust
```

	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
rho						
_cons	.8044783	.0534763	15.04	0.000	.6996667	.90929
xb						
w						
D1.	-.5154978	.0335506	-15.36	0.000	-.5812557	-.4497399
LD.	.4059309	.0637294	6.37	0.000	.2810235	.5308384
k						
D1.	.3556204	.0390892	9.10	0.000	.2790071	.4322337
LD.	-.2204521	.046439	-4.75	0.000	-.3114709	-.1294332

```
Instruments for equation 1:
        XT-style: L(2/.).n
        Standard: D.w LD.w D.k LD.k
```

Our results match those you would obtain with the command

```
xtabond n L(0/1).(w k), lags(1) noconstant twostep
```

◁

❑ Technical note

If we had specified `vce(robust)` in our call to `gmm`, we would have obtained the traditional sandwich-based robust covariance matrix, but our standard errors would not match those we would obtain by specifying `vce(robust)` with the `xtabond` command. The `xtabond`, `xtdpd`, and `xtdpdsys` commands implement a bias-corrected robust VCE for the two-step GMM dynamic panel-data estimator. Traditional VCEs computed after the two-step dynamic panel-data estimator have been shown to exhibit often-severe bias; see Windmeijer (2005).

❑

Neither of the two dynamic panel-data examples (17 and 18) we have fit so far include a constant term. When a constant term is included, the dynamic panel-data estimator is in fact a two-equation system estimator. For notational simplicity, consider a simple model containing just a constant term and one lag of the dependent variable:

$$y_{it} = \alpha + \rho y_{i,t-1} + u_i + \epsilon_{it}$$

First-differencing to remove the u_i term, we have

$$y_{it} - y_{i,t-1} = \rho(y_{i,t-1} - y_{i,t-2}) + (\epsilon_{it} - \epsilon_{i,t-1}) \tag{21}$$

This has also eliminated the constant term. If we assume $E(u_i) = 0$, which is reasonable if a constant term is included in the model, then we can recover α by including the moment condition

$$y_{it} = \alpha + \rho y_{i,t-1} + \epsilon'_{it} \tag{22}$$

where $\epsilon'_{it} = u_i + \epsilon_{it}$. The parameter ρ continues to be identified by (21), so the only instrument we use with (22) is a constant term. As before, the error term $(\epsilon_{i,t} - \epsilon_{i,t-1})$ is necessarily autocorrelated with correlation coefficient -0.5, though the error term ϵ'_{it} is white noise. Therefore, our initial weight matrix should be

$$\widehat{\mathbf{W}} = \left(\frac{1}{N}\sum\nolimits_i \mathbf{Z}_i'\mathbf{H}\mathbf{Z}_i\right)^{-1}$$

where

$$\mathbf{H} = \begin{bmatrix} \mathbf{H}_D & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix}$$

and $\mathbf{I}$ is a conformable identity matrix.

One complication arises concerning the relevant estimation sample. Looking at (21), we apparently lose the first two observations from each panel because of the presence of $y_{i,t-2}$, but in (22), we need to sacrifice only one observation for $y_{i,t-1}$. For most multiple-equation models, we need to use the same estimation sample for all equations. However, in dynamic panel-data models, we can use more observations to fit the equation in level form [(22) here] than the equation in first differences [equation (21)]. To request this behavior, we specify the `nocommonesample` option to `gmm`. That option tells `gmm` to use as many observations as possible for each equation, ignoring the loss of observations due to lagging or differencing.

▷ Example 19: Arellano–Bond estimator with constant term

Here we fit the model

$$\mathtt{n}_{it} = \alpha + \rho\,\mathtt{n}_{i,t-1} + u_i + \epsilon_{it}$$

Without specifying derivatives, our command would be

```
. gmm (D.n - {rho}*LD.n) (n - {alpha} - {rho}*L.n),
> xtinstruments(1: n, lags(2/.)) instruments(1:, noconstant) onestep
> winitial(xt DL) vce(unadj) nocommonesample
```

We would specify `winitial(xt DL)` to obtain the required initial weight matrix. The notation `DL` indicates that our first residual equation is in first differences and that the second residual equation is in levels (not first-differenced). We exclude a constant in the instrument list for the first equation because first-differencing removed the constant term. Because we do not specify the `instruments()` option for the second residual equation, a constant is used by default.

This example also provides us the opportunity to illustrate how to specify derivatives for multiple-equation GMM models. Within the `derivative()` option, instead of specifying just the parameter name, now you must specify the equation name or number, a slash, and the parameter name to which the derivative applies. In Stata, we type

```
. gmm (D.n - {rho}*LD.n) (n - {alpha} - {rho}*L.n),
> xtinstruments(1: n, lags(2/.)) instruments(1:, noconstant)
> derivative(1/rho = -1*LD.n) derivative(2/alpha = -1)
> derivative(2/rho = -1*L.n) winitial(xt DL) vce(unadj) nocommonesample onestep
Step 1
Iteration 0:   GMM criterion Q(b) =  .09894466
Iteration 1:   GMM criterion Q(b) =  .00023508
Iteration 2:   GMM criterion Q(b) =  .00023508

GMM estimation

Number of parameters =   2
Number of moments    =  29
Initial weight matrix: XT DL                      Number of obs   =   *
```

	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
/rho	1.023349	.0608293	16.82	0.000	.9041259	1.142572
/alpha	-.0690864	.0660343	-1.05	0.295	-.1985112	.0603384

```
* Number of observations for equation 1: 751
  Number of observations for equation 2: 891

Instruments for equation 1:
        XT-style: L(2/.).n
Instruments for equation 2:
        Standard: _cons
```

These results are identical to those we would obtain by typing

```
xtabond n, lags(1)
```

Because we specified `nocommonesample`, `gmm` did not report the number of observations used in the header of the output. In this dataset, there are in fact 1,031 observations on 140 panels. In the second equation, the presence of the lagged value of `n` reduces the sample size for that equation to $1031 - 140 = 891$. In the first equation, we lose the first two observations per panel because of lagging and differencing, which leads to 751 usable observations. These tallies are listed after the coefficient table in the output.

◁

❑ Technical note

Specifying

```
xtinstruments(x1 x2 x3, lags(1/3))
```

differs from

```
instruments(L(1/3).(x1 x2 x3))
```

in how observations are excluded from the estimation sample. When you use the latter syntax, `gmm` must exclude the first three observations from each panel when computing the residual equation: you requested that three lags of each regressor be used as instruments, so the first residual that could be interacted with those instruments is the one for $t = 4$. On the other hand, when you use `xtinstruments()`, you are telling `gmm` that you would like to use up to the first three lags of `x1`, `x2`, and `x3` as instruments but that using just one lag is acceptable. Because most panel datasets have a relatively modest number of observations per panel, dynamic instrument lists are typically used so that the number of usable observations is maximized. Dynamic instrument lists also accommodate the fact that there are more valid instruments for later time periods than earlier time periods.

Specifying panel-style instruments using the `xtinstruments()` option also affects how the standard instruments specified in the `instruments()` option are treated. To illustrate, we will suppose that we have a balanced panel dataset with $T = 5$ observations per panel and that we specify

```
. gmm ..., xtinstruments(w, lags(1/2)) instruments(x)
```

We will lose the first observation because we need at least one lag of `w` to serve as an instrument. Our instrument matrix for panel i will therefore be

$$\mathbf{Z}_i = \begin{bmatrix} w_{i1} & 0 & 0 & 0 \\ 0 & w_{i1} & 0 & 0 \\ 0 & w_{i2} & 0 & 0 \\ 0 & 0 & w_{i2} & 0 \\ 0 & 0 & w_{i3} & 0 \\ 0 & 0 & 0 & w_{i3} \\ 0 & 0 & 0 & w_{i4} \\ x_{i2} & x_{i3} & x_{i4} & x_{i5} \\ 1 & 1 & 1 & 1 \end{bmatrix} \tag{23}$$

The vector of ones in the final row represents the constant term implied by the `instruments()` option. Because we lost the first observation, the residual vector $\mathbf{u}_i$ will be 4×1. Thus, our moment conditions for the ith panel can be written in matrix notation as

$$E\{\mathbf{Z}_i\mathbf{u}_i(\boldsymbol{\beta})\} = E\left\{\mathbf{Z}_i \begin{bmatrix} u_{i2}(\boldsymbol{\beta}) \\ u_{i3}(\boldsymbol{\beta}) \\ u_{i4}(\boldsymbol{\beta}) \\ u_{i5}(\boldsymbol{\beta}) \end{bmatrix}\right\} = \mathbf{0}$$

The moment conditions corresponding to the final two rows of (23) say that

$$E\left\{\sum_{t=2}^{T=4} x_{it}u_{it}(\boldsymbol{\beta})\right\} = 0 \qquad \text{and} \qquad E\left\{\sum_{t=2}^{T=4} u_{it}(\boldsymbol{\beta})\right\} = 0$$

Because we specified panel-style instruments with the `xtinstruments()` option, `gmm` no longer uses moment conditions for strictly exogenous variables of the form $E\{x_{it}u_{it}(\beta)\} = 0$ for each t. Instead, the moment conditions now stipulate that the average (over t) of $x_{it}u_{it}(\boldsymbol{\beta})$ has expectation zero. This corresponds to the approach proposed by Arellano and Bond (1991, 280) and others.

When you request panel-style instruments with the `xtinstruments()` option, the number of instruments in the $\mathbf{Z}_i$ matrix increases quadratically in the number of periods. The dynamic panel-data estimators we have discussed in this section are designed for datasets that contain a large number of panels and a modest number of time periods. When the number of time periods is large, estimators that use standard (non-panel-style) instruments are more appropriate.

❑

We have focused on the Arellano–Bond dynamic panel-data estimator because of its relative simplicity. `gmm` can additionally fit any models that can be formulated using the `xtdpd` and `xtdpdsys` commands; see [XT] **xtdpd** and [XT] **xtdpdsys**. The key is to determine the appropriate instruments to use for the level and difference equations. You may find it useful to fit a version of your model with those commands to determine what instruments and XT-style instruments to use. We conclude this section with an example using the Arellano–Bover/Blundell–Bond estimator.

▷ Example 20: Arellano–Bover/Blundell–Bond estimator

We fit a small model that includes one lag of the dependent variable `n` as a regressor as well as the contemporaneous and first lag of `w`, which we assume are strictly exogenous. We could fit our model with `xtdpdsys` by using the syntax

