Implementing Quantile Selection Models in Stata[*]

Ercio Muñoz[†] Mariel Siravegna[‡]

April 21, 2020

Abstract

This article describes **qregsel**, a Stata module to implement a copula-based sample selection correction for quantile regression recently proposed by Arellano and Bonhomme (2017, Econometrica 85(1): 1-28). This user-written command exploits the newly available Stata 16 capabilities to solve linear programming problems, and the integration with Python. We illustrate the use of **qregsel** with an empirical example using the data employed in the Stata base reference manual for the `heckman` command.

Keywords: *sample selection, quantile regression, copula method, python*

[*]We thank Jim Albrecht and Wim Vijverberg for useful comments and suggestions.
[†]CUNY Graduate Center; email: emunozsaavedra@gc.cuny.edu.
[‡]Georgetown University; email: mcs92@georgetown.edu.

1 Introduction

Non-random sample selection is a well known issue in empirical economics. Since the seminal work of Heckman (1979), much progress has been made in methods that extend the original model or relax some of its assumptions. For example, Vella (1998) provides a survey of methods for estimating models with sample selection bias in this line.

Although most of the effort has been focused on models that estimate the conditional mean, the literature in econometrics has also tackled the problem of non-random sample selection in the context of quantile regression. For example, Arellano and Bonhomme (2017a) offer a survey of recently proposed methods with a focus on a copula-based sample selection model suggested in Arellano and Bonhomme (2017b).

As discussed in Arellano and Bonhomme (2017a), the flexible copula-based approach has an advantage over methodologies that are based on the control function approach. The latter impose conditions on the data that may not be compatible with quantile models if the model is non-additive with non-linear quantile curves on the selected sample (see Huber and Melly (2015)).

In this paper, we briefly discuss the copula-based approach proposed by Arellano and Bonhomme (2017b) and present a new Stata module called `qregsel` that implements it.[1] This user-written command exploits Stata 16 new capabilities to solve linear programming problems, and the integration with Python as an option to achieve faster execution. In addition, we illustrate the method with an empirical exercise in which we estimate a quantile regression model with sample selection using the Stata base reference manual example for the `heckman` command.

The paper is organized as follows. Section 2 describes the methodology. Section 3 describes the `qregsel` command and its syntax. In section 4 we illustrate the use of the command with the empirical example, and we conclude in Section 5.

2 Methodology

In this section we briefly review the quantile selection model of Arellano and Bonhomme (2017b). The goal is to obtain a consistent estimator when there is sample selection in a non-additive model, which precludes the use of the control function approach. The assumption of additivity does not hold in general, as argued by Huber and Melly (2015) in the context of testing.

2.1 The Model

Sample selection is modeled using a bivariate cumulative distribution function or copula of the percentile error in the latent outcome equation and the error in the sample selection equation. The copula parameters are estimated by minimizing a method-of-

1. A copula-based maximum-likelihood method for the conditional mean is already available in Stata (see Hasebe (2013)).

moments criterion that exploits variation in excluded regressors to achieve identification. Then the quantile regression parameters are obtained by minimizing a rotated check function, which preserves the linear programming structure of the standard linear quantile regression (see Koenker and Bassett (1978)).

Consider a general outcome equation specification where the quantile functions are linear:

$$Y^* = Q(U, X) = x'\beta(\tau) \tag{1}$$

where Y^* is the the latent outcome variable (e.g. wage offers), the function Q is the τ-th conditional quantile of Y^* given the covariates X (e.g. education, experience, etc.), and U is the error term of the outcome equation.

The participation equation is defined as:

$$D = I\{V \leq p(Z)\} \tag{2}$$

where D takes values equal to 1 when the latent variable is observable (e.g. employment) and 0 otherwise, Z contains X and at least one covariate B that do not appear in the outcome equation (e.g., a determinant of employment that does not affect wages directly), *p(Z)* is a propensity score, and V is an error term of the selection equation.

Define G_x as the conditional copula function, which measures the dependence between U and V as:

$$G_x(\tau, p) \equiv G(\tau, p; \rho) = \frac{C(\tau, p; \rho)}{p} \tag{3}$$

where the numerator is the unconditional copula of *(U,V)*, the denominator is the propensity score, and ρ is the copula parameter that governs the dependence between the error in the outcome equation and the error in the participation decision.[2] G_x maps rank τ in the distribution of latent outcomes (given X=x) to ranks $G_x(\tau, p(Z))$ in the distribution of observed outcomes conditional on participation (given Z=z). Namely, the conditional $G_x(\tau, p(z))$-quantile of observed outcome (that is when $D = 1$) coincides with the conditional τ-quantile of latent outcome, and this is true for each $\tau \in (0, 1)$. The key implication from this equation is that, if we are able to estimate the mapping $G_x(\tau, p)$ from latent to observed ranks, we are able to estimate the quantiles of observed outcomes corrected for selection.

