# Extending Stata's capabilities for asymptotic covariance matrix estimation

Christopher F Baum & Mark E Schaffer

Boston College/DIW Berlin     Heriot–Watt University/CEPR/IZA

UK Stata Users Group Meeting, September 2014

# Overview

Our `avar` Stata routine constructs the “filling” for a number of flavors of “sandwich” covariance matrix estimators, including HAC, one- and two-way clustering, common cross-panel autocorrelated errors, etc.

We show how `avar` may be used as a building block to construct VCEs that go beyond the Eicker–Huber–White and one-way cluster-robust VCEs provided by official Stata’s `_robust` command. The `avar` routine may also be used to provide multiple-equation VCE estimates in circumstances not handled by official Stata’s `suest` command. Finally, we demonstrate how `avar`’s capabilities may be utilized in a general-purpose GMM–CUE estimator.

# Overview

Our `avar` Stata routine constructs the “filling” for a number of flavors of “sandwich” covariance matrix estimators, including HAC, one- and two-way clustering, common cross-panel autocorrelated errors, etc.

We show how `avar` may be used as a building block to construct VCEs that go beyond the Eicker–Huber–White and one-way cluster-robust VCEs provided by official Stata's `_robust` command. The `avar` routine may also be used to provide multiple-equation VCE estimates in circumstances not handled by official Stata's `suest` command. Finally, we demonstrate how `avar`'s capabilities may be utilized in a general-purpose GMM–CUE estimator.

# Introduction

`avar` is a routine for estimating $S$, the asymptotic variance of $(1/N)Z'e$, where $Z$ is an $N \times L$ matrix of variables, $e$ is an $N \times p$ matrix of residuals, and $N$ is the sample size. Typically, $S$ would be used to form a sandwich-type estimate of the variance of an estimator, where $S$ is the “filling" of the sandwich.

`avar` can estimate VCEs for single and multiple equations that are robust to various violations of the assumption of *i.i.d.* errors, including heteroskedasticity, autocorrelation, one- and two-way clustering, common cross-panel disturbances, etc. It supports time-series and panel data.

# Introduction

`avar` is a routine for estimating $S$, the asymptotic variance of $(1/N)Z'e$, where $Z$ is an $N \times L$ matrix of variables, $e$ is an $N \times p$ matrix of residuals, and $N$ is the sample size. Typically, $S$ would be used to form a sandwich-type estimate of the variance of an estimator, where $S$ is the “filling" of the sandwich.

`avar` can estimate VCEs for single and multiple equations that are robust to various violations of the assumption of *i.i.d.* errors, including heteroskedasticity, autocorrelation, one- and two-way clustering, common cross-panel disturbances, etc. It supports time-series and panel data.

For example, in the most basic application, linear regression, $e$ is a $N \times 1$ vector of residuals from an OLS estimation and $Z$ is the $N \times L$ matrix of regressors.

The variance of the OLS estimator, $Var(\hat{\beta})$, can be estimated by the “sandwich" $N \times DSD$, where the “filling" $S$ is the estimate of the asymptotic variance of $(1/N)Z'e$ and the “bread" is $D = (1/N)(X'X)^{-1}$.

This estimate of the variance of the OLS estimator is as robust as is $S$. For instance, if $S$ is robust to heteroskedasticity, $Var(\hat{\beta})$ will be robust as well. If $S$ is robust to two-way clustering, so will be $Var(\hat{\beta})$.

For example, in the most basic application, linear regression, $e$ is a $N \times 1$ vector of residuals from an OLS estimation and $Z$ is the $N \times L$ matrix of regressors.

The variance of the OLS estimator, $Var(\hat{\beta})$, can be estimated by the "sandwich" $N \times DSD$, where the "filling" $S$ is the estimate of the asymptotic variance of $(1/N)Z'e$ and the "bread" is $D = (1/N)(X'X)^{-1}$.

This estimate of the variance of the OLS estimator is as robust as is $S$. For instance, if $S$ is robust to heteroskedasticity, $Var(\hat{\beta})$ will be robust as well. If $S$ is robust to two-way clustering, so will be $Var(\hat{\beta})$.