```
xtdpdsys n L(0/1).w, lags(1) twostep
```

When we apply virtually all the syntax issues we have discussed so far, the equivalent gmm command is

```
. gmm (n - {rho}*L.n - {w}*w - {lagw}*L.w - {c})
>     (D.n - {rho}*LD.n - {w}*D.w - {lagw}*LD.w),
>     xtinst(1: D.n, lags(1/1)) xtinst(2: n, lags(2/.))
>     inst(2: D.w LD.w, noconstant)
>     deriv(1/rho = -1*L.n) deriv(1/w = -1*w)
>     deriv(1/lagw = -1*L.w) deriv(1/c = -1)
>     deriv(2/rho = -1*LD.n) deriv(2/w = -1*D.w)
>     deriv(2/lagw = -1*LD.w)
>     winit(xt LD) wmatrix(robust) vce(unadjusted) nocommonesample
Step 1
Iteration 0:   GMM criterion Q(b) =  .10170339
Iteration 1:   GMM criterion Q(b) =  .00022772
Iteration 2:   GMM criterion Q(b) =  .00022772
Step 2
Iteration 0:   GMM criterion Q(b) =  .59965014
Iteration 1:   GMM criterion Q(b) =  .56578186
Iteration 2:   GMM criterion Q(b) =  .56578186
GMM estimation
Number of parameters =   4
Number of moments    =  39
Initial weight matrix: XT LD                      Number of obs   =   *
GMM weight matrix:     Robust
```

	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
/rho	1.122738	.0206512	54.37	0.000	1.082263	1.163214
/w	-.6719909	.0246148	-27.30	0.000	-.7202351	-.6237468
/lagw	.571274	.0403243	14.17	0.000	.4922398	.6503083
/c	.154309	.17241	0.90	0.371	-.1836084	.4922263

```
* Number of observations for equation 1: 891
  Number of observations for equation 2: 751
Instruments for equation 1:
        XT-style: LD.n
        Standard: _cons
Instruments for equation 2:
        XT-style: L(2/.).n
        Standard: D.w LD.w
```

◁

Details of moment-evaluator programs

In examples 10, 11, 12, 13, and 14, we used moment-evaluator programs to evaluate moment conditions that could not be specified using the interactive version of gmm. In example 13, we also showed how to pass additional information to an evaluator program. Here we discuss how to make moment-evaluator programs provide derivatives and accept weights.

The complete specification for a moment-evaluator program's syntax statement is

```
syntax varlist if [weight], at(name) options [derivatives(varlist)]
```

The macro ‘varlist’ contains the list of variables that we are to fill in with the values of our residual equations. The macro ‘if’ represents an if condition that restricts the estimation sample. The macro ‘at’ represents a vector containing the parameter values at which we are to evaluate our

residual equations. `options` represent other options that you specify in your call to `gmm` and want to have passed to your moment-evaluator programs. In example 13, we included the `mylhs()` option so that we could pass the name of the dependent variable to our evaluator program.

Two new elements of the `syntax` statement allow for weights and derivatives. `weight` specifies the types of weights your program allows. The interactive version of `gmm` allows for `fweights`, `aweights`, and `pweights`. However, unless you explicitly allow your moment evaluator program to accept weights, you cannot specify weights in your call to `gmm` with the moment-evaluator program version.

The `derivatives()` option is used to pass to your program a set of variables that you are to fill in with the derivatives of your residual equations with respect to the parameters.

To indicate that your program can calculate derivatives, you specify either the `hasderivatives` or the `haslfderivatives` option to `gmm`. The `hasderivatives` option indicates that your program calculates parameter-level derivatives; that method requires more work but can be applied to any GMM problem. The `haslfderivatives` option requires less work but can be used only when the model's residual equations satisfy certain restrictions and when you use the {*lcname*`:`*varlist*} syntax with the `parameters()` option.

We first consider how to write the derivative computation logic to work with the `hasderivatives` option and provide an example; then, we do the same for the `haslfderivatives` option.

Say that you specify k parameters in the `nparameters()` or `parameters()` option and q equations in the `nequations()` or `equations()` option and that you specify `hasderivatives`. Then, `'derivatives'` will contain $k \times q$ variables. The first k variables are for the derivatives of the first residual equation with respect to the k parameters, the second k variables are for the derivatives of the second residual equation, and so on.

▷ Example 21: Specifying derivatives with simple parameter names

To focus on how to specify derivatives, we return to the simple moment-evaluator program we used in example 11, in which we had three regressors, and extend it to supply derivatives. The residual equation corresponding to moment condition (9) is

$$u_{it}(\boldsymbol{\beta}) = y_{it} - \mu_{it}\frac{\overline{y}_i}{\overline{\mu}_i}$$

where μ_{it}, $\overline{\mu}_i$, and $\overline{y}_i$ were defined previously. Now,

$$\frac{\partial}{\partial \beta_j} u_{it}(\boldsymbol{\beta}) = -\mu_{it}\frac{\overline{y}_i}{\overline{\mu}_i^2}\left(x_{it}^{(j)}\overline{\mu}_i - \frac{1}{T}\sum_{l=1}^{l=T} x_{il}^{(j)}\mu_{il}\right) \tag{24}$$

where $x_{it}^{(j)}$ represents the jth element of $\mathbf{x}_{it}$.

Our moment-evaluator program is

```
program gmm_poideriv
        version 17.0
        syntax varlist if, at(name) [derivatives(varlist)]
        quietly {
                // Calculate residuals as before
                tempvar mu mubar ybar
                generate double `mu' = exp(x1*`at'[1,1] + x2*`at'[1,2]     ///
                                           + x3*`at'[1,3]) `if'
                egen double `mubar' = mean(`mu') `if', by(id)
                egen double `ybar' = mean(y) `if', by(id)
                replace `varlist' = y - `mu'*`ybar'/`mubar' `if'
                // Did -gmm- request derivatives?
                if "`derivatives'" == "" {
                        exit                    // no, so we are done
                }
                // Calculate derivatives
                // We need the panel means of x1*mu, x2*mu, and x3*mu
                tempvar work x1mubar x2mubar x3mubar
                generate double `work' = x1*`mu' `if'
                egen double `x1mubar' = mean(`work') `if', by(id)
                replace `work' = x2*`mu' `if'
                egen double `x2mubar' = mean(`work') `if', by(id)
                replace `work' = x3*`mu' `if'
                egen double `x3mubar' = mean(`work') `if', by(id)
                local d1: word 1 of `derivatives'
                local d2: word 2 of `derivatives'
                local d3: word 3 of `derivatives'
                replace `d1' = -1*`mu'*`ybar'/`mubar'^2*(x1*`mubar' - `x1mubar')
                replace `d2' = -1*`mu'*`ybar'/`mubar'^2*(x2*`mubar' - `x2mubar')
                replace `d3' = -1*`mu'*`ybar'/`mubar'^2*(x3*`mubar' - `x3mubar')
        }
end
```

The `derivatives()` option is made optional in the `syntax` statement by placing it in square brackets. If `gmm` needs to evaluate your residual equations but does not need derivatives at that time, then the `derivatives()` option will be empty. In our program, we check to see whether that is the case and, if so, `exit` without calculating derivatives. As is often the case with [R] **ml** as well, the portion of our program devoted to derivatives is longer than the code to compute the objective function.

The first part of our derivative code computes the term

$$\frac{1}{T}\sum_{l=1}^{l=T} x_{il}^{(j)}\mu_{il} \tag{25}$$

for $x_{it}^{(j)} =$ `x1`, `x2`, and, `x3`. The `` `derivatives' `` macro contains three variable names corresponding to the three parameters of the `` `at' `` matrix. We extract those names into local macros `` `d1' ``, `` `d2' ``, and `` `d3' `` and then fill in the variables those macros represent with the derivatives shown in (24).

With our program written, we fit our model by typing

