The Stata Journal (2003)
3, Number 4, pp. 351–360

Instrumental variables, bootstrapping, and generalized linear models

James W. Hardin
Arnold School of Public Health
University of South Carolina
Columbia, SC 29208

Henrik Schmiediche
Department of Statistics MS-3143
Texas A&M University
College Station, TX 77843-3143

Raymond J. Carroll
Department of Statistics MS-3143
Texas A&M University
College Station, TX 77843-3143

Abstract. This paper discusses and illustrates the `qvf` command for fitting generalized linear models. The differences between this new command and Stata's `glm` command are highlighted. One of the most notable features of the `qvf` command is its ability to include instrumental variables. This functionality was added specifically to address measurement error but may be utilized by the user for other purposes. The `qvf` command was developed in the C-language using Stata's new plugin features and executes much faster than the `glm` ado-file.

Keywords: st0049, measurement error, instrumental variables, Murphy–Topel, bootstrap, generalized linear models

1 Introduction

This paper describes software for analyzing measurement error models. Software production by StataCorp was funded by a National Institutes of Health (NIH) Small Business Innovation Research Grant (SBIR). The goal of the work described in the grant is the production of software to analyze statistical models where one or more covariates are measured with error. The software development included two major features. The first development feature is the development of the Stata program to support communication to dynamically linked user-written computer code. StataCorp was responsible for this development and support for user-written code in the C/C++ programming languages was added to Stata version 8. Stata refers to compiled user-written code as plugins and maintains documentation on their web site at *http://www.stata.com/support/plugins*.

Following the presentation in Carroll, Ruppert, and Stefanski (1995), we will discuss the instrumental variables method for handling additive measurement error in generalized linear models (GLMs); see Hardin and Hilbe (2001) for detailed information on GLMs. The associated software may be used whether or not there is measurement error in a particular analysis.

The project described was supported by Grant Number R44 RR12435 from the National Institutes of Health, National Center for Research Resources. The contents of this article are solely the responsibility of the authors and do not necessarily represent the official views of the National Center for Research Resources.

Utilizing the short introduction in Hardin and Carroll (2003), we show the usage of the new `qvf` command. In many cases, this new command will produce identical results to the `glm` command.

2 Comparison to the existing glm command

The subsections that follow compare the existing `glm` command with the new `qvf` command. Both of these command can use the observed or expected information matrix during optimization and to form default standard errors. To use the expected information matrix in `glm` (specified in the iteratively reweighted least squares algorithm); the user specifies the `irls` option with the `glm` command. To use the observed information matrix, the user specifies the `oim` option with the `qvf` command.

2.1 Missing features

There are a few features of the `glm` command that are not included in the new `qvf` command. The most notable difference is that the `qvf` command does not support the `predict` command. The genesis of the new command was to support measurement error analysis, and in most cases the first step of the analysis is the generation of expected values for a missing covariate. This step would be required for prediction and is not available for any of the commands included in the SBIR development.

The `glm` command also produces model diagnostics that are not present in output from the `qvf` command. These diagnostics include calculation of Akaike's information criterion, the Bayes information criterion, and log likelihoods.

2.2 Additions and enhancements

There are several additions and enhancements over what the `glm` command will provide.

The new command has built-in support for bootstrapping. Since the support was written into the C-language code development, model fitting for the bootstrap samples is very fast.

The `qvf` command has built-in support for instrumental variables. This is actually the *raison d'être* for `qvf`. The purpose of writing a new command when one already existed was a desire to make the fast GLM we created for `simex` available for general purpose use and out of the recognition that we needed to support instrumental variables to support the measurement error analyses; these analyses are, in fact, the motivation for the SBIR project.

The `qvf` command also supports calculation of Murphy–Topel type standard errors, see Murphy and Topel (1985) and Hardin (2002), when fitting generalized linear models with instrumental variables.

2.3 Equivalent model specifications

The `qvf` and `glm` commands produce equivalent results for model fits where there are no instrumental variables. There may be numeric differences in results due to different convergence criteria.

Both commands support the calculation of standard errors based on the expected information matrix, observed information matrix, robust variance matrix, semi-robust variance matrix, and the cluster variants of the robust matrix calculations. Jackknife variance estimates are only allowed with the `glm` command, and Murphy–Topel variance estimates are allowed only with the `qvf` command (only when instrumental variables are specified).