For example, in the most basic application, linear regression, $e$ is a $N \times 1$ vector of residuals from an OLS estimation and $Z$ is the $N \times L$ matrix of regressors.

The variance of the OLS estimator, $Var(\hat{\beta})$, can be estimated by the "sandwich" $N \times DSD$, where the "filling" $S$ is the estimate of the asymptotic variance of $(1/N)Z'e$ and the "bread" is $D = (1/N)(X'X)^{-1}$.

This estimate of the variance of the OLS estimator is as robust as is $S$. For instance, if $S$ is robust to heteroskedasticity, $Var(\hat{\beta})$ will be robust as well. If $S$ is robust to two-way clustering, so will be $Var(\hat{\beta})$.

In the multiple-equation context, $e$ is a $N \times p$ matrix of residuals from $p$ different equations. `avar` will return an estimate of the asymptotic VCE, including the covariances across equations. `avar` can estimate an $S$ that coincides with the $Var(\hat{\beta})$ reported by the Stata routines `sureg`, `reg3` and `suest`.

In its current implementation, `avar` employs listwise deletion. This means it will only use observations for which there are no missing values in any of the variables in the `evarlist`. This is the behavior of `sureg` and `reg3`, but not that of `suest` or `gsem`.

In the multiple-equation context, $e$ is a $N \times p$ matrix of residuals from $p$ different equations. `avar` will return an estimate of the asymptotic VCE, including the covariances across equations. `avar` can estimate an $S$ that coincides with the $Var(\hat{\beta})$ reported by the Stata routines `sureg`, `reg3` and `suest`.

In its current implementation, `avar` employs listwise deletion. This means it will only use observations for which there are no missing values in any of the variables in the `evarlist`. This is the behavior of `sureg` and `reg3`, but not that of `suest` or `gsem`.

# Syntax

In the single-equation context:

`avar` *evarname* `(`*zvarlist*`)` [*weight*] [*if*] [*in*] `[, vce_options misc_options]`

In the multiple-equation context:

`avar (`*evarlist*`)` `(`*zvarlist*`)` [*weight*] [*if*] [*in*] `[, vce_options misc_options]`

The `vce_options` include:

- robust: heteroskedasticity-robust VCE
- bw(#): bandwidth for kernel-robust (AC or HAC) VCE
- kernel(*string*): kernel for kernel-robust (AC or HAC) VCE
- cluster(*cvarlist*): one- or two-way cluster-robust VCE; may be combined with bw(#) if data are tsset and one of the variables in *cvarlist* is the time variable
- dkraay(#): VCE robust to autocorrelated cross-panel disturbances with bandwidth=#, per Driscoll and Kraay (*REStat*, 1998); equivalent to cluster(*tvar*) + bw(#) where *tvar* is the time variable
- kiefer: VCE robust to within-panel autocorrelation, per Kiefer (*J.Econometrics*, 1980): equivalent to kernel(tru) + bw(#) where # is the full length of the panel

`avar` uses the Mata library `livreg2.mlib` used by `ivreg2`, `ranktest` and related routines. For a more detailed discussion of the various VCE options see the `ivreg2` help file.

The `vce_options` include:

- robust: heteroskedasticity-robust VCE
- bw(#): bandwidth for kernel-robust (AC or HAC) VCE
- kernel(*string*): kernel for kernel-robust (AC or HAC) VCE
- cluster(*cvarlist*): one- or two-way cluster-robust VCE; may be combined with bw(#) if data are tsset and one of the variables in *cvarlist* is the time variable
- dkraay(#): VCE robust to autocorrelated cross-panel disturbances with bandwidth=#, per Driscoll and Kraay (*REStat*, 1998); equivalent to cluster(*tvar*) + bw(#) where *tvar* is the time variable
- kiefer: VCE robust to within-panel autocorrelation, per Kiefer (*J.Econometrics*, 1980): equivalent to kernel(tru) + bw(#) where # is the full length of the panel

`avar` uses the Mata library `livreg2.mlib` used by `ivreg2`, `ranktest` and related routines. For a more detailed discussion of the various VCE options see the `ivreg2` help file.