```
. use https://www.stata-press.com/data/r17/poisson1, clear
. gmm gmm_poideriv, nequations(1) parameters(b1 b2 b3)
> instruments(x1 x2 x3, noconstant) vce(cluster id) onestep hasderivatives
Step 1
Iteration 0:   GMM criterion Q(b) =   51.99142
Iteration 1:   GMM criterion Q(b) =  .04345191
Iteration 2:   GMM criterion Q(b) =  8.720e-06
Iteration 3:   GMM criterion Q(b) =  7.115e-13
Iteration 4:   GMM criterion Q(b) =  5.130e-27
note: model is exactly identified.
GMM estimation
Number of parameters =   3
Number of moments    =   3
Initial weight matrix: Unadjusted                 Number of obs   =        409
                                  (Std. err. adjusted for 45 clusters in id)
------------------------------------------------------------------------------
             |               Robust
             | Coefficient  std. err.      z    P>|z|     [95% conf. interval]
-------------+----------------------------------------------------------------
         /b1 |     1.94866   .1000265    19.48   0.000     1.752612    2.144709
         /b2 |    -2.966119  .0923592   -32.12   0.000    -3.14714    -2.785099
         /b3 |     1.008634  .1156561     8.72   0.000      .781952    1.235315
------------------------------------------------------------------------------
Instruments for equation 1: x1 x2 x3
```

Our results are identical to those in example 11. Another way to verify that our program calculates derivatives correctly would be to type

```
. gmm gmm_poideriv, nequations(1) parameters(b1 b2 b3)
> instruments(x1 x2 x3, noconstant) vce(cluster id) onestep
```

Without the `hasderivatives` or `haslfderivatives` option, `gmm` will not request derivatives from your program, even if it contains code to compute them. If you have trouble obtaining convergence with the `hasderivatives` or `haslfderivatives` option but do not have trouble without specifying one of them, then you need to recheck your derivatives.

◁

After example 11, we remarked that the evaluator program would have to be changed to accommodate different regressors. We then showed how you can specify parameters using the syntax {*lcname*:*varlist*} and then use `matrix score` to compute linear combinations of variables. To specify derivatives when you specify parameters using this notation, ensure that your residual equations satisfy the "linear-form restriction" analogous to the restrictions of linear-form evaluators used by `ml`. See [R] **ml** and Gould, Pitblado, and Poi (2010) for more information about linear-form evaluators.

A GMM residual equation satisfies the linear-form restriction if the equation can be written in terms of a single observation in the dataset and if the equation for observation i does not depend on any observations $j \neq i$. Cross-sectional models satisfy the linear-form restriction. Time-series models satisfy the linear-form restriction only when no lags or leads are used.

Panel-data models often do not satisfy the linear-form restriction. For example, recall moment condition (9) for a panel Poisson model. That residual equation included panel-level mean terms $\bar{y}_i$ and $\bar{\mu}_i$, so the residual equation for an individual observation depends on all the observations in the same panel.

When a residual equation does not satisfy the linear-form restriction, neither will its derivatives. To apply the chain rule, we need a way to multiply the *lcname*-level derivative by each of the variables

in the equation to obtain parameter-level derivatives. In (24), for example, there is no way to factor out each $x_{it}^{(j)}$ variable and obtain an *lcname*-level derivative that we then multiply by each of the $x_{it}^{(j)}$s.

Suppose we do have a model with $q = 2$ moment conditions, both of which do satisfy the linear-form restriction, and we specify the `parameters()` option like this:

```
. gmm ..., parameters({eq1:x1 x2 _cons} {eq2:_cons} {eq3:x1 x2 _cons})
```

We have specified $n = 3$ *lcname*s in the `parameters()` option: `eq1`, `eq2`, and `eq3`. When we specify the `haslfderivatives` option, `gmm` will pass $n \times q = 3 \times 2 = 6$ variables in the `derivatives()` option. The first three variables are to be filled with

$$\frac{\partial}{\partial \texttt{eq1}} u_{1i}(\boldsymbol{\beta}) \qquad \frac{\partial}{\partial \texttt{eq2}} u_{1i}(\boldsymbol{\beta}) \quad \text{and} \quad \frac{\partial}{\partial \texttt{eq3}} u_{1i}(\boldsymbol{\beta})$$

where $u_{1i}(\boldsymbol{\beta})$ is the ith observation for the first moment equation. Then, the second three variables are to be filled with

$$\frac{\partial}{\partial \texttt{eq1}} u_{2i}(\boldsymbol{\beta}) \qquad \frac{\partial}{\partial \texttt{eq2}} u_{2i}(\boldsymbol{\beta}) \quad \text{and} \quad \frac{\partial}{\partial \texttt{eq3}} u_{2i}(\boldsymbol{\beta})$$

where $u_{2i}(\boldsymbol{\beta})$ is the second moment equation. In this example, we filled in a total of six variables with derivatives. If we instead used the `hasderivatives` option, we would have filled $k \times q = 7 \times 2 = 14$ variables; moreover, if we wanted to change the number of variables in our model, we would have modified our evaluator program.

▷ Example 22: Specifying derivatives with linear-form residual equations

In examples 9 and 10, we showed how to specify derivatives with an exponential regression model when using the interactive version of `gmm`. Here we show how to write a moment-evaluator program for the exponential regression model, including derivatives.

The residual equation for observation i is

$$u_i = y_i - \exp(\mathbf{x}_i'\boldsymbol{\beta})$$

where $\mathbf{x}_i$ may include a constant term. The derivative with respect to the linear combination $\mathbf{x}_i'\boldsymbol{\beta}$ is

$$\frac{\partial u_i}{\partial \mathbf{x}_i'\boldsymbol{\beta}} = -\exp(\mathbf{x}_i'\boldsymbol{\beta})$$

To verify that this residual equation satisfies the linear-form restriction, we see that for the jth element of $\boldsymbol{\beta}$, we have

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

so that given $\partial u_i/\partial \mathbf{x}_i'\boldsymbol{\beta}$, `gmm` can apply the chain rule to obtain the derivatives with respect to the individual parameters.

Our moment-evaluator program is

```
program gmm_poideriv2
        version 17.0
        syntax varlist if, at(name) [derivatives(varlist)]
        quietly {
                tempvar mu
                matrix score double `mu' = `at' `if', eq(#1)
                replace `mu' = exp(`mu')
                local depvar : coleq `at'
                local depvar : word 1 of `depvar'
                replace `varlist' = `depvar' - `mu' `if'
                // Did -gmm- request derivatives?
                if "`derivatives'" == "" {
                        exit                    // no, so we are done
                }
                // Calculate derivatives
                // The derivatives macro only has one variable
                // for this model.
                replace `derivatives' = -1*`mu' `if'
        }
end
```

To fit our model of doctor visits treating `income` as an endogenous regressor, we type

```
. use https://www.stata-press.com/data/r17/docvisits
. gmm gmm_poideriv2, nequations(1)
>     instruments(private chronic female age black hispanic)
>     parameters({docvis:private chronic female income _cons}) haslfderivatives
Step 1
Iteration 0:   GMM criterion Q(b) =  16.910173
Iteration 1:   GMM criterion Q(b) =  .82270871
Iteration 2:   GMM criterion Q(b) =  .21831995
Iteration 3:   GMM criterion Q(b) =  .12685934
Iteration 4:   GMM criterion Q(b) =  .12672369
Iteration 5:   GMM criterion Q(b) =  .12672365
Step 2
Iteration 0:   GMM criterion Q(b) =  .00234641
Iteration 1:   GMM criterion Q(b) =  .00215957
Iteration 2:   GMM criterion Q(b) =  .00215911
Iteration 3:   GMM criterion Q(b) =  .00215911
GMM estimation
Number of parameters =   5
Number of moments    =   7
Initial weight matrix: Unadjusted                 Number of obs   =      4,412
GMM weight matrix:     Robust
```

	Coefficient	Robust std. err.	z	P>|z|	[95% conf.	interval]
private	.535335	.159904	3.35	0.001	.221929	.848741
chronic	1.090126	.0617659	17.65	0.000	.9690668	1.211185
female	.6636579	.0959885	6.91	0.000	.475524	.8517918
income	.0142855	.0027162	5.26	0.000	.0089618	.0196092
_cons	-.5983477	.138433	-4.32	0.000	-.8696714	-.327024

```
Instruments for equation 1: private chronic female age black hispanic _cons
```

Our results match those shown in example 10.

We can change the variables in our model just by changing the `parameters()` and `instruments()` options; we do not need to make any changes to the moment-evaluator program, because we used linear-form derivatives. ◁

Depending on your model, allowing your moment-evaluator program to accept weights may be as easy as modifying the `syntax` command to allow them, or it may require significantly more work. If your program uses only commands such as `generate` and `replace`, then just modifying the `syntax` command is all you need to do; `gmm` takes care of applying the weights to the observation-level residuals when computing the sample moments, derivatives, and weight matrices. On the other hand, if your moment-evaluator program computes residuals using statistics that depend on multiple observations, then you must apply the weights passed to your program when computing those statistics.

In our examples of panel Poisson with strictly exogenous regressors (11 and 21), we used the statistics $\overline{\mu}_i$ and $\overline{y}_i$ when computing the residuals. If we are to allow weights with our moment-evaluator program, then we must incorporate those weights when computing $\overline{\mu}_i$ and $\overline{y}_i$. Moreover, looking at the derivative in (24), we see that the term highlighted in (25) is in fact a sample mean, so we must incorporate weights when computing it.

▷ Example 23: Panel Poisson with derivatives and weights

Here we modify the program in example 21 to accept frequency weights. One complication arises: we had been using `egen` to compute $\overline{\mu}_i$ and $\overline{y}_i$. `egen` does not accept weights, so we must compute $\overline{\mu}_i$ and $\overline{y}_i$ ourselves, incorporating any weights the user may specify. Our program is