3 QVF syntax and options

In the subsections that follow, we will show the three main features of `qvf`. First is that `qvf` is similar to `glm`. We will demonstrate the fast internal `qvf` bootstrap. Finally, we will show how to use `qvf` to analyze data with instrumental variables.

3.1 QVF as GLM

The `qvf` and `glm` commands use similar syntax and display similar output. Sometimes it is possible to simply substitute `qvf` for `glm`. For example,

```
. use http://www.stata-press.com/data/r8/lbw
(Hosmer & Lemeshow data)
. qvf low age lwt race smoke ptl ht ui, family(bin) link(logit)
Generalized linear models                         No. of obs      =       189
Optimization     : MQL Fisher scoring             Residual df     =       181
                   (IRLS EIM)                     Scale param     =         1
Deviance         =   204.347465                   (1/df) Deviance =  1.128992
Pearson          =   180.583157                   (1/df) Pearson  =   .997697
Variance Function: V(u) = u(1-u)                  [Bernoulli]
Link Function    : g(u) = log(u/(1-u))            [Logit]
Standard Errors  : EIM Hessian
```

low	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
age	-.0332327	.0357713	-0.93	0.353	-.1033431	.0368777
lwt	-.0120948	.0066158	-1.83	0.068	-.0250617	.000872
race	.4462621	.2150445	2.08	0.038	.0247827	.8677415
smoke	.9255414	.3980923	2.32	0.020	.1452947	1.705788
ptl	.5397383	.3469187	1.56	0.120	-.1402099	1.219686
ht	1.799337	.6872194	2.62	0.009	.4524118	3.146262
ui	.7148045	.4634311	1.54	0.123	-.1935037	1.623113
_cons	-.1029665	1.284609	-0.08	0.936	-2.620754	2.414821

Replacing `qvf` with `glm`

```
glm low age lwt race smoke ptl ht ui, family(bin) link(logit)
```

produces identical results. Note that `qvf` uses iterated, reweighted least squares (IRLS) optimization, while `glm` uses Newton–Raphson optimization of the log likelihood. This difference could be adjusted for by adding the `irls` option to the `glm` command.

The options that are recognized by both the `qvf` and `glm` commands are `family`, `link`, `ltolerance`, `iterate`, `[ln]offset`, `level`, `oim`, `robust`, `cluster`, `scale`, and `vfactor`. Options specific to `qvf` are `mtopel`, which calculates Murphy–Topel variance estimator (available only for instrumental variables models), and `mess`, which controls the messages the `qvf` plugin command displays. Finally, `qvf` uses the options `bstrap`, `brep()`, `btrim()`, `seed()`, `saving()`, and `replace` to control the fast internal bootstrap.

3.2 The fast bootstrap

The `qvf` command can calculate the bootstrap estimator of variance using a fast internal bootstrap. To demonstrate the bootstrap, we will use this simulated dataset:

```
. set seed 1
. set obs 2500
obs was 0, now 2500
. gen x1  = uniform()
. gen x2  = uniform()
. gen x3  = uniform()
. gen x4  = uniform()
. gen err = invnorm(uniform())
. gen y = 5 + 1*x1 + 2*x2 + 3*x3 + 4*x4 + err
```

To calculate a 999 replicate bootstrap variance estimator using `qvf`, we type

```
. qvf y x1 x2 x3 x4, bstrap brep(999) btrim(.05) seed(1)
Generalized linear models                         No. of obs      =      2500
Optimization     : MQL Fisher scoring             Residual df     =      2495
                   (IRLS EIM)                     Scale param     =  1.040753
Deviance         =  2596.677816                   (1/df) Deviance =  1.040753
Pearson          =  2596.678735                   (1/df) Pearson  =  1.040753
Variance Function: V(u) = 1                       [Gaussian]
Link Function    : g(u) = u                       [Identity]
Standard Errors  : Bootstrap
```

y	Coef.	Bootstrap Std. Err.	z	P>|z|	[95% Conf.	Interval]
x1	1.000742	.0683909	14.63	0.000	.8666982	1.134786
x2	1.996032	.0699536	28.53	0.000	1.858925	2.133139
x3	3.052939	.0683828	44.64	0.000	2.918911	3.186967
x4	3.967281	.0683857	58.01	0.000	3.833247	4.101314
_cons	4.998845	.0698154	71.60	0.000	4.862009	5.13568