The `vce_options` include:

- robust: heteroskedasticity-robust VCE
- bw(#): bandwidth for kernel-robust (AC or HAC) VCE
- kernel(*string*): kernel for kernel-robust (AC or HAC) VCE
- cluster(*cvarlist*): one- or two-way cluster-robust VCE; may be combined with bw(#) if data are tsset and one of the variables in *cvarlist* is the time variable
- dkraay(#): VCE robust to autocorrelated cross-panel disturbances with bandwidth=#, per Driscoll and Kraay (*REStat*, 1998); equivalent to cluster(*tvar*) + bw(#) where *tvar* is the time variable
- kiefer: VCE robust to within-panel autocorrelation, per Kiefer (*J.Econometrics*, 1980): equivalent to kernel(tru) + bw(#) where # is the full length of the panel

`avar` uses the Mata library `livreg2.mlib` used by `ivreg2`, `ranktest` and related routines. For a more detailed discussion of the various VCE options see the `ivreg2` help file.

The `vce_options` include:

- robust: heteroskedasticity-robust VCE
- bw(#): bandwidth for kernel-robust (AC or HAC) VCE
- kernel(*string*): kernel for kernel-robust (AC or HAC) VCE
- cluster(*cvarlist*): one- or two-way cluster-robust VCE; may be combined with bw(#) if data are tsset and one of the variables in *cvarlist* is the time variable
- dkraay(#): VCE robust to autocorrelated cross-panel disturbances with bandwidth=#, per Driscoll and Kraay (*REStat*, 1998); equivalent to cluster(*tvar*) + bw(#) where *tvar* is the time variable
- kiefer: VCE robust to within-panel autocorrelation, per Kiefer (*J.Econometrics*, 1980): equivalent to kernel(tru) + bw(#) where # is the full length of the panel

`avar` uses the Mata library `livreg2.mlib` used by `ivreg2`, `ranktest` and related routines. For a more detailed discussion of the various VCE options see the `ivreg2` help file.

The `vce_options` include:

- robust: heteroskedasticity-robust VCE
- bw(#): bandwidth for kernel-robust (AC or HAC) VCE
- kernel(*string*): kernel for kernel-robust (AC or HAC) VCE
- cluster(*cvarlist*): one- or two-way cluster-robust VCE; may be combined with bw(#) if data are tsset and one of the variables in *cvarlist* is the time variable
- dkraay(#): VCE robust to autocorrelated cross-panel disturbances with bandwidth=#, per Driscoll and Kraay (*REStat*, 1998); equivalent to cluster(*tvar*) + bw(#) where *tvar* is the time variable
- kiefer: VCE robust to within-panel autocorrelation, per Kiefer (*J.Econometrics*, 1980): equivalent to kernel(tru) + bw(#) where # is the full length of the panel

`avar` uses the Mata library `livreg2.mlib` used by `ivreg2`, `ranktest` and related routines. For a more detailed discussion of the various VCE options see the `ivreg2` help file.

The `vce_options` include:

- robust: heteroskedasticity-robust VCE
- bw(#): bandwidth for kernel-robust (AC or HAC) VCE
- kernel(*string*): kernel for kernel-robust (AC or HAC) VCE
- cluster(*cvarlist*): one- or two-way cluster-robust VCE; may be combined with bw(#) if data are tsset and one of the variables in *cvarlist* is the time variable
- dkraay(#): VCE robust to autocorrelated cross-panel disturbances with bandwidth=#, per Driscoll and Kraay (*REStat*, 1998); equivalent to cluster(*tvar*) + bw(#) where *tvar* is the time variable
- kiefer: VCE robust to within-panel autocorrelation, per Kiefer (*J.Econometrics*, 1980): equivalent to kernel(tru) + bw(#) where # is the full length of the panel

`avar` uses the Mata library `livreg2.mlib` used by `ivreg2`, `ranktest` and related routines. For a more detailed discussion of the various VCE options see the `ivreg2` help file.