```
program gmm_poiderivfw
        version 17.0
        syntax varlist if [fweight/], at(name) [derivatives(varlist)]
        quietly {
                if "`exp'" == "" {        // no weights
                        local exp 1       // weight each observation equally
                }
                // Calculate residuals as before
                tempvar mu mubar ybar sumwt
                generate double `mu' = exp(x1*`at'[1,1] + x2*`at'[1,2]    ///
                                                + x3*`at'[1,3]) `if'
                bysort id: generate double `sumwt' = sum(`exp')
                by id: generate double `mubar' = sum(`mu'*`exp')
                by id: generate double `ybar' = sum(y*`exp')
                by id: replace `mubar' = `mubar'[_N] / `sumwt'[_N]
                by id: replace `ybar' = `ybar'[_N] / `sumwt'[_N]
                replace `varlist' = y - `mu'*`ybar'/`mubar' `if'
                // Did -gmm- request derivatives?
                if "`derivatives'" == "" {
                        exit        // no, so we are done
                }
                // Calculate derivatives
                // We need the panel means of x1*mu, x2*mu, and x3*mu
                tempvar work x1mubar x2mubar x3mubar
                generate double `work' = x1*`mu' `if'
                by id: generate double `x1mubar' = sum(`work'*`exp')
                by id: replace `x1mubar' = `x1mubar'[_N] / `sumwt'[_N]
                replace `work' = x2*`mu' `if'
                by id: generate double `x2mubar' = sum(`work'*`exp')
                by id: replace `x2mubar' = `x2mubar'[_N] / `sumwt'[_N]
                replace `work' = x3*`mu' `if'
                by id: generate double `x3mubar' = sum(`work'*`exp')
                by id: replace `x3mubar' = `x3mubar'[_N] / `sumwt'[_N]
```

```
                local d1: word 1 of `derivatives'
                local d2: word 2 of `derivatives'
                local d3: word 3 of `derivatives'
                replace `d1' = -1*`mu'*`ybar'/`mubar'^2*(x1*`mubar' - `x1mubar')
                replace `d2' = -1*`mu'*`ybar'/`mubar'^2*(x2*`mubar' - `x2mubar')
                replace `d3' = -1*`mu'*`ybar'/`mubar'^2*(x3*`mubar' - `x3mubar')
        }
end
```

Our `syntax` command now indicates that `fweights` are allowed. The first part of our code looks at the macro `` `exp' ``. If it is empty, then the user did not specify weights in their call to `gmm`; and we set the macro equal to 1 so that we weight each observation equally. After we compute μ_{it}, we calculate $\overline{\mu}_i$ and $\overline{y}_i$, accounting for weights. To compute frequency-weighted means for each panel, we just multiply each observation by its respective weight, sum over all observations in the panel, then divide by the sum of the weights for the panel. (See [U] **20.24 Weighted estimation** for information on how to handle `aweights` and `pweights`.) We use the same procedure to compute the frequency-weighted variant of expression (25) in the derivative calculations. To use our program, we type

```
. use https://www.stata-press.com/data/r17/poissonwts
. gmm gmm_poiderivfw [fw=fwt], nequations(1) parameters(b1 b2 b3)
> instructions(x1 x2 x3, noconstant) vce(cluster id) onestep hasderivatives
Step 1
Iteration 0:   GMM criterion Q(b) =    49.8292
Iteration 1:   GMM criterion Q(b) =  .11136736
Iteration 2:   GMM criterion Q(b) =  .00008519
Iteration 3:   GMM criterion Q(b) =  7.110e-11
Iteration 4:   GMM criterion Q(b) =  5.596e-23
note: model is exactly identified.
GMM estimation
Number of parameters =   3
Number of moments    =   3
Initial weight matrix: Unadjusted                 Number of obs   =        819
                                  (Std. err. adjusted for 45 clusters in id)
------------------------------------------------------------------------------
             |               Robust
             | Coefficient  std. err.      z    P>|z|     [95% conf. interval]
-------------+----------------------------------------------------------------
         /b1 |    1.967766   .111795    17.60   0.000     1.748652    2.186881
         /b2 |   -3.060838   .0935561  -32.72   0.000    -3.244205   -2.877472
         /b3 |    1.037594   .1184227     8.76   0.000       .80549    1.269698
------------------------------------------------------------------------------
Instruments for equation 1: x1 x2 x3
```

Testing whether our program works correctly with frequency weights is easy. A frequency-weighted dataset is just a compact form of a larger dataset in which identical observations are omitted and a frequency-weight variable is included to tell us how many times each observation in the smaller dataset appears in the larger dataset. Therefore, we can `expand` our smaller dataset by the frequency-weight variable and then refit our model without specifying frequency weights. If we obtain the same results, our program works correctly. When we type