On a 3.2GHz Pentium 4, this will take less than a second. The only required option, `bstrap`, tells `qvf` to calculate the bootstrap estimator of variance. The `brep` option indicates the number of replications (default is 199), and the `btrim` option causes `qvf` to trim $2 \times 2.5\%$ of the bootstrap replication prior to the calculation of the bootstrap variances. The `seed` option initializes the random number generator used by the bootstrap. The only reason to specify a seed is to generate identical results over multiple runs; this is rarely used. Note that the Stata `set seed` command does not affect the internal random number generator used by the `qvf` plugin.

The equivalent Stata `bootstrap` command using glm (with no trimming) would be

```
. bootstrap "glm y x1 x2 x3 x4" _b, reps(999)
```

which requires about 55 seconds on the same system. You can use the `qvf` command with the Stata `bootstrap` command if you require some of the additional functionality that the Stata `bootstrap` command provides

```
. bootstrap "qvf y x1 x2 x3 x4" _b, reps(999)
```

The above requires about 23 seconds. We can observe from these times that about 60% the speed improvement in our internal bootstrap comes from the fast GLM and 40% from internalizing the bootstrap. The more complicated or larger the model or dataset, the more this ratio will favor the fast GLM.

The `qvf` will calculate the estimated time it will take to produce a result. If `qvf` determines it will take more than 30 seconds, a *time to completion* estimate is printed. This estimate is fairly accurate if the workload on the computer during the initial few seconds of the bootstrap is representative of conditions during the entire bootstrap. Even if workload conditions change dramatically, no further estimate is printed.

Bootstrap replicates and confidence intervals

As with the Stata `bootstrap` command, the confidence intervals displayed by `qvf` are based on the variance matrix and are not calculated based on percentiles of the bootstrap replicates. The `qvf saving()` and `replace` options can be used to save the individual bootstrap replicates to a file and to perform further analysis on the bootstrap replicates.

The `saving()` option will save each bootstrap replicate, comma separated, in the same order they are displayed. The iteration count for that replicate is added as well. So, for our example, the stored data are listed in the following order: `x1`, `x2`, `x3`, `x4`, and `cons`. Note that the `saving()` option will save all bootstrap replicates without regard to any trimming percentages.

For example, the mean bootstrap parameter estimate is sometimes desired:

```
. qui qvf y x1 x2 x3 x4, bstrap brep(999) seed(1) saving("bootrep.txt") replace
. infile x1 x2 x3 x4 cons _skip(1) using bootrep.txt, clear
(999 observations read)
```

```
. summarize
    Variable |       Obs        Mean    Std. Dev.       Min        Max
-------------+--------------------------------------------------------
          x1 |       999    .9982838    .0702145    .765967   1.265923
          x2 |       999    1.997864    .0716246   1.785184    2.28865
          x3 |       999    3.054167    .0702604   2.844582   3.263436
          x4 |       999     3.97179    .0709933   3.751332   4.165956
        cons |       999    4.994867    .0724652   4.744656   5.238493
```

To display median bootstrap parameter estimates and confidence intervals based on standard percentiles, simply add the `detail` option to the `summarize` command above. To calculate a specific percentile based confidence interval, try

```
. _pctile x1, p(2.5, 50, 97.5)
. return list
scalars:
                r(r1) =  .8637648224830627
                r(r2) =  1.000898361206055
                r(r3) =  1.135726928710938
```

which displays the median bootstrap replicate for parameter x_1 and its 95% percentile based confidence interval: $[.8638, 1.1357]$.

3.3 Instrumental variables

To use `qvf` to fit GLM-type models with instrumental variables, we use the following syntax:

`qvf` *depvar varlist*$_1$ *varlist*$_2$ `(`*varlist*$_1$ *varlist*$_3$`)` $[\ldots]$ $[$`,` *options*$]$

where

varlist$_1$**:** the exogenous variables for which we have no instruments.

varlist$_2$**:** the endogenous variables for which we have instruments *varlist*$_3$.

varlist$_3$**:** the exogenous instrumental variables for the *varlist*$_2$ variables.