2.2 Estimation

Given all the above, Arellano and Bonhomme (2017a)'s estimation algorithm can be summarized in 3 steps: estimation of the propensity score, estimation of the degree of selection via the cumulative distribution function of the percentile error in the outcome equation and the error in the participation decision, and then, using the estimated parameter, the computation of quantile estimates through rotated quantile regression.

2. We focus only on the implementation of models with single-parameter copulas.

The first step consists of estimating the propensity score γ by a probit regression:

$$\hat{\gamma} = argmax_a \sum_{i=1}^{N} D_i ln\Phi(Z_i'a) + (1 - D_i) ln\Phi(-Z_i'a) \tag{4}$$

The second step is to estimate ρ by generalized method of moments, which allow us to obtain an observation-specific measure of dependence between the rank error in the equation of interest and the rank error in the selection equation. This step consists of working with a parametric copula and deriving moment restrictions on the copula parameter:

$$\hat{\rho} = argmin_c \| \sum_{i=1}^{N} \sum_{l=1}^{L} D_i \varphi(\tau_l, Z_i)[\mathbf{1}\{Y_i \leq X_i' \hat{\beta}_{\tau_l}(c)\} - G(\tau_l, p(Z_i'; \hat{\gamma}), c)] \| \tag{5}$$

where $\|.\|$ is the Euclidean norm, $\tau_1 < \tau_2 < \ldots < \tau_L$ is a finite grid on $(0, 1)$, and the instrument functions are defined as $\varphi(\tau, Z_i)$ where the dim $\varphi \leq$ dim ρ and:

$$\hat{\beta}_\tau(c) = argmin_{b(\tau)} \sum_{i=1}^{N} D_i [G(\tau, p(Z_i'\hat{\gamma}); c)(Y_i - X_i'b)^+ + \tag{6}$$

$$(1 - G(\tau, p(Z_i'\hat{\gamma}; c))(Y_i - X_i'b)^-] \tag{7}$$

where $a^+ = max\{a, 0\}$, $a^- = max\{-a, 0\}$, and the grid of τ values on the unit interval as well as the instrument function are chosen by the researcher.

Lastly, using $\hat{\gamma}$ and $\hat{\rho}$ obtained above, the third step consists in computing $\hat{G}_{\tau i} = G(\tau, p(Z_i^{'}\hat{\gamma}); \hat{\rho})$ for all i to estimate $\beta(\tau)$ by minimizing a rotated check function of the form:

$$\hat{\beta}(\tau) = argmin_{b(\tau)} \sum_{i=1}^{N} D_i [\hat{G}_{\tau i}(Y_i - X_i'b(\tau))^+ + (1 - \hat{G}_{\tau i})(Y_i - X_i'b(\tau))^-] \tag{8}$$

where $\hat{\beta}(\tau)$ will be a consistent estimator of the τ-th quantile regression coefficient.

2.3 Copulas

The Arellano and Bonhomme (2017a) analysis covers the case where the copula is left unrestricted but for the implementation they focus on the case of identification where the copula depends on a low-dimensional vector of parameters.

In our empirical implementation, we only consider the case of a reduced set of one-dimensional copulas. We include the Gaussian, Frank, Farlie-Gumbel-Morgenstern (FGM), and Ali-Mikhail-Haq (AMH). Table 1 provides their respective functional forms.

Table 1: Copula functions

Copula name	$C(U,V;\rho)$	Range of ρ
Gaussian	$\Phi_2\{\Phi^{-1}(U),\Phi^{-1}(V);\rho\}$	$-1\leq\rho\leq 1$
Frank	$-\rho^{-1}log\{1+\frac{(e^{-\rho U}-1)(e^{-\rho V}-1)}{(e^{-\rho}-1)}\}$	$-\infty\leq\rho\leq\infty$
FGM	$UV\{1+\rho(1-U)(1-V)\}$	$-1\leq\rho\leq 1$
AHM	$UV\{1-\rho(1-U)(1-V)\}^{-1}$	$-1\leq\rho\leq 1$