```
. expand fw
. gmm gmm_poiderivfw, nequations(1) parameters(b1 b2 b3)
> instructions(x1 x2 x3, noconstant) vce(cluster id) onestep
```

we obtain the same results as before.

◁

Stored results

gmm stores the following in e():

Scalars

e(N)	number of observations
e(k)	number of parameters
e(k_eq)	number of equations in e(b)
e(k_eq_model)	number of equations in overall model test
e(k_aux)	number of auxiliary parameters
e(n_moments)	number of moments
e(n_eq)	number of equations in moment-evaluator program
e(Q)	criterion function
e(J)	Hansen J χ^2 statistic
e(J_df)	J statistic degrees of freedom
e(k_*i*)	number of parameters in equation i
e(has_xtinst)	1 if panel-style instruments specified, 0 otherwise
e(N_clust)	number of clusters
e(type)	1 if interactive version, 2 if moment-evaluator program version
e(rank)	rank of e(V)
e(ic)	number of iterations used by iterative GMM estimator
e(converged)	1 if converged, 0 otherwise

Macros

e(cmd)	gmm
e(cmdline)	command as typed
e(title)	title specified in title()
e(title_2)	title specified in title2()
e(clustvar)	name of cluster variable
e(inst_*i*)	equation i instruments
e(eqnames)	equation names
e(winit)	initial weight matrix used
e(winitname)	name of user-supplied initial weight matrix
e(estimator)	onestep, twostep, or igmm
e(rhs)	variables specified in variables()
e(params_*i*)	equation i parameters
e(wmatrix)	*wmtype* specified in wmatrix()
e(vce)	*vcetype* specified in vce()
e(vcetype)	title used to label Std. err.
e(params)	parameter names
e(sexp_*i*)	substitutable expression for equation i
e(evalprog)	moment-evaluator program
e(evalopts)	options passed to moment-evaluator program
e(nocommonesample)	nocommonesample, if specified
e(technique)	optimization technique
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(init)	initial values of the estimators
e(Wuser)	user-supplied initial weight matrix
e(W)	weight matrix used for final round of estimation
e(S)	moment covariance matrix used in robust VCE computations
e(G)	averages of derivatives of moment conditions
e(N_byequation)	number of observations per equation, if nocommonesample specified
e(V)	variance–covariance matrix
e(V_modelbased)	model-based variance

Functions

`e(sample)`	marks estimation sample

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

Let q denote the number of moment conditions. For observation i, $i = 1, \ldots, N$, write the jth moment equation as $\mathbf{z}_{ij}u_{ij}(\boldsymbol{\beta}_j)$ for $j = 1, \ldots, q$. $\mathbf{z}_{ij}$ is a $1 \times m_j$ vector, where m_j is the number of instruments specified for equation j. Let $m = m_1 + \cdots + m_q$.

Our notation can incorporate moment conditions of the form $h_{ij}(\mathbf{w}_{ij}; \boldsymbol{\beta}_j)$ with instruments $\mathbf{w}_{ij}$ by defining $\mathbf{z}_{ij} = 1$ and $u_{ij}(\boldsymbol{\beta}_j) = h_{ij}(\mathbf{w}_{ij}; \beta_j)$, so except when necessary, we do not distinguish between the two types of moment conditions. We could instead use notation so that all our moment conditions are of the form $h_{ij}(\mathbf{w}_{ij}; \beta_j)$, or we could adopt notation that explicitly combines both forms of moment equations. However, because moment conditions of the form $\mathbf{z}_{ij}'u_{ij}(\boldsymbol{\beta}_j)$ are arguably more common, we use that notation.

Let $\boldsymbol{\beta}$ denote a $k \times 1$ vector of parameters, consisting of all the unique parameters of $\boldsymbol{\beta}_1, \ldots, \boldsymbol{\beta}_q$. Then, we can stack the moment conditions and write them more compactly as $\mathbf{Z}_i'\mathbf{u}_i(\boldsymbol{\beta})$, where

$$\mathbf{Z}_i = \begin{bmatrix} \mathbf{z}_{i1} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{z}_{i2} & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{z}_{iq} \end{bmatrix} \quad \text{and} \quad \mathbf{u}_i(\boldsymbol{\beta}) = \begin{bmatrix} u_{i1}(\boldsymbol{\beta}_1) \\ u_{i2}(\boldsymbol{\beta}_2) \\ \vdots \\ u_{iq}(\boldsymbol{\beta}_q) \end{bmatrix}$$

The GMM estimator $\widehat{\boldsymbol{\beta}}$ is the value of β that minimizes

$$Q(\boldsymbol{\beta}) = \left\{ N^{-1} \sum_{i=1}^{N} \mathbf{Z}_i'\mathbf{u}_i(\boldsymbol{\beta}) \right\}' \mathbf{W} \left\{ N^{-1} \sum_{i=1}^{N} \mathbf{Z}_i'\mathbf{u}_i(\boldsymbol{\beta}) \right\} \tag{A1}$$

for $q \times q$ weight matrix $\mathbf{W}$.

By default, `gmm` minimizes (A1) using the Gauss–Newton method. See Hayashi (2000, 498) for a derivation. This technique is typically faster than quasi-Newton methods and does not require second-order derivatives.

Methods and formulas are presented under the following headings:

Initial weight matrix

If you specify `winitial(unadjusted)`, then we create matrix $\mathbf{\Lambda}$ with typical submatrix

$$\mathbf{\Lambda}_{rs} = N^{-1}\sum_{i=1}^{N}\mathbf{z}_{ir}'\mathbf{z}_{is}$$

for $r = 1, \ldots, q$ and $s = 1, \ldots, q$. If you include the `independent` suboption, then we set $\mathbf{\Lambda}_{rs} = \mathbf{0}$ for $r \neq s$. The weight matrix $\mathbf{W}$ equals $\mathbf{\Lambda}^{-1}$.

If you specify `winitial(identity)`, then we set $\mathbf{W} = \mathbf{I}_q$.

If you specify `winitial(xt` *xtspec*`)`, then you must specify one or two items in *xtspec*, one for each equation. `gmm` allows you to specify at most two moment equations when you specify `winitial(xt` *xtspec*`)`, one in first-differences, and one in levels. We create the block-diagonal matrix $\mathbf{H}$ with typical block $\mathbf{H}_j$. If the jth element of *xtspec* is "L", then $\mathbf{H}_j$ is the identity matrix of suitable dimension. If the jth element of *xtspec* is "D", then

$$\mathbf{H}_j = \begin{bmatrix} 1 & -0.5 & 0 & \ldots & 0 & 0 \\ -0.5 & 1 & -0.5 & \ldots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \ldots & 1 & -0.5 \\ 0 & 0 & 0 & \ldots & -0.5 & 1 \end{bmatrix}$$

Then,

$$\mathbf{\Lambda}_H = N_g^{-1}\sum_{g=1}^{g=N_G}\mathbf{Z}_g'\mathbf{H}\mathbf{Z}_g$$

where g indexes panels in the dataset, N_G is the number of panels, $\mathbf{Z}_g$ is the full instrument matrix for panel g, and $\mathbf{W} = \mathbf{\Lambda}_H^{-1}$. See *Panel-style instruments* below for a discussion of how $\mathbf{Z}_g$ is formed.

If you specify `winitial(`*matname*`)`, then we set $\mathbf{W}$ equal to Stata matrix *matname*.

Weight matrix

Specification of the weight matrix applies only to the two-step and iterative estimators. When you use the `onestep` option, the `wmatrix()` option is ignored.

We first evaluate (A1) using the initial weight matrix described above and then compute $\mathbf{u}_i(\widehat{\boldsymbol{\beta}})$. In all cases, $\mathbf{W} = \mathbf{\Lambda}^{-1}$. If you specify `wmatrix(unadjusted)`, then we create $\mathbf{\Lambda}$ to have typical submatrix

$$\mathbf{\Lambda}_{rs} = \sigma_{rs}\,N^{-1}\sum_{i=1}^{N}\mathbf{z}_{ir}'\mathbf{z}_{is}$$

where

$$\sigma_{rs} = N^{-1}\sum_{i=1}^{N}u_{ir}(\widehat{\boldsymbol{\beta}})u_{is}(\widehat{\boldsymbol{\beta}})$$

and r and s index moment equations. For all types of weight matrices, if the `independent` suboption is specified, then $\mathbf{\Lambda}_{rs} = \mathbf{0}$ for $r \neq s$, where $\mathbf{\Lambda}_{rs}$ measures the covariance between moment conditions for equations r and s.

If you specify `wmatrix(robust)`, then

$$\mathbf{\Lambda} = N^{-1}\sum_{i=1}^{N}\mathbf{Z}_i\mathbf{u}_i(\widehat{\boldsymbol{\beta}})\mathbf{u}_i'(\widehat{\boldsymbol{\beta}})\mathbf{Z}_i'$$

If you specify `wmatrix(cluster` *clustvar*`)`, then

$$\mathbf{\Lambda} = N^{-1}\sum_{c=1}^{c=N_C}\mathbf{q}_c\mathbf{q}_c'$$

where c indexes clusters, N_C is the number of clusters, and

$$\mathbf{q}_c = \sum_{i\in c_j}\mathbf{Z}_i\mathbf{u}_i(\widehat{\boldsymbol{\beta}})$$

If you specify `wmatrix(hac` *kernel* $\left[\#\right]$`)`, then

$$\begin{aligned}\mathbf{\Lambda} =& N^{-1}\sum_{i=1}^{N}\mathbf{Z}_i\mathbf{u}_i(\widehat{\boldsymbol{\beta}})\mathbf{u}_i(\widehat{\boldsymbol{\beta}})'\mathbf{Z}_i' \quad + \\ & N^{-1}\sum_{l=1}^{l=n-1}\sum_{i=l+1}^{N}K(l,m)\left\{\mathbf{Z}_i\mathbf{u}_i(\widehat{\boldsymbol{\beta}})\mathbf{u}_{i-l}'(\widehat{\boldsymbol{\beta}})\mathbf{Z}_{i-l}' + \mathbf{Z}_{i-l}\mathbf{u}_{i-l}(\widehat{\boldsymbol{\beta}})\mathbf{u}_i'(\widehat{\boldsymbol{\beta}})\mathbf{Z}_i'\right\}\end{aligned}$$

where $m = \#$ if $\#$ is specified and $m = N-2$ otherwise. Define $z = l/(m+1)$. If *kernel* is `bartlett` or `nwest`, then

$$K(l,m) = \begin{cases}1-z & 0\le z\le 1\\ 0 & \text{otherwise}\end{cases}$$

If *kernel* is `parzen` or `gallant`, then

$$K(l,m) = \begin{cases}1-6z^2+6z^3 & 0\le z\le 0.5\\ 2(1-z)^3 & 0.5< z\le 1\\ 0 & \text{otherwise}\end{cases}$$

If *kernel* is `quadraticspectral` or `andrews`, then

$$K(l,m) = \begin{cases}1 & z=0\\ 3\{\sin(\theta)/\theta-\cos(\theta)\}/\theta^2 & \text{otherwise}\end{cases}$$

where $\theta = 6\pi z/5$.