This syntax will fit a GLM-type model using instrumental variables of *depvar* on *varlist*$_1$ and *varlist*$_2$ using *varlist*$_3$ (along with *varlist*$_1$) as instruments for *varlist*$_2$.

Three estimators of variance are available when using instrumental variables:

`robust`: the Huber/White/sandwich estimator of variance is the default for `qvf` when instrumental variables are used.

`mtopel`: the Murphy–Topel variance estimator. This estimator is only available when instrumental variables are present.

`bstrap`: the bootstrap estimator of variance.

To demonstrate qvf with instrumental variables, we will use the simulated dataset we created in section 3.2. First, we generate two instruments that are correlated $\rho = .8$ with the endogenous variables x_3 and x_4:

```
. gen t1 = .8*x3 + .6*invnorm(uniform())
. gen t2 = .8*x4 + .6*invnorm(uniform())
```

The analysis itself is

```
. qvf y x1 x2 x3 x4 (x1 x2 t1 t2)
IV Generalized linear models                    No. of obs      =      2500
Optimization     : MQL Fisher scoring           Residual df     =      2495
                   (IRLS EIM)                   Scale param     =  2.879088
Deviance         =  7183.324663                 (1/df) Deviance =  2.879088
Pearson          =   7183.32456                 (1/df) Pearson  =  2.879088
Variance Function: V(u) = 1                     [Gaussian]
Link Function    : g(u) = u                     [Identity]
Standard Errors  : OIM Sandwich
```

y	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
x1	1.003203	.0707847	14.17	0.000	.8644672	1.141938
x2	1.995794	.0731505	27.28	0.000	1.852421	2.139166
x3	3.137879	.212108	14.79	0.000	2.722155	3.553603
x4	4.09093	.1969634	20.77	0.000	3.704889	4.476971
_cons	4.892393	.1614647	30.30	0.000	4.575928	5.208858

As expected, the standard errors for x_3 and x_4 are larger then when no instrumental variables are present. The equivalent ivreg syntax would be

```
. ivreg y x1 x2 (x3 x4 = t1 t2), robust
```

Note the slight difference in syntax. The qvf command follows the syntax of the Stata regress command for a two-stage least-squares analysis.

4 Formal Stata syntax

qvf *depvar* [*varlist*] [(*varlist*)] [*weight*] [if *exp*] [in *range*] [message(#)
family(*string*) link(*string*) ltolerance(#) iterate(#)
[ln]offset(*varname*) mtopel level(#) oim robust cluster(*varname*)
scale(*string*) vfactor(#) bstrap brep(#) btrim(#) seed(#)
saving(*string*) replace]

General options

message(#) specifies the desired (observed) level of printed messages of the plugin module. Users can use this option to suppress or request warning and informational messages.

0) Display nothing, not even fatal error messages.

1) Display fatal error messages only.

2) Display warning messages (default).

3) Display informational messages.

4) Display more informational messages.

Note that the ado-code that handles the I/O to the plugin may still print error messages regardless of the message level setting.

The message command can also be used to see intermediate details of the internal calculations of the code. These were used by the authors to debug the code. The notation and mnemonics used are not documented and may not correspond to anything in the printed documentation. Furthermore, the numbers may be in a raw and unadjusted format that is difficult to interpret.

5, 6, 7) Display details with increasing verbosity.

Message levels are cumulative.

`family(`*string*`)` specifies the distribution of the dependent variable. The `gaussian` family is the default. The choices and valid family and link combinations are the same as for Stata's `glm` command.

`link(`*string*`)` specifies the link function. The default is the canonical link for the `family()` specified. The choices and valid family and link combinations are the same as for Stata's `glm` command.

`ltolerance(#)` specifies the convergence criterion for the change in deviance between iterations. The default is $1e-6$.

`iterate(#)` specifies the maximum number of iterations allowed in fitting the model; The default is 100. It is rare that one needs to increase this.

[`ln`]`offset(`*varname*`)` specifies an offset to be added to the linear predictor. See [R] **glm** for more information.

Standard error options

`mtopel` specifies that the Murphy–Topel variance estimator should be used. This option is valid for instrumental variables models.