2.4 Rotated quantile regression

As previously mentioned, the quantile estimates are obtained by minimizing a rotated check function (See equation 8). The minimization problem can be written as the following linear programming problem:[3]

$$Min_{\beta_\tau,u,v}\sum_{i=1}^{N}\hat{G}_{\tau i}u_i+(1-\hat{G}_{\tau i})v_i \tag{9}$$

such that:

$$\mathbf{y}-\mathbf{X}\beta_\tau=\mathbf{u}-\mathbf{v} \tag{10}$$
$$\mathbf{u}\geq\mathbf{0}_n \tag{11}$$
$$\mathbf{v}\geq\mathbf{0}_n \tag{12}$$

where $\mathbf{0}_n$ is a vector of 0s, $\mathbf{X}$ is the matrix of observations of the covariates, $\mathbf{y}$ is the vector of observations of the outcome, and $\mathbf{u}$ and $\mathbf{v}$ are added to the inequality constraint to transform it into an equality.

We solve this linear programming problem using the LinearProgram() class in Stata, and alternatively using the Stata integration with Python. The latter is much faster than the former. We provide a comparison of the execution time using the LinearProgram() class, the integration with Python, and a code in Matlab[4] in our empirical application at the end of Section 4.

3 The qregsel command

In this section we describe the `qregsel` command to implement a copula-based sample selection correction in quantile regression.

3. This closely follows the quantile regression example for linear programming available in the Mata reference manual (see example 3 for LinearProgram() in StataCorp (2019a)).
4. The code used to run the exercise is provided in the Appendix.

3.1 Syntax

The syntax of the `qregsel` command is:

`qregsel` *depvar* [*indepvars*] [*if*] [*in*] , *select(*[*depvar*$_S$ =] *varlist*$_S$*)*

`quantile(#) grid_min(grid_minvalue) grid_max(grid_maxvalue)`

`grid_length(grid_lengthvalue)` [`copula(`*copula*`) noconstant py plot`]

3.2 Options

`select(`[*depvar*$_S$ =] *varlist*$_S$`)` specifies the selection equation. If *depvar*$_S$ is specified, it should be coded as 0 and 1, with 0 indicating an outcome not observed for an observation and 1 indicating an outcome observed for an observation. *select()* is required.

`quantile(#)` estimate # quantiles. *quantile()* is required.

`grid_min(grid_minvalue)` specifies the minimum value to be considered in the grid search. *grid_min()* is required.

`grid_max(grid_maxvalue)` specifies the maximum value to be considered in the grid search. *grid_max()* is required.

`grid_length(grid_length)` specifies the length of the (evenly spaced) grid to be considered in the grid search. *grid_length()* is required.

`copula(`*copula*`)` specifies a copula function governing the dependence between the errors in the outcome equation and selection equation. *copula* may be one of the following: *gaussian*, *frank*, *fgm*, and *amh*. The default is *copula(gaussian)*.

`noconstant` suppresses the constant term in the outcome equation.

`py` calls python to estimate the rotated quantile regression. Python 3 or higher, and the packages pandas, io, requests, numpy, and cvxopt are required to be installed. This option is recommended because of speed considerations.

`plot` generates a graph of the value of the objective function being minimized over the values of rho (the parameter of the copula) considered in the grid search.

3.3 Returned values

`qregsel` saves the following in e():

Scalars	
`e(N)`	Number of observations
`e(rank)`	Number of parameters
`e(df_r)`	Degrees of freedom
`e(rho)`	Copula parameter
Macros	
`e(copula)`	Specified copula
`e(depvar)`	Dependent variable
`e(inddepvar)`	Independent variables
`e(cmdline)`	Command line
`e(outcome_eq)`	Outcome equation
`e(select_eq)`	Selection equation
`e(predict)`	Predict command name
`e(py)`	Called Python integration
`e(cmd)`	Command name
`e(title)`	Quantile selection model
Matrices	
`e(coefs)`	Coefficient matrix
`e(grid)`	Values of the objective function minimized over the grid
Functions	
`e(sample)`	Marks estimation sample

3.4 Prediction

After the execution of `qregsel`, the `predict` command is available to compute a counterfactual of the outcome variable corrected for sample selection. Here is its syntax:

`predict newvar` [*if*] [*in*]

The counterfactual outcomes are constructed by randomly generating an integer q between 1 and 99 for each individual in the full sample, and then using the quantile coefficients associated with each draw of q to produce a prediction of the qth quantile of the outcome distribution. This approach follows the conditional quantile decomposition method of Machado and Mata (2005) and has been recently applied for example in Bollinger et al. (2019).

3.5 Inference

Confidence intervals for any of the parameters can be estimated using methods such as the conventional nonparametric bootstrap, or alternatively using subsampling (Politis et al. (1999)) as done in Arellano and Bonhomme (2017b) due to the computational advantage when using large sample sizes.