If `wmatrix(hac` *kernel* `opt)` is specified, then `gmm` uses Newey and West's (1994) automatic lag-selection algorithm, which proceeds as follows. Define $\mathbf{h}$ to be an $m\times 1$ vector of ones. Note that this definition of $\mathbf{h}$ is slightly different from the one used by `ivregress`. There the element of $\mathbf{h}$ corresponding to the constant term equals 0, effectively ignoring the effect of the constant in determining the optimal lag length. Here we include the effect of the constant term. Now, define

$$
\begin{aligned}
f_i &= \{\mathbf{Z}_i'\mathbf{u}_i(\boldsymbol{\beta})\}'\,\mathbf{h} \\
\widehat{\sigma}_j &= N^{-1}\sum_{i=j+1}^{N} f_i f_{i-j} \qquad j = 0,\ldots,m^* \\
\widehat{s}^{(q)} &= 2\sum_{j=1}^{j=m^*} \widehat{\sigma}_j j^q \\
\widehat{s}^{(0)} &= \widehat{\sigma}_0 + 2\sum_{j=1}^{j=m^*} \widehat{\sigma}_j \\
\widehat{\gamma} &= c_\gamma \left\{ \left(\frac{\widehat{s}^{(q)}}{\widehat{s}^{(0)}} \right)^2 \right\}^{1/(2q+1)} \\
m &= \widehat{\gamma} N^{1/(2q+1)}
\end{aligned}
$$

where q, m^*, and c_γ depend on the kernel specified:

Kernel	q	m^*	c_γ
Bartlett/Newey–West	1	int $\{20(T/100)^{2/9}\}$	1.1447
Parzen/Gallant	2	int $\{20(T/100)^{4/25}\}$	2.6614
Quadratic spectral/Andrews	2	int $\{20(T/100)^{2/25}\}$	1.3221

Here int(x) denotes the integer obtained by truncating x toward zero. For the Bartlett and Parzen kernels, the optimal lag is min$\{$int$(m), m^*\}$. For the quadratic spectral kernel, the optimal lag is min$\{m, m^*\}$.

If `wmatrix(hac` *kernel* `opt #)` is specified, then `gmm` uses # instead of 20 in the definition of m^* above to select the optimal lag.

Variance–covariance matrix

If you specify `vce(unadjusted)`, then the VCE matrix is computed as

$$
\text{Var}(\widehat{\boldsymbol{\beta}}) = N^{-1}\left\{ \overline{\mathbf{G}}(\widehat{\boldsymbol{\beta}})'\mathbf{W}\overline{\mathbf{G}}(\widehat{\boldsymbol{\beta}}) \right\}^{-1} \tag{A2}
$$

where

$$
\overline{\mathbf{G}}(\widehat{\boldsymbol{\beta}}) = N^{-1}\sum_{i=1}^{N} \mathbf{Z}_i' \left. \frac{\partial \mathbf{u}_i(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}'} \right|_{\boldsymbol{\beta}=\widehat{\boldsymbol{\beta}}} \tag{A3}
$$

For the two-step and iterated estimators, we use the weight matrix $\mathbf{W}$ that was used to compute the final-round estimate $\widehat{\boldsymbol{\beta}}$.

When you do not specify analytic derivatives, `gmm` must compute the Jacobian matrix (A3) numerically. By default, `gmm` computes each element of the matrix individually by using the Mata `deriv()` function; see [M-5] **deriv( )**. This procedure results in accurate derivatives but can be slow if your model has many instruments or parameters.

When you specify the `quickderivatives` option, `gmm` computes all derivatives corresponding to parameter β_j, $j = 1, \ldots, q$, at once, using two-sided derivatives with a step size of $|\beta_j|\epsilon^{1/3}$, where ϵ is the machine precision of a double precision number (approximately 2.22045×10^{-16}). This method requires just two evaluations of the model's moments to compute an entire column of (A3) and therefore has the most impact when you specify many instruments or moment equations so that (A3) has many rows.

For the one-step estimator, how the unadjusted VCE is computed depends on the type of initial weight matrix requested and the form of the moment equations. If you specify two or more moment equations of the form $h_{ij}(\mathbf{w}_{ij}; \boldsymbol{\beta}_j)$, then `gmm` issues a note and computes a heteroskedasticity-robust VCE because here the matrix $\mathbf{Z}'\mathbf{Z}$ is necessarily singular; moreover, here you must use the identity matrix as the initial weight matrix. Otherwise, if you specify `winitial(unadjusted)` or `winitial(identity)`, then `gmm` first computes an unadjusted weight matrix based on $\widehat{\boldsymbol{\beta}}$ before evaluating (A2). If you specify `winitial(`*matname*`)`, then (A2) is evaluated on the basis of *matname*; the user is responsible for verifying that the VCE and other statistics so produced are appropriate.

All types of robust VCEs computed by `gmm` take the form

$$\text{Var}(\widehat{\boldsymbol{\beta}}) = N^{-1}\left\{\overline{\mathbf{G}}(\widehat{\boldsymbol{\beta}})'\mathbf{W}\overline{\mathbf{G}}(\widehat{\boldsymbol{\beta}})\right\}^{-1}\overline{\mathbf{G}}(\widehat{\boldsymbol{\beta}})'\mathbf{W}\mathbf{S}\mathbf{W}\overline{\mathbf{G}}(\widehat{\boldsymbol{\beta}})\left\{\overline{\mathbf{G}}(\widehat{\boldsymbol{\beta}})'\mathbf{W}\overline{\mathbf{G}}(\widehat{\boldsymbol{\beta}})\right\}^{-1}$$

For the one-step estimator, $\mathbf{W}$ represents the initial weight matrix requested using the `winitial()` option, and $\mathbf{S}$ is computed on the basis of the specification of the `vce()` option. The formulas for the $\mathbf{S}$ matrix are identical to the ones that define the $\boldsymbol{\Lambda}$ matrix in *Weight matrix* above, except that $\mathbf{S}$ is computed after the moment equations are reevaluated using the final estimate of $\widehat{\boldsymbol{\beta}}$. For the two-step and iterated GMM estimators, computation of $\mathbf{W}$ is controlled by the `wmatrix()` option on the basis of the penultimate estimate of $\widehat{\boldsymbol{\beta}}$.

For details on computation of the VCE matrix with dynamic panel-data models, see *Panel-style instruments* below.

Hansen's J statistic

Hansen's (1982) J test of overidentifying restrictions is $J = N \times Q(\widehat{\boldsymbol{\beta}})$. $J \sim \chi^2(m-k)$. If $m < k$, `gmm` issues an error message without estimating the parameters. If $m = k$, the model is just-identified and J is saved as missing ("."). For the two-step and iterated GMM estimators, the J statistic is based on the last-computed weight matrix as determined by the `wmatrix()` option. For the one-step estimator, `gmm` recomputes a weight matrix as described in the second paragraph of *Variance–covariance matrix* above. To obtain Hansen's J statistic, you use `estat overid`; see [R] **gmm postestimation**.

Panel-style instruments

Here we discuss several issues that arise only when you specify panel-style instruments by using the `xtinstruments()` option. When you specify the `xtinstruments()` option, we can no longer consider the instruments for one observation in isolation; instead, we must consider the instrument matrix for an entire panel at once. In the following discussion, we let T denote the number of time periods in a panel. To accommodate unbalanced datasets, conceptually we simply use zeros as instruments and residuals for time periods that are missing in a panel.

We consider the case where you specify both an equation in levels and an equation in differences, yielding two residual equations. Let $u^L_{pt}(\boldsymbol{\beta})$ denote the residual for the level equation for panel p in period t, and let $u^D_{pt}(\boldsymbol{\beta})$ denote the residual for the corresponding difference equation. Now, define the $2T \times 1$ vector $\mathbf{u}_p(\boldsymbol{\beta})$ as

$$\mathbf{u}_p(\boldsymbol{\beta}) = [u^L_{p1}(\boldsymbol{\beta}), u^L_{p2}(\boldsymbol{\beta}), \ldots, u^L_{pT}(\boldsymbol{\beta}), u^D_{p2}(\boldsymbol{\beta}), u^D_{p3}(\boldsymbol{\beta}), \ldots, u^D_{pT}(\boldsymbol{\beta})]$$

The $T+1$ element of $\mathbf{u}_p$ is $u^D_{p2}(\boldsymbol{\beta})$ because we lose the first observation of the difference equation because of differencing.

We write the moment conditions for the pth panel as $\mathbf{Z}_p\mathbf{u}_p(\boldsymbol{\beta})$. To see how $\mathbf{Z}_p$ is defined, we will let $\mathbf{w}^L_{pt}$ and $\mathbf{w}^D_{pt}$ denote the vectors of panel-style instruments for the level and difference equations, respectively, and let time be denoted by t; we discuss their dimensions momentarily. Also let $\mathbf{x}^L_{pt}$ and $\mathbf{x}^D_{pt}$ denote the vectors of instruments specified in `instruments()` for the level and difference equations at time t. Without loss of generality, for our discussion, we assume that you specify the level equation first. Then, $\mathbf{Z}_p$ has the form

$$\mathbf{Z}_p = \begin{bmatrix}
\mathbf{w}^L_1 & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} \\
\mathbf{0} & \mathbf{w}^L_2 & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} \\
\vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\
\mathbf{0} & \mathbf{0} & \cdots & \mathbf{w}^L_T & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} \\
\mathbf{x}^L_1 & \mathbf{x}^L_2 & \cdots & \mathbf{x}^L_T & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} \\
\mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{w}^D_1 & \mathbf{0} & \cdots & \mathbf{0} \\
\mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{w}^D_2 & \cdots & \mathbf{0} \\
\vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\
\mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{w}^D_T \\
\mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{x}^D_1 & \mathbf{x}^D_2 & \cdots & \mathbf{x}^D_T
\end{bmatrix} \tag{A4}$$

To see how the $\mathbf{w}$ vectors are formed, we will suppose you specify