The `misc_options` include:

- partial(*pvarlist*): partial out variables in *pvarlist* from variables in *evarlist* and *zvarlist*; constant also partialled-out
- noconstant: suppress constant from *zvarlist*
- demean: center (demean) variables in *evarlist* and *zvarlist* (implies `noconstant`)
- smata(*mname*): name for `avar` matrix left in Mata environment if requested

The `misc_options` include:

- partial(*pvarlist*): partial out variables in *pvarlist* from variables in *evarlist* and *zvarlist*; constant also partialled-out
- noconstant: suppress constant from *zvarlist*
- demean: center (demean) variables in *evarlist* and *zvarlist* (implies `noconstant`)
- smata(*mname*): name for `avar` matrix left in Mata environment if requested

The `misc_options` include:

- partial(*pvarlist*): partial out variables in *pvarlist* from variables in *evarlist* and *zvarlist*; constant also partialled-out
- noconstant: suppress constant from *zvarlist*
- demean: center (demean) variables in *evarlist* and *zvarlist* (implies `noconstant`)
- smata(*mname*): name for `avar` matrix left in Mata environment if requested

The `misc_options` include:

- partial(*pvarlist*): partial out variables in *pvarlist* from variables in *evarlist* and *zvarlist*; constant also partialled-out
- noconstant: suppress constant from *zvarlist*
- demean: center (demean) variables in *evarlist* and *zvarlist* (implies `noconstant`)
- smata(*mname*): name for `avar` matrix left in Mata environment if requested

# Examples

To illustrate the use of `avar`:

```
. set obs 100
obs was 0, now 100
. set seed 12345
.
. gen double y1 = runiform()
. gen double y2 = runiform()
. gen double x1 = runiform()
. gen double x2 = runiform()
. gen double z1 = runiform()
```

```
. qui reg y1 x1 x2
. predict double e1, resid
. qui reg y2 x1 x2
. predict double e2, resid
```

```
. gen int id1 = _n
. gen int t1  = _n
. tsset t1
        time variable:  t1, 1 to 100
                delta:  1 unit
. gen int id2 = ceil(_n/5)
. gen int t2  = 5-(id2*5-t1)
```

```
. avar e1 (x1 x2), smata(my_avar_matrix)
(obs=100)
              x1          x2       _cons
   x1   .02582509
   x2    .0215673   .03283692
_cons   .04087871   .04725347   .09002288
. mata: my_avar_matrix
[symmetric]
                  1              2              3
    +----------------------------------------------+
  1 |  .0258250907                                 |
  2 |  .0215673004    .0328369188                  |
  3 |  .0408787115    .0472534679    .0900228813   |
    +----------------------------------------------+
```

# To reproduce the VCE for regression with robust standard errors:

```
. qui reg y1 x1 x2, robust
. mat V1 = e(V)
. mat accum XX=x1 x2
(obs=100)
. mat Sxx=XX*1/r(N)
. mat Sxxi=syminv(Sxx)
. avar e1 (x1 x2), robust
(obs=100)
                x1          x2       _cons
   x1     .0254583
   x2    .02255302   .03282114
_cons    .04091317   .04773456   .09002288
. mat S=r(S)
. mat V2 = Sxxi*S*Sxxi*1/r(N)
. mat V2 = V2 * e(N)/e(df_r)
. mat check = V1 - V2
. mat list check
symmetric check[3,3]
                 x1           x2        _cons
   x1             0
   x2     2.168e-19   -1.561e-17
_cons    -3.469e-18    1.388e-17   -8.674e-18
```

# To reproduce the VCE for regression with HAC standard errors:

```
. qui newey y1 x1 x2, lag(3)
. mat V1 = e(V)
. mat accum XX=x1 x2
(obs=100)
. mat Sxx=XX*1/r(N)
. mat Sxxi=syminv(Sxx)
. avar e1 (x1 x2), rob bw(4) kernel(bartlett)
(obs=100)
                 x1          x2       _cons
   x1   .02244546
   x2   .01749385   .02439654
_cons   .03122968   .03308884   .06511836
. mat S=r(S)
. mat V2 = Sxxi*S*Sxxi*1/r(N)
. mat V2 = V2 * e(N)/e(df_r)
. mat check = V1 - V2
. mat list check
symmetric check[3,3]
                  x1           x2        _cons
   x1  -5.204e-18
   x2  -4.337e-19  -5.204e-18
_cons   8.674e-19  -5.204e-18   4.337e-18
```