In our empirical application we illustrate how to use bootstrap to create a confidence interval for the copula parameter.

4 Empirical Example

We use the fictional data set used in the documentation of the Heckman selection model in the Stata base reference manual (see StataCorp (2019b)) to study wages of women. As in the example, we assume that the hourly wage is a function of education and age, whereas the likelihood of working (and hence the wage being observed) is a function of marital status, the number of children at home, and (implicitly) the wage (via the inclusion of age and education). We do not take the logarithm of wage as it is usually done, however the variable in the fictional data set has already a bell-shaped histogram (see Figure 1). In addition, we follow the example in the Stata 16 base reference manual by not including squared age as it is standard in this type of regression.

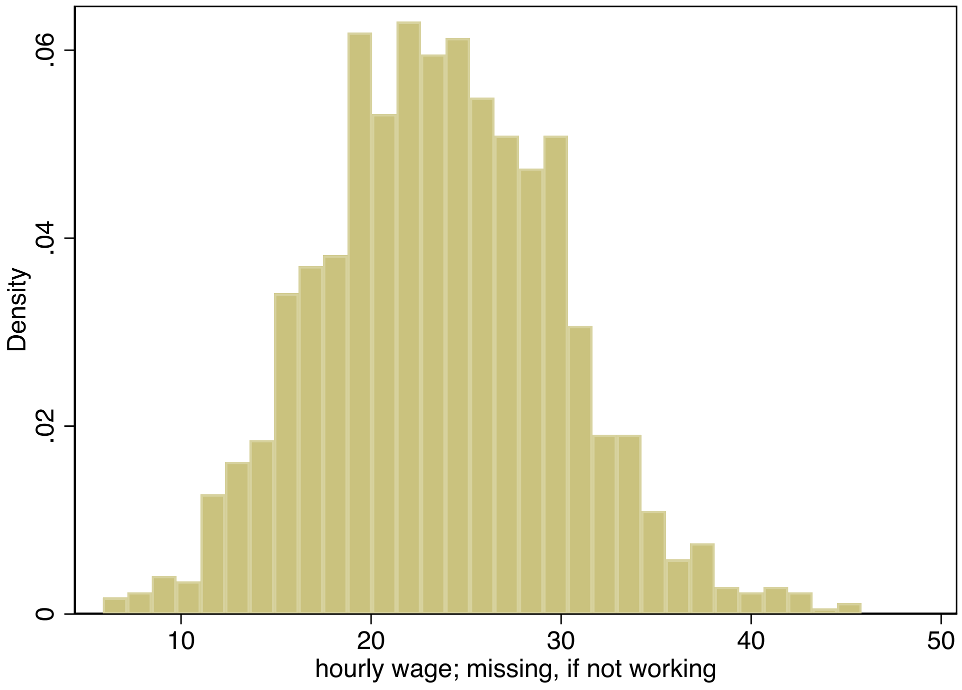

Figure 1: Histogram of wage

First, we estimate a quantile regression over the quantiles 0.1, 0.5, and 0.9 without corrections for sample selection.

```
. webuse womenwk,clear

. sqreg wage educ age, quantile(.1 .5 .9)
(fitting base model)

Bootstrap replications (20)
----+--- 1 ---+--- 2 ---+--- 3 ---+--- 4 ---+--- 5
....................
```

```
Simultaneous quantile regression                    Number of obs =      1,343
  bootstrap(20) SEs                                 .10 Pseudo R2 =     0.1068
                                                    .50 Pseudo R2 =     0.1429
                                                    .90 Pseudo R2 =     0.1523

------------------------------------------------------------------------------
             |              Bootstrap
        wage |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
q10          |
   education |   .8578176   .0651588    13.17   0.000     .7299933     .985642
         age |   .1234271   .0247599     4.98   0.000     .0748547    .1719995
       _cons |   .5154006   1.298974     0.40   0.692    -2.032842    3.063644
-------------+----------------------------------------------------------------
q50          |
   education |   .9064927   .0734632    12.34   0.000     .7623772    1.050608
         age |    .160184   .0249218     6.43   0.000      .111294    .2090739
       _cons |   5.312029   1.235723     4.30   0.000     2.887867     7.73619
-------------+----------------------------------------------------------------
q90          |
   education |    .930661   .0998569     9.32   0.000     .7347682    1.126554
         age |   .1579835   .0331353     4.77   0.000     .0929808    .2229863
       _cons |   12.20975    1.90783     6.40   0.000     8.467094    15.95241
------------------------------------------------------------------------------
```