`level(#)` specifies the confidence level, in percent, for confidence intervals of the coefficients.

`oim` specifies that the variance matrix should be calculated using the observed information matrix (OIM) rather than the usual expected information matrix (EIM).

`robust` specifies that the Huber/White/sandwich estimator of variance is to be used in place of the traditional calculation. This is the default for instrumental variables. `robust` combined with `cluster()` allows observations which are not independent within cluster (although they must be independent between clusters).

`cluster(`*varname*`)` specifies that the observations are independent across groups (clusters). See [R] **glm** for more information.

`scale(x2|dev|#)` overrides the default scale parameter. See [R] **glm** for more information.

`vfactor(#)` specifies a scalar by which to multiply the resulting variance matrix. See [R] **glm** for more information.

Bootstrap options

`bstrap` specifies that bootstrap standard errors should be calculated. The bootstrap is internal to the code for the regression calibration command. The estimated time to perform the bootstrap will be displayed should the bootstrap require more than 30 seconds.

`brep(#)` specifies the number of bootstrap samples generated to calculate the bootstrap standard errors of the fitted coefficients. The default is `brep(199)`.

`btrim(#)` specifies the amount of trimming applied to the collection of bootstrap samples prior to calculation of the bootstrap standard errors. The default is `btrim(.02)`, meaning that 1% of the samples (rounded) will be trimmed at each end. Trimming has no effect on the parameter estimates.

When the bootstrap is run with `mess(3)`, an informational message similar to this one will display,

```
Average number of iterations per   GLM call: 3.6
Maximum number of iterations for a GLM call: 5
Minimum number of iterations for a GLM call: 3
Trimming total of 4 bootstrap replications (2.0%).
Maximum change in standard errors due to trimming: 2.4%
```

indicating that 4 samples (2 on each end) were trimmed and that this trimming resulted in a 2.4% change in magnitude of one of the standard errors. All other standard errors changed less than 2.4%. This simple diagnostic gives an indication of how trimming influenced the bootstrap standard errors.

`seed(#)` specifies a random number seed used in generating random samples for the bootstrap calculations. This option has no effect if bootstrapping is not specified. Its main purpose is to allow repeatability of bootstrap results. The default is `seed(0)`, which will seed the random number generator using the system clock.

`saving(`*string*`)` specifies the file to which the bootstrap replicates will be saved. The `saving()` option will save each bootstrap replicate, comma separated, in the same order they are displayed. The iteration count for that replicate is added as well. Note that the `saving()` option will save all bootstrap replicates without regard to any trimming percentages.

`replace` replaces the file specified in the `saving()` option (if that file already exists).

5 References

Carroll, R. J., D. Ruppert, and L. A. Stefanski. 1995. *Measurement Error in Nonlinear Models*. London: Chapman & Hall.

Hardin, J. and J. Hilbe. 2001. *Generalized Linear Models and Extensions*. College Station, TX: Stata Press.

Hardin, J. W. 2002. The robust variance estimator for two-stage models. *Stata Journal* 2(3): 253–265.

Hardin, J. W. and R. J. Carroll. 2003. Measurement error, GLMs, and notational conventions. *Stata Journal* 3(4): 328–340.

Murphy, K. M. and R. H. Topel. 1985. Estimation and inference in two-step econometric models. *Journal of Business and Economic Statistics* 3(4): 370–379.

About the Authors

James W. Hardin (jhardin@gwm.sc.edu), is an Associate Research Professor, Department of Epidemiology and Biostatistics, and a Research Scientist, Center for Health Services and Policy Research, Arnold School of Public Health, Carolina Plaza Suite 1120, University of South Carolina, Columbia, SC 29208, USA.

Henrik Schmiediche (henrik@stat.tamu.edu), is a Senior Lecturer and Senior Systems Analyst, Department of Statistics, MS 3143, Texas A&M University, College Station, TX 77843-3143, USA.

Raymond J. Carroll (carroll@stat.tamu.edu) is a Distinguished Professor, Department of Statistics, MS 3143, Texas A&M University, College Station, TX 77843-3143, USA.

Research by StataCorp was supported by National Institutes of Health (NIH) Small Business Innovation Research Grant (SBIR) (2R44RR12435-02).