Illustrate the use of `cluster` and `partial` options:

```
. avar e1 (x1 x2), cluster(id2) partial(z1)
(obs=100)
             x1          x2
x1   .00941314
x2   .00201633   .00678255
```

Illustrate use with multiple equations:

```
. avar (e1 e2) (x1 x2), robust
(obs=100)
                  e1:          e1:          e1:          e2:          e2:          e
> 2:
                  x1           x2        _cons           x1           x2        _con
> s
   e1:x1     .0254583
   e1:x2    .02255302    .03282114
e1:_cons    .04091317    .04773456    .09002288
   e2:x1   -.00227822   -.00121366   -.00331922    .02699087
   e2:x2   -.00121366    .00161014    .00109857    .02206868     .0330155
e2:_cons   -.00331922    .00109857   -.00102351    .04024825    .04646162    .0856638
> 5
```

Note that this covariance matrix cannot be produced by `sureg`, as it only supports a classical VCE.

To reproduce the cluster-robust VCE produced by `suest`:

```
. qui mat accum XX=x1 x2
. mat Sxx=XX*1/r(N)
. mat Sxxi=syminv(Sxx)
. qui reg y1 x1 x2
. est store eq_1
. qui reg y2 x1 x2
. est store eq_2
. qui suest eq_1 eq_2, cluster(id2)
. mat V1 = e(V)
. mat V1a = V1[1..3,1..3]
. mat V1b = V1[5..7,1..3]
. mat V1c = V1[5..7,5..7]
. mat V1 = (V1a, V1b´) \ (V1b, V1c)
```

```
. qui avar (e1 e2) (x1 x2), rob cluster(id2)
. mat S=r(S)
. local cn : colfullnames S
. mat KSxxi= I(2)#Sxxi
. mat V2 = KSxxi*S*KSxxi*1/r(N)
. mat V2 = V2 * e(N_clust)/(e(N_clust)-1)
. mat colnames V2=`cn´
. mat rownames V2=`cn´
. assert mreldif(V1,V2) < 1e-7
```

# avar as a building block

Users can employ `avar` as a building block for their own estimation routines, whether these estimation routines are intended for a wide audience or are for their own use.

As an example of the former, `weakiv` by Finlay, Magnusson and Schaffer is an estimation routine for weak-instrument-robust inference, an alternative to standard IV/GMM estimation. This command, available from SSC, can provide estimates for all the non-i.i.d. options supported by `avar`.

# avar as a building block

Users can employ `avar` as a building block for their own estimation routines, whether these estimation routines are intended for a wide audience or are for their own use.

As an example of the former, `weakiv` by Finlay, Magnusson and Schaffer is an estimation routine for weak-instrument-robust inference, an alternative to standard IV/GMM estimation. This command, available from SSC, can provide estimates for all the non-i.i.d. options supported by `avar`.

`weakivtest` by Montiel Olea, Pflueger and Wang is a postestimation routine for testing for the presence of weak instruments following IV or LIML estimation. Their innovation is that `weakivtest` calculates the Montiel–Pflueger test (*J.Bus.Ec.Stat.*, 2013), which unlike existing tests can be made robust to various non-i.i.d. alternatives. `weakivtest` (which the authors wrote independently) uses `avar` to construct the appropriate robust $S$, and is available from SSC.

The rest of the presentation is an example of the latter: a user-written estimation routine that is designed for a specific problem, GMM–CUE, and that uses `avar` to construct an appropriate robust $S$.

`weakivtest` by Montiel Olea, Pflueger and Wang is a postestimation routine for testing for the presence of weak instruments following IV or LIML estimation. Their innovation is that `weakivtest` calculates the Montiel–Pflueger test (*J.Bus.Ec.Stat.*, 2013), which unlike existing tests can be made robust to various non-i.i.d. alternatives. `weakivtest` (which the authors wrote independently) uses `avar` to construct the appropriate robust $S$, and is available from SSC.