Next we turn to the estimation of a quantile regression accounting for sample selection by using the command **qregsel** with a Gaussian copula. We specify a grid that goes between -.9 and .9 with steps of length .05.

```
. timer clear

. timer on 1

. global wage_eqn wage educ age

. global seleqn married children educ age

. qregsel $wage_eqn, select($seleqn) quantile(.1 .5 .9) copula(gaussian) py ///
>  grid_min(-.9) grid_max(.9) grid_length(.05) plot

Quantile selection model                         Number of obs      =     1343
             |        q10         q50         q90
-------------+------------------------------------
   education |   1.096502    1.018001    .8892478
         age |   .1959865    .2073785     .229075
       _cons |  -8.150604     .302514    8.807486

. ereturn list

scalars:
                  e(N) =  1343
               e(rank) =  3
               e(df_r) =  1340
                e(rho) =  -.7

macros:
             e(copula) : "gaussian"
             e(depvar) : "wage"
          e(indepvars) : "education age _cons"
            e(cmdline) : "qregsel wage  education age, select(married children educ age)"
         e(outcome_eq) : "wage  education age"
       e(selection_eq) : "married children educ age"
                 e(py) : "py"
                 e(cmd) : "qregsel"
             e(predict) : "qregsel_p"
               e(title) : "Quantile selection model"
matrices:
               e(coefs) :  3 x 3
                e(grid) :  37 x 1
functions:
              e(sample)
. timer off 1
. timer list 1
   1:    200.35 /        1 =     200.3470
```

This estimation took approximately 200 seconds in a Windows computer with Stata/IC 16.1 that had an Intel Core i5 @2.9 GHz processor, with 6 physical cores, and 32 GB of RAM. Using LinearProgram() class (i.e. removing the option py) took 7,148 seconds. Finally, using Matlab R2015b and the code provided in the Appendix, which it is based on a code obtained from the supplemental material in Arellano and Bonhomme (2017b), the execution took approximately 4 seconds.

Figure 3 shows the plot we obtain from specifying the option plot. The value of rho that minimizes the criterion function is equal to -0.7, as stored in e(rho).

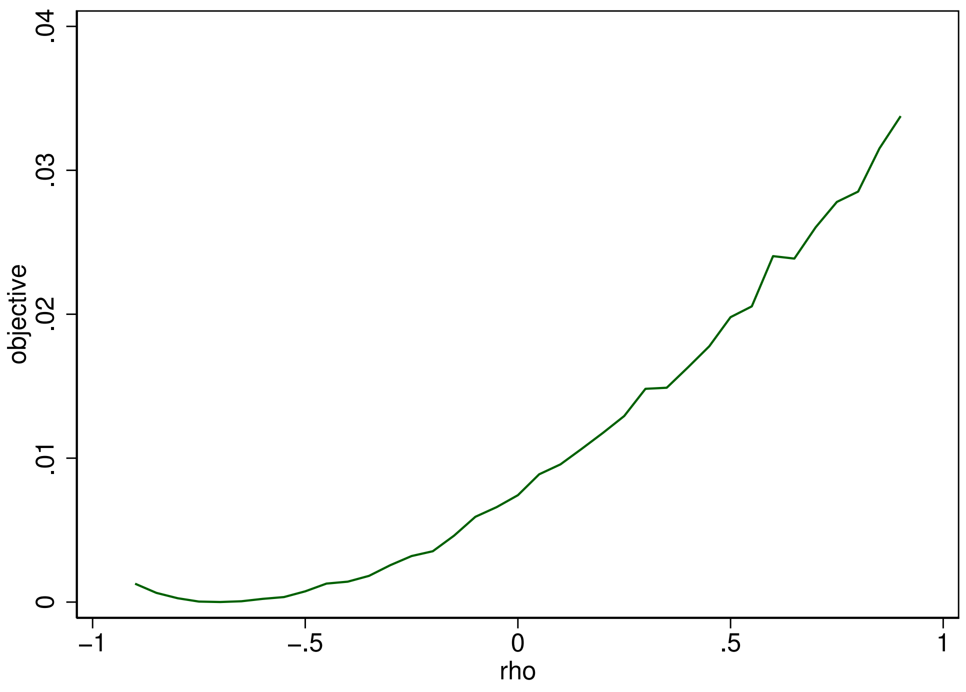

Figure 2: Grid for minimization

After the estimation a counterfactual distribution that is corrected for sample selection may be generated with the post estimation command `predict` as follows.

```
. set seed 1
. predict wage_hat
. _pctile wage_hat, nq(20)
. mat qs = J(19,3,.)
. forvalues i=1/19{
  2. mat qs[`i´,1] = r(r`i´)
  3. }
. _pctile wage, nq(20)
. forvalues i=1/19{
  2. mat qs[`i´,2] = r(r`i´)
  3. mat qs[`i´,3] = `i´
  4. }
. svmat qs, name(quantiles)
. twoway connected quantiles1 quantiles2 quantiles3, scheme(s1color) ///
>  xtitle("Ventile") ytitle("Wage") legend(order(1 "Corrected" 2 "Uncorrected"))
```