```
xtinstruments(eq(1): d, lags(a/b))
```

Then, $\mathbf{w}^L_t$ will be a $(b-a+1) \times 1$ vector consisting of $d_{t-a}, \ldots, d_{t-b}$. If $(t-a) \leq 0$, then instead, we set $\mathbf{w}^L_t = 0$. If $(t-a) > 0$ but $(t-b) \leq 0$, then we create $\mathbf{w}^L_t$ to consist of $d_{t-a}, \ldots, d_1$. With this definition, $(b-a+1)$ defines the maximum number of lags of d used, but `gmm` will proceed with fewer lags if all $(b-a+1)$ lags are not available. If you specify two panel-style instruments, `d` and `e`, say, then $\mathbf{w}^L_t$ will consist of $d_{t-a}, \ldots, d_{t-b}, e_{t-a}, \ldots, e_{t-b}$. $\mathbf{w}^D_t$ is handled analogously.

The $\mathbf{x}^L_t$ vectors are simply $j \times 1$ vectors, where j is the number of regular instruments specified with the `instruments()` option; these vectors include a "1" unless you specify the `noconstant` suboption.

Looking carefully at (A4), you will notice that for dynamic panel-data models, moment conditions corresponding to the instruments $\mathbf{x}^L_{pt}$ take the form

$$E\left[\sum_{t=1}^{t=T} \mathbf{x}^L_{pt} u^L_{pt}(\boldsymbol{\beta})\right] = \mathbf{0}$$

and likewise for $\mathbf{x}^D_{pt}$. Instead of having separate moment conditions for each time period, there is one moment condition equal to the average of individual periods' moments. See Arellano and Bond (1991, 280). To include separate moment conditions for each time period, instead of specifying, say,