The rest of the presentation is an example of the latter: a user-written estimation routine that is designed for a specific problem, GMM–CUE, and that uses `avar` to construct an appropriate robust $S$.

# Application: user-specified GMM-CUE

The features of `avar` are likely to be most useful when it is used as a building block for other estimation routines. As an illustration, we have constructed a routine to implement a general-purpose GMM-CUE (Generalized Method of Moments/Continously Updated Estimator).

This routine, `gmmcue.ado`, requires that the user provide a Mata function, `m_gbar()`, which takes a single argument: a Mata structure. The function must compute the residuals for each observation, analogous to the way in which a function evaluator for Stata's `gmm` command works, and computes a matrix `gbar` which contains the sample average values $(1/N)Z'e$.

# Application: user-specified GMM-CUE

The features of `avar` are likely to be most useful when it is used as a building block for other estimation routines. As an illustration, we have constructed a routine to implement a general-purpose GMM-CUE (Generalized Method of Moments/Continously Updated Estimator).

This routine, `gmmcue.ado`, requires that the user provide a Mata function, `m_gbar()`, which takes a single argument: a Mata structure. The function must compute the residuals for each observation, analogous to the way in which a function evaluator for Stata's `gmm` command works, and computes a matrix `gbar` which contains the sample average values $(1/N)Z'e$.

# Syntax

`gmmcue (`*yvarlist*`)` [ *weight* ] [ *if* ] [ *in* ] [, `initb(`*numlist*`)` `initbmat(`*string*`)` *xvars*`(`*varlist*`)` *zvars*`(`*varlist*`)` `vceopt(`*string*`)` ]

where the *yvarlist* may contain one or more dependent variables (or be empty, as we will show); the `xvarlist` may contain regressors; and the `zvarlist` may contain instruments. The `vceopt` choices are those available in `avar`. Initial parameter values may be provided in `initb()` or as a matrix from a prior consistent estimation.

As an illustration of the user-written Mata function for standard regression problems:

```
. mata:
------------------------------------------------- mata (type end to exit) --------
: real matrix m_gbar(struct ms_cuestruct scalar cuestruct)
> {
>
> // ********* BEGIN USER-DEFINED GBAR SECTION ********** //
>
> // Calculate residuals e
>         (*cuestruct.e)[.,.] = *cuestruct.Y - *cuestruct.X * *cuestruct.b'
> // Calculate average moments gbar
>         gbar = 1/cuestruct.N * quadcross(*cuestruct.Z, *cuestruct.e)
>
> // ********** END USER-DEFINED GBAR SECTION *********** //
>
>         return(gbar)
> }
: end
----------------------------------------------------------------------------------
```

We may then call `gmmcue`, which in turn calls `avar`, in a do-file which defines this function:

```
. sysuse auto, clear
(1978 Automobile Data)
. gen byte one=1
. gmmcue price, xvars(mpg one) zvars(weight turn trunk one) initb(1 1) nolog
GMM-CUE estimates                                          Number of obs = 74
vceopt(.) =
```

| | Coef. | Std. Err. | z | P>\|z\| | [95% Conf. | Interval] |
|---|---|---|---|---|---|---|
| b1 | -346.8264 | 66.43345 | -5.22 | 0.000 | -477.0336 | -216.6192 |
| b2 | 13551.72 | 1448.28 | 9.36 | 0.000 | 10713.14 | 16390.3 |

```
Sargan-Hansen J statistic:  10.429
Chi-sq(  2  )        P-val = 0.0054
```

By merely specifying options, we may estimate using different assumptions on the VCE:

```
. gmmcue price, xvars(mpg one) zvars(weight turn trunk one) initb(1 1) ///
> vceopt(rob) nolog
GMM-CUE estimates                                          Number of obs = 74
vceopt(.) = rob
------------------------------------------------------------------------------
             |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
          b1 |  -266.6691   61.93988    -4.31   0.000     -388.069   -145.2692
          b2 |   11439.57   1459.829     7.84   0.000     8578.361    14300.79
------------------------------------------------------------------------------
Sargan-Hansen J statistic:   10.667
Chi-sq(  2  )        P-val = 0.0048
```

```
. g clustr = mod(_n,10)

. gmmcue price, xvars(mpg one) zvars(weight turn trunk one) initb(1 1) ///
> vceopt(cluster(clustr)) nolog

GMM-CUE estimates                                      Number of obs = 74
vceopt(.) = cluster(clustr)
------------------------------------------------------------------------------
             |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
          b1 |  -306.9181   23.67029   -12.97   0.000     -353.311   -260.5252
          b2 |    12269.8    606.978    20.21   0.000     11080.14    13459.45
------------------------------------------------------------------------------

Sargan-Hansen J statistic:   6.234
Chi-sq(  2  )        P-val = 0.0443
```