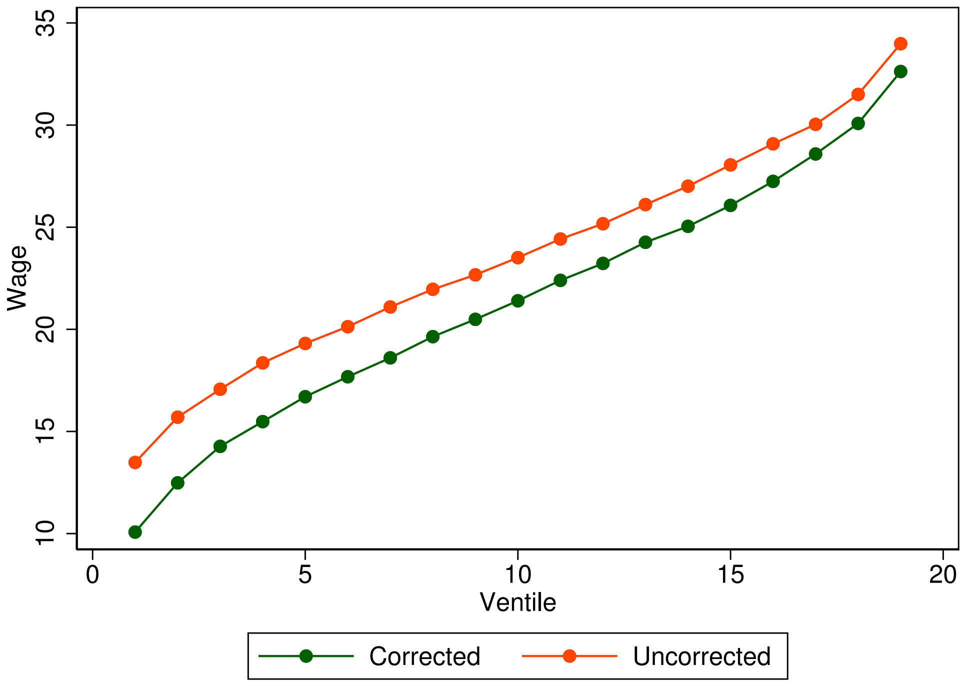

Figure 3: Corrected versus uncorrected quantiles

Finally, we illustrate the use of the `bootstrap` command to construct a confidence interval for the copula parameter ρ using 100 replications. This could be easily extended to the coefficients of the quantile regression or some other statistic computed with the counterfactual distribution.

```
. set seed 12345

. webuse womenwk,clear

. g d=!missing(wage)

. global wage_eqn wage educ age

. global seleqn married children educ age

. capture program drop myqregsel

. program myqregsel, eclass
  1.      version 16
  2.      tempname bb
  3.      quietly qregsel $wage_eqn, select($seleqn) quantile(.5) py ///
>           grid_min(-.9) grid_max(.9) grid_length(.05)
  4.      matrix `bb´=e(rho)
  5.      ereturn post `bb´
  6.      ereturn local cmd="bootstrap"
  7. end

. bootstrap _b, reps(100) nowarn: myqregsel
(running myqregsel on estimation sample)

Bootstrap replications (100)
────┼─── 1 ───┼─── 2 ───┼─── 3 ───┼─── 4 ───┼─── 5
..................................................    50
..................................................   100

Bootstrap results                               Number of obs     =      2,000
                                                Replications      =        100

──────────────────────────────────────────────────────────────────────────────
             │   Observed   Bootstrap                         Normal-based
             │      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
─────────────┼────────────────────────────────────────────────────────────────
          c1 │        -.7   .0748922    -9.35   0.000     -.846786    -.553214
──────────────────────────────────────────────────────────────────────────────
```

5 Concluding remarks

In this article, we introduce a new Stata module called **qregsel**, which implements a copula-based method proposed in Arellano and Bonhomme (2017b) to correct for sample selection in quantile regressions. The command uses the new Stata 16 capabilities for linear programming to solve the rotated quantile regression problem and the integration with Python to achieve greater speed.