```
instruments(1: x)
```

you could instead first generate a variable called one equal to unity for all observations and specify

```
xtinstruments(1: x one)
```

(Creating the variable one is necessary because a constant is not automatically included in variable lists specified in xtinstruments().)

Unbalanced panels are essentially handled by including zeros in rows and columns of $\mathbf{Z}_p$ and $\mathbf{u}_p(\boldsymbol{\beta})$ corresponding to missing time periods. However, the numbers of instruments and moment conditions reported by gmm do not reflect this trickery and instead reflect the numbers of instruments and moment conditions that are not manipulated in this way. Moreover, gmm includes code to work through these situations efficiently without actually having to fill in zeros.

When you specify winitial(xt ...), the one-step unadjusted VCE is computed as

$$\text{Var}(\widehat{\boldsymbol{\beta}}) = \widehat{\sigma}_1^2 \boldsymbol{\Lambda}_H$$

where $\boldsymbol{\Lambda}_H$ was defined previously as

$$\widehat{\sigma}_1^2 = (N-k)^{-1} \sum_{p=1}^{p=P} \mathbf{u}_p^D(\widehat{\boldsymbol{\beta}})$$

and $\mathbf{u}_p^D(\widehat{\boldsymbol{\beta}}) = [u_{p2}^D(\widehat{\boldsymbol{\beta}}), \ldots, u_{pT}^D(\widehat{\boldsymbol{\beta}})]$. Here we use $(N-k)^{-1}$ instead of N^{-1} to match xtdpd.

Marginal predictions with unconditional standard errors

Here we describe how margins computes unconditional standard errors when used with the vce(unconditional) option after gmm. These standard errors account for the estimation of parameters in gmm before margins is used to make marginal predictions. They also account for variation in the covariates over the population.

margins with the vce(unconditional) option uses linearization to estimate the unconditional variance of $\widehat{\boldsymbol{\beta}}$. Linearization uses the variance estimator for the total of a score variable for the marginal prediction $p(\widehat{\boldsymbol{\beta}})$ as an approximate estimator for $\text{Var}\{p(\widehat{\boldsymbol{\beta}})\}$. See [SVY] **Variance estimation** for more details. Our derivation of the standard errors here is similar to the derivation of the standard errors for two-step M estimators. See Wooldridge (2010, sec. 12.4) for the latter.

Let $\mathbf{x}_i$ be a vector of covariate values, which includes all variables used in calculating the moment conditions, and let $f(\mathbf{x}_i, \boldsymbol{\beta})$ be a scalar-valued function returning the value of the predictions of interest.

margins computes estimates of

$$\theta = p(\boldsymbol{\beta}) = E\left\{f(\mathbf{x}_i, \boldsymbol{\beta}) | \mathbf{x}_i \in S_p\right\}$$

where S_p is a subpopulation of interest.

The indicator $\delta_i(S_p)$ indicates whether observation i is in subpopulation S_p,

$$\delta_i(S_p) = \begin{cases} 1, & i \in S_p \\ 0, & i \notin S_p \end{cases}$$

Let $\widehat{\boldsymbol{\beta}}$ be the vector of parameter estimates. Then, `margins` estimates $\theta = p(\boldsymbol{\beta})$ via

$$\widehat{\theta} = \frac{1}{w.} \sum_{i=1}^{N} \delta_i(S_p) w_i f(\mathbf{x}_i, \widehat{\boldsymbol{\beta}})$$

where

$$w. = \sum_{i=1}^{N} \delta_i(S_p) w_i$$

and w_i is the weight for the ith observation.

In minimizing (A1), we see that the GMM estimator $\widehat{\boldsymbol{\beta}}$ is the value of $\boldsymbol{\beta}$ that solves the score equations

$$\mathbf{0} = \overline{\mathbf{G}}(\boldsymbol{\beta})' \mathbf{W} \left\{ N^{-1} \sum_{i=1}^{N} \mathbf{Z}_i' \mathbf{u}_i(\boldsymbol{\beta}) \right\}$$

where $\overline{\mathbf{G}}(\boldsymbol{\beta})$ was defined in (A3).

By the mean-value theorem, for some points $\boldsymbol{\beta}_1, \ldots, \boldsymbol{\beta}_q$ between $\boldsymbol{\beta}$ and $\widehat{\boldsymbol{\beta}}$, we have

$$\mathbf{0} = \overline{\mathbf{G}}(\widehat{\boldsymbol{\beta}})' \mathbf{W} \left[\left\{ N^{-1} \sum_{i=1}^{N} \mathbf{Z}_i' \mathbf{u}_i(\boldsymbol{\beta}) \right\} + \overline{\mathbf{G}}_m \left(\widehat{\boldsymbol{\beta}} - \boldsymbol{\beta} \right) \right]$$

where

$$\overline{\mathbf{G}}_m = N^{-1} \sum_{i=1}^{N} \left[\frac{\partial \left\{ \mathbf{Z}_i' \mathbf{u}_i(\boldsymbol{\beta}_j) \right\}_j}{\partial \boldsymbol{\beta}_j'} \right]_{j,l}$$

So we have

$$\sqrt{N} \left(\widehat{\boldsymbol{\beta}} - \boldsymbol{\beta} \right) = - \left\{ \overline{\mathbf{G}}(\widehat{\boldsymbol{\beta}})' \mathbf{W} \overline{\mathbf{G}}_m \right\}^{-1} \overline{\mathbf{G}}(\widehat{\boldsymbol{\beta}})' \mathbf{W} \left\{ N^{-0.5} \sum_{i=1}^{N} \mathbf{Z}_i' \mathbf{u}_i(\boldsymbol{\beta}) \right\} \tag{A5}$$

The margin $\widehat{\theta}$ is the solution to the score equations

$$\sum_{i=1}^{N} s_i(\theta, \widehat{\boldsymbol{\beta}}) = \mathbf{0}$$

where

$$s_i(\theta, \boldsymbol{\beta}) = w_i \delta_i(S_p) \left\{ f(\mathbf{x}_i, \boldsymbol{\beta}) - \theta \right\}$$

When we do a mean-value expansion about point θ_1 between θ and $\widehat{\theta}$, we get

$$\mathbf{0} = \sum_{i=1}^{N} s_i(\theta, \widehat{\boldsymbol{\beta}}) - w.(\widehat{\theta} - \theta)$$

So we have

$$\sqrt{N}(\widehat{\theta} - \theta) = w.^{-1} N^{-0.5} \sum_{i=1}^{N} s_i(\theta, \widehat{\boldsymbol{\beta}})$$

Using the mean-value theorem again, for point $\boldsymbol{\beta}_m$ between $\boldsymbol{\beta}$ and $\widehat{\boldsymbol{\beta}}$, we have

$$w.^{-1} N^{-0.5} \sum_{i=1}^{N} s_i(\theta, \widehat{\boldsymbol{\beta}}) = w.^{-1} \left\{ N^{-0.5} \sum_{i=1}^{N} s_i(\theta, \boldsymbol{\beta}) + \sqrt{N} \overline{\mathbf{J}}(\boldsymbol{\beta}_m) \left(\widehat{\boldsymbol{\beta}} - \boldsymbol{\beta} \right) \right\}$$

where $\overline{\mathbf{J}}(\boldsymbol{\beta}_m)$ is the Jacobian of the margin at $\boldsymbol{\beta}_m$,

$$\overline{\mathbf{J}}(\boldsymbol{\beta}_m) = \left\{ N^{-1} \sum_{i=1}^{N} w_i \delta_i(S_p) \frac{\partial f(\mathbf{x}_i, \boldsymbol{\beta}_m)}{\partial \boldsymbol{\beta}_m'} \right\}$$

Using (A5), we get

$$\sqrt{N}(\widehat{\theta} - \theta) = w_{\cdot}^{-1} N^{-0.5} \left[\sum_{i=1}^{N} s_i(\theta, \boldsymbol{\beta}) - \overline{\mathbf{J}}(\boldsymbol{\beta}_m) \left\{ \overline{\mathbf{G}}(\widehat{\boldsymbol{\beta}})' \mathbf{W} \overline{\mathbf{G}}_m \right\}^{-1} \overline{\mathbf{G}}(\widehat{\boldsymbol{\beta}})' \mathbf{W} \left\{ \sum_{i=1}^{N} \mathbf{Z}_i' \mathbf{u}_i(\boldsymbol{\beta}) \right\} \right]$$

$\widehat{\theta}$ is asymptotically normal, and a consistent estimator of its variance is given by

$$\widehat{\text{Var}}\{\sqrt{N}(\widehat{\theta} - \theta)\} = \sum_{i=1}^{N} w_{\cdot}^{-2} \left[s_i(\widehat{\theta}, \widehat{\boldsymbol{\beta}}) - \overline{\mathbf{J}}(\widehat{\boldsymbol{\beta}}) \left\{ \overline{\mathbf{G}}(\widehat{\boldsymbol{\beta}})' \mathbf{W} \overline{\mathbf{G}}(\widehat{\boldsymbol{\beta}}) \right\}^{-1} \overline{\mathbf{G}}(\widehat{\boldsymbol{\beta}})' \mathbf{W} \mathbf{Z}_i' \mathbf{u}_i(\widehat{\boldsymbol{\beta}}) \right]^2$$

See Wooldridge (2010, sec 12.4 and 12.5) for details.

`gmm` returns $N^{-1} \left\{ \overline{\mathbf{G}}(\widehat{\boldsymbol{\beta}})' \mathbf{W} \overline{\mathbf{G}}(\widehat{\boldsymbol{\beta}}) \right\}^{-1}$ as the model-based variance. The scores are $-\overline{\mathbf{G}}(\widehat{\boldsymbol{\beta}})' \mathbf{W} \mathbf{Z}_i' \mathbf{u}_i(\widehat{\boldsymbol{\beta}})$ and may be predicted in postestimation; see [R] **gmm postestimation**. These scores correspond to derivatives of the criterion function $Q(\beta)$, scaled by $-1/2$. See Cameron and Trivedi (2005, sec. 6.3.2) for more details.

`margins` estimates the asymptotic standard error of $\widehat{\theta}$ from the model-based variance, the scores, and its own predictions of s_i and the Jacobian $\overline{\mathbf{J}}(\widehat{\boldsymbol{\beta}})$.

References

Andrews, D. W. K., W. Kim, and X. Shi. 2017. Commands for testing conditional moment inequalities and equalities. *Stata Journal* 17: 56–72.

Arellano, M., and S. Bond. 1991. Some tests of specification for panel data: Monte Carlo evidence and an application to employment equations. *Review of Economic Studies* 58: 277–297. https://doi.org/10.2307/2297968.

Baum, C. F. 2006. *An Introduction to Modern Econometrics Using Stata*. College Station, TX: Stata Press.

Blundell, R. W., R. Griffith, and F. Windmeijer. 2002. Individual effects and dynamics in count data models. *Journal of Econometrics* 108: 113–131. https://doi.org/10.1016/S0304-4076(01)00108-7.

Cameron, A. C., and P. K. Trivedi. 2005. *Microeconometrics: Methods and Applications*. New York: Cambridge University Press.

——. 2013. *Regression Analysis of Count Data*. 2nd ed. New York: Cambridge University Press.

——. 2022. *Microeconometrics Using Stata*. 2nd ed. College Station, TX: Stata Press.

Cattaneo, M. D. 2010. Efficient semiparametric estimation of multi-valued treatment effects under ignorability. *Journal of Econometrics* 155: 138–154. https://doi.org/10.1016/j.jeconom.2009.09.023.

Chamberlain, G. 1992. Comment: Sequential moment restrictions in panel data. *Journal of Business and Economic Statistics* 10: 20–26. https://doi.org/10.2307/1391799.

Clarke, D., and B. Matta. 2018. Practical considerations for questionable IVs. *Stata Journal* 18: 663–691.

Davidson, R., and J. G. MacKinnon. 1993. *Estimation and Inference in Econometrics*. New York: Oxford University Press.

Doris, A., D. O'Neill, and O. Sweetman. 2011. GMM estimation of the covariance structure of longitudinal data on earnings. *Stata Journal* 11: 439–459.

Drukker, D. M. 2014. Using gmm to solve two-step estimation problems. *The Stata Blog: Not Elsewhere Classified.* http://blog.stata.com/2014/12/08/using-gmm-to-solve-two-step-estimation-problems/.

——. 2015. 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/.

——. 2016. An ordered-probit inverse probability weighted (IPW) estimator. *The Stata Blog: Not Elsewhere Classified.* http://blog.stata.com/2016/09/13/an-ordered-probit-inverse-probability-weighted-ipw-estimator/.

Erickson, T., R. Parham, and T. M. Whited. 2017. Fitting the errors-in-variables model using high-order cumulants and moments. *Stata Journal* 17: 116–129.

Flynn, Z. L., and L. M. Magnusson. 2013. Parametric inference using structural break tests. *Stata Journal* 13: 836–861.

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. 2018. *Econometric Analysis*. 8th ed. New York: Pearson.

Hall, A. R. 2005. *Generalized Method of Moments*. Oxford: Oxford University Press.

Hamilton, J. D. 1994. *Time Series Analysis*. Princeton, NJ: Princeton University Press.

Hansen, L. P. 1982. Large sample properties of generalized method of moments estimators. *Econometrica* 50: 1029–1054. https://doi.org/10.2307/1912775.

Hansen, L. P., and K. J. Singleton. 1982. Generalized instrumental variables estimation of nonlinear rational expectations models. *Econometrica* 50: 1269–1286. https://doi.org/10.2307/1911873.

Hayashi, F. 2000. *Econometrics*. Princeton, NJ: Princeton University Press.

Lindsey, C. 2016a. Estimating covariate effects after gmm. *The Stata Blog: Not Elsewhere Classified.* http://blog.stata.com/2016/10/04/estimating-covariate-effects-after-gmm/.

——. 2016b. Testing model specification and using the program version of gmm. *The Stata Blog: Not Elsewhere Classified.* http://blog.stata.com/2016/02/11/testing-model-specification-and-using-the-program-version-of-gmm/.

Lindsey, C., and J. Luedicke. 2016. Solving missing data problems using inverse-probability-weighted estimators. *The Stata Blog: Not Elsewhere Classified.* http://blog.stata.com/2016/10/11/solving-missing-data-problems-using-inverse-probability-weighted-estimators/.

Lindsey, C., and E. Pinzon. 2016. Multiple-equation models: Estimation and marginal effects using gmm. *The Stata Blog: Not Elsewhere Classified.* http://blog.stata.com/2016/08/02/multiple-equation-models-estimation-and-marginal-effects-using-gmm/.

Manski, C. F. 1988. *Analog Estimation Methods in Econometrics*. New York: Chapman & Hall/CRC.

Mátyás, L. 1999. *Generalized Method of Moments Estimation*. Cambridge: Cambridge University Press.

Mullahy, J. 1997. Instrumental-variable estimation of count data models: Applications to models of cigarette smoking behavior. *Review of Economics and Statistics* 79: 586–593. https://doi.org/10.1162/003465397557169.

Newey, W. K., and K. D. West. 1994. Automatic lag selection in covariance matrix estimation. *Review of Economic Studies* 61: 631–653. https://doi.org/10.2307/2297912.

Nickell, S. J. 1981. Biases in dynamic models with fixed effects. *Econometrica* 49: 1417–1426. https://doi.org/10.2307/1911408.

Pinzon, E. 2016. Estimation under omitted confounders, endogeneity, omitted variable bias, and related problems. *The Stata Blog: Not Elsewhere Classified.* http://blog.stata.com/2017/02/20/estimation-under-omitted-confounders-endogeneity-omitted-variable-bias-and-related-problems/.

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/.

——. 2016. Tests of forecast accuracy and forecast encompassing. *The Stata Blog: Not Elsewhere Classified.* http://blog.stata.com/2016/06/01/tests-of-forecast-accuracy-and-forecast-encompassing/.

Rios-Avila, F., and G. Canavire-Bacarreza. 2018. Standard-error correction in two-stage optimization models: A quasi–maximum likelihood estimation approach. *Stata Journal* 18: 206–222.

Ruud, P. A. 2000. *An Introduction to Classical Econometric Theory*. New York: Oxford University Press.

Wilde, J. 2008. A note on GMM estimation of probit models with endogenous regressors. *Statistical Papers* 49: 471–484. https://doi.org/10.1007/s00362-006-0027-2.

Windmeijer, F. 2000. Moment conditions for fixed effects count data models with endogenous regressors. *Economics Letters* 68: 21–24. https://doi.org/10.1016/S0165-1765(00)00228-7.

——. 2005. A finite sample correction for the variance of linear efficient two-step GMM estimators. *Journal of Econometrics* 126: 25–51. https://doi.org/10.1016/j.jeconom.2004.02.005.

Windmeijer, F., and J. M. C. Santos Silva. 1997. Endogeneity in count data models: An application to demand for health care. *Journal of Applied Econometrics* 12: 281–294. https://doi.org/10.1002/(SICI)1099-1255(199705)12:3⟨281::AID-JAE436⟩3.0.CO;2-1.

Wooldridge, J. M. 1997. Multiplicative panel data models without the strict exogeneity assumption. *Econometric Theory* 13: 667–678. https://doi.org/10.1017/S0266466600006125.

——. 1999. Distribution-free estimation of some nonlinear panel data models. *Journal of Econometrics* 90: 77–97. https://doi.org/10.1016/S0304-4076(98)00033-5.

——. 2010. *Econometric Analysis of Cross Section and Panel Data*. 2nd ed. Cambridge, MA: MIT Press.

Ye, X., and Y. Sun. 2018. Heteroskedasticity- and autocorrelation-robust F and t tests in Stata. *Stata Journal* 18: 951–980.

Also see

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

[R] **ivregress** — Single-equation instrumental-variables regression

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

[R] **mlexp** — Maximum likelihood estimation of user-specified expressions

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

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

[XT] **xtabond** — Arellano–Bond linear dynamic panel-data estimation

[XT] **xtdpd** — Linear dynamic panel-data estimation

[XT] **xtdpdsys** — Arellano–Bover/Blundell–Bond linear dynamic panel-data estimation

[U] 20 Estimation and postestimation commands