# Illustration with two-way clustering in panel data:

```
. webuse grunfeld, clear
. gen byte one=1
. qui ivreg2 invest (kstock = mvalue year), gmm2s robust
. mat b = e(b)
. gmmcue invest, xvars(kstock one) zvars(mvalue year one) initbmat(b) ///
> vceopt(cluster(company year))
Iteration 0:   f(p) =  5.0023632  (not concave)
Iteration 1:   f(p) =  4.2956314  (not concave)
Iteration 2:   f(p) =   4.278281  (not concave)
Iteration 3:   f(p) =  4.2776094
Iteration 4:   f(p) =  4.2775708
Iteration 5:   f(p) =  4.2775634
Iteration 6:   f(p) =  4.2775629
GMM-CUE estimates                                    Number of obs = 200
vceopt(.) = cluster(company year)
------------------------------------------------------------------------------
             |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
          b1 |   .7338768   .1234618     5.94   0.000      .491896    .9758575
          b2 |  -18.85657   38.08971    -0.50   0.621    -93.51103    55.79789
------------------------------------------------------------------------------
Sargan-Hansen J statistic:   4.278
Chi-sq(  1  )        P-val = 0.0386
```

## Illustration with Newey–West HAC VCE in a panel context:

```
. gmmcue invest, xvars(kstock one) zvars(mvalue year one) initbmat(b) ///
> vceopt(bw(5) kernel(bartlett)) nolog
GMM-CUE estimates                                       Number of obs = 200
vceopt(.) = bw(5) kernel(bartlett)
------------------------------------------------------------------------------
             |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
          b1 |   1.038736   .1454092     7.14   0.000     .7537396    1.323733
          b2 |  -141.5256   51.34809    -2.76   0.006     -242.166   -40.88518
------------------------------------------------------------------------------
Sargan-Hansen J statistic:  15.719
Chi-sq(  1  )        P-val = 0.0001
```

## Illustration with time series operators in a panel context:

```
. gmmcue F.invest, xvars(D.kstock one) zvars(L(0/2).mvalue year one)  ///
> initbmat(b) vceopt(robust) nolog
GMM-CUE estimates                                    Number of obs = 170
vceopt(.) = robust
------------------------------------------------------------------------------
             |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
          b1 |   6.838157   1.224357     5.59   0.000     4.438461    9.237854
          b2 |  -44.87452   24.56177    -1.83   0.068    -93.01471    3.265674
------------------------------------------------------------------------------
Sargan-Hansen J statistic:   5.386
Chi-sq(  3  )         P-val = 0.1456
```

`gmmcue` can handle not only standard regression problems, but other optimization problems which may be solved in the GMM framework. In the Stata manual description of `gmm`, Example 13 shows how an Euler equation may be fit to consumption, following Hansen and Singleton (*Econometrica*, 1982). Using a constant relative risk aversion (CRRA) utility function, the Euler equation is

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

where $r_t$ is the interest rate (the return on three-month Treasury bills) and $c_t$ is aggregate consumption expenditures. In this model, $\beta$ is the discount factor; it is near 1, the agent is patient, willing to forgo current consumption. If it is near zero, the agent prefers to consume more now. The parameter $\gamma$ characterizes the utility function; if it is zero, the function is linear, whereas as it nears 1, utility is a function of the log of consumption.