6 References

Arellano, M., and S. Bonhomme. 2017a. Sample Selection in Quantile Regression: A Survey. In *Handbook of Quantile Regression*, ed. R. Koenker, V. Chernozhukov, X. He, and L. Peng, 1st ed., chap. 13, 463. Chapman and Hall/CRC.

———. 2017b. Quantile Selection Models With an Application to Understanding Changes in Wage Inequality. *Econometrica* 85(1): 1–28.

Bollinger, C. R., B. Hirsch, C. Hokayem, and J. P. Ziliak. 2019. Trouble in the Tails? What We Know about Earnings Nonresponse Thirty Years after Lillard, Smith, and Welch. *Journal of Political Economy* 127(5): 2143–2185.

Hasebe, T. 2013. Copula-based Maximum-Likelihood Estimation of Sample-Selection Models. *The Stata Journal* 13: 547–573.

Heckman, J. J. 1979. Sample Selection Bias as a Specification Error. *Econometrica* 47(1): 153–161.

Huber, M., and B. Melly. 2015. A Test of the Conditional Independence Assumption in Sample Selection Models. *Journal of Applied Econometrics* 30(7): 1144–1168.

Koenker, R., and G. Bassett. 1978. Regression Quantiles. *Econometrica* 46(1): 33–50.

Machado, J. A. F., and J. Mata. 2005. Counterfactual Decomposition of Changes in Wage Distributions using Quantile Regression. *Journal of Applied Econometrics* 20: 445–465.

Politis, D., J. Romano, and M. Wolf. 1999. *(Springer Series in Statistics) Dimitris N. Politis, Joseph P. Romano, Michael Wolf-Subsampling-Springer (1999).pdf*. Springer Series in Statistics.

StataCorp. 2019a. *Mata Reference Manual*. College Station, TX: Stata Press.

———. 2019b. Stata 16 Base Reference Manual. College Station, TX: Stata Press.

Vella, F. 1998. Estimating Models with Sample Selection Bias: A Survey. *The Journal of Human Resources* 33(1): 127–169.