# The user-written Mata function for this model may be written as:

```
. mata:
------------------------------------------------- mata (type end to exit) ------
: real matrix m_gbar(struct ms_cuestruct scalar cuestruct)
> {
>
> // ********* BEGIN USER-DEFINED GBAR SECTION ********** //
>
>         A = 1 :+ (*cuestruct.X)[.,1]
>         B = (*cuestruct.b)[1,1] * A
>         C = ((*cuestruct.X)[.,2]):^(-(*cuestruct.b)[1,2])
>         D = 1 :- B :* C
>         (*cuestruct.e)[.,.] = D
> // Calculate average moments gbar
>         gbar = 1/cuestruct.N * quadcross(*cuestruct.Z, *cuestruct.e)
>
>         // ********** END USER-DEFINED GBAR SECTION *********** //
>
>         return(gbar)
> }
:
: end
--------------------------------------------------------------------------------
```

We first estimate the model with `gmm` to get reasonable starting values, using as instruments `L.r L2.r cgrowth L.cgrowth`, where `r` is the interest rate and `cgrowth` is one-period consumption growth:

```
. use http://www.stata-press.com/data/r12/cr, clear
. generate cgrowth = c / L.c
(1 missing value generated)
. gen byte one=1
. qui gmm (1 - {b=1}*(1+F.r)*(F.cgrowth)^(-1*{gamma=1})),          ///
>         inst(L.r L2.r cgrowth L.cgrowth) wmat(hac nw 4) twostep
. estat overid
  Test of overidentifying restriction:
  Hansen´s J chi2(3) = 14.2532 (p = 0.0026)
. mat b_2step=e(b)
```

We then invoke `gmmcue` to estimate $(\beta, \gamma)$, using a AC estimator with bandwidth=5, corresponding to four lags:

```
. gmmcue, xvars(F.r F.cgrowth) zvars(L.r L2.r cgrowth L.cgrowth one)      ///
>         vceopt(bw(5) kernel(bartlett)) initb(1 1) nolog
GMM-CUE estimates                                       Number of obs = 239
vceopt(.) = bw(5) kernel(bartlett)
------------------------------------------------------------------------------
             |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
          b1 |   1.010785   .0128247    78.82   0.000      .985649    1.035921
          b2 |  -.1190646   1.034646    -0.12   0.908    -2.146934    1.908805
------------------------------------------------------------------------------
Sargan-Hansen J statistic:  48.259
Chi-sq(  3  )         P-val = 0.0000
```

In these estimates, the agent appears to be patient, and linearity of the utility function cannot be rejected.

# GMM-CUE vs. iterated GMM

Although it is still under development, our `gmmcue` routine adds a significant capability to Stata. GMM-CUE is not the same as the "iterated GMM" implemented by Stata's `gmm` command, as the latter does not impose the same constraints in defining the optimum. For instance, in the *i.i.d.* case, GMM-CUE reduces to LIML, but iterating the GMM estimator to convergence does not produce the LIML estimates.

Steve Bond argues that “However there is no (asymptotic) efficiency gain in using the iterated GMM estimator compared to using the two-step GMM estimator, and no clear evidence that iterated GMM has better small sample properties.” (lecture notes, Nuffield College)

In contrast, Newey and Smith (*Econometrica*, 2004) and Anatolyev (*Econometrica*, 2005) claim that although iterated GMM is asymptotically equivalent to GMM-CUE, the latter has a smaller second order asymptotic bias.

Steve Bond argues that "However there is no (asymptotic) efficiency gain in using the iterated GMM estimator compared to using the two-step GMM estimator, and no clear evidence that iterated GMM has better small sample properties." (lecture notes, Nuffield College)

In contrast, Newey and Smith (*Econometrica*, 2004) and Anatolyev (*Econometrica*, 2005) claim that although iterated GMM is asymptotically equivalent to GMM-CUE, the latter has a smaller second order asymptotic bias.

# Conclusions

In summary, `avar` can play a useful role in producing a number of VCE estimates based upon differing assumptions of the underlying error processes, in both a single-equation and multiple-equation framework. Scrutiny of the code illustrates the flexibility and usefulness of Mata's structures and pointer data type. The `gmmcue` routine illustrates how `avar` may be used in the context of a general-purpose GMM routine, and adds a useful tool to Stata's GMM capabilities.