Appendix A: Matlab code

```
%% Code for selection-corrected quantile regression based on
    the code of
%% Manuel Arellano and Stephane Bonhomme.
%% The code implements the method using womenwk's data set
    from Stata.

%%% The code makes use of rq.m

%% Inputs:
%% participation indicator D (binary)
%% outcome variable Y
%% matrix of covariates X, not including the constant
%% matrix of excluded covariates B, not including the constant
    .
%% Z=(X,B) is constructed below

 tic
 clear
 load womenwk.mat;
Y=data(:,6);
D=data(:,6);
D(~isnan(D))=1;
D(isnan(D))=0;
X=data(:,2:3);
B=data(:,4:5);

%% Specification in this version:
%% propensity score: probit
%% copula: Gaussian
%% instrument function: varphi(Z) = propensity score p(Z)
%% grid of tau's for tau=.1,...,.9
%% estimates of beta(tau) for tau=.1,.5,.9
%% grid search for rho with 37 equidistant values

%% Estimate propensity score
Z=[X B];
gamma=glmfit(Z,D,'binomial','link','probit');
pZ=normcdf(gamma(1)+Z*gamma(2:end));

%% Instrument function
varphi=pZ;

%% Select participants
```

```
Y1=Y(D==1);
pZ1=pZ(D==1);
X1=X(D==1,:);
varphi1=varphi(D==1);
[N1 colx]=size(X1);

%%% Percentile values to estimate rho
vectau=(.1:.1:.9)';
[t n]=size(vectau);

%%% Start loop on rho
vecrhoa=(-.9:.05:.9)';
[s n]=size(vecrhoa);

%%% Objective function to be minimized
warning('off')
object=zeros(s,1);
for j=1:s
    rhoa=vecrhoa(j);
    obj=0;
        for k=1:t
            tau=vectau(k);
            %%% Gaussian copula
            G=copulacdf('Gaussian',[tau*ones(N1,1) pZ1],rhoa)./
                pZ1;
            %%% Rotated quantile regression
            beta=rq([ones(N1,1) X1],Y1,G);
            obj=obj+(mean( varphi1.*((Y1<=beta(1)+X1*beta(2:
                colx+1))-G) ));
         end
    object(j)=obj^2;
    j
end

%%% Minimize the objective function
[C I]=min(object);
rho=vecrhoa(I);

%%% Optional: plot objective function
plot(vecrhoa(1:37),object(1:37))

%%% Estimate selection-corrected quantile parameters beta(tau)
beta=zeros(colx+1,3);
i=1;
for tau=.1:.4:.9
```

```
    %%% Gaussian copula
    G=copulacdf('Gaussian',[tau*ones(N1,1) pZ1],rho)./pZ1;
    %%% Rotated quantile regression
     beta(:,i)=rq([ones(N1,1) X1],Y1,G);
     i=i+1;
end

%%% Outputs
display('copula parameter')
rho
display('quantile coefficients')
beta
%%% Note: the first column in beta corresponds to the intercept
    .
%%% different columns are different quantiles
toc
```

```
function b = rq(X, y, p)
% Construct the dual problem of quantile regression
% Solve it with lp_fnm
%
% Function rq_fnm of Daniel Morillo & Roger Koenker
% Found at: http://www.econ.uiuc.edu/~roger/rqn/rq.ox
% Translated from Ox to Matlab by Paul Eilers 1999
%
[m n] = size(X);
u = ones(m, 1);
a = (1 - p) .* u;
b = -lp_fnm(X', -y', X' * a, u, a)';

function y = lp_fnm(A, c, b, u, x)
% Solve a linear program by the interior point method:
% min(c * u), s.t. A * x = b and 0 < x < u
% An initial feasible solution has to be provided as x
%
% Function lp_fnm of Daniel Morillo & Roger Koenker
% Found at: http://www.econ.uiuc.edu/~roger/rqn/rq.ox
% Translated from Ox to Matlab by Paul Eilers 1999
% Modified slightly by Roger Koenker, April, 2001.

% Set some constants
  beta = 0.9995;
  small = 1e-5;
  max_it = 50;
  [m n] = size(A);
```

```

% Generate inital feasible point
  s = u - x;

  y = (A' \  c')';
  r = c - y * A;
  z = r .* (r > 0);
  w = z - r;
  gap = c * x - y * b + w * u;

% Start iterations
  it = 0;
  while gap > small & it < max_it
    it = it + 1;

%   Compute affine step
    q = 1 ./ (z' ./ x + w' ./ s);
    r = z - w;
    Q = sparse(1:n,1:n,q);
    AQ = A * Q;
    AQA = AQ * A';
    rhs = AQ * r';
    dy = (AQA \ rhs)';
    dx = q .* (dy * A - r)';
    ds = -dx;
    dz = -z .* (1 + dx ./ x)';
    dw = -w .* (1 + ds ./ s)';

%   Compute maximum allowable step lengths
    fx = bound(x, dx);
    fs = bound(s, ds);
    fw = bound(w, dw);
    fz = bound(z, dz);
    fp = min(fx, fs);
    fd = min(fw, fz);
    fp = min(min(beta * fp), 1);
    fd = min(min(beta * fd), 1);

%   If full step is feasible, take it. Otherwise modify it
    if min(fp, fd) < 1

%     Update mu
      mu = z * x + w * s;
      g = (z + fd * dz) * (x + fp * dx) + (w + fd * dw) * (s +
          fp * ds);
      mu = mu * (g / mu) ^3 / ( 2* n);

%      Compute modified step
       dxdz = dx .* dz';
       dsdw = ds .* dw';
       xinv = 1 ./ x;
       sinv = 1 ./ s;
       xi = mu * (xinv - sinv);
       rhs = rhs + A * ( q .* (dxdz - dsdw -xi));
       dy = (AQA \ rhs)';

       dx = q .* (A' * dy' + xi - r' -dxdz + dsdw);
       ds = -dx;
       dz = mu * xinv' - z - xinv' .* z .* dx' - dxdz';
       dw = mu * sinv' - w - sinv' .* w .* ds' - dsdw';

%      Compute maximum allowable step lengths
       fx = bound(x, dx);
       fs = bound(s, ds);
       fw = bound(w, dw);
       fz = bound(z, dz);
       fp = min(fx, fs);
       fd = min(fw, fz);
       fp = min(min(beta * fp), 1);
       fd = min(min(beta * fd), 1);

    end

%   Take the step
    x = x + fp * dx;
    s = s + fp * ds;
    y = y + fd * dy;
    w = w + fd * dw;
    z = z + fd * dz;

    gap = c * x - y * b + w * u;
%     disp(gap);
  end




function b = bound(x, dx)
% Fill vector with allowed step lengths
% Support function for lp_fnm
```

```
b = 1e20 + 0 * x;
f = find(dx < 0);
b(f) = -x(f) ./ dx(f);
```

Appendix B: Getting Python environment running

Python is a general-purpose programming language created in 1989 by Guido van Rossum, which is free and open source. In this Appendix we explain how to get a Python environment running.

We start by installing a Python distribution called Anaconda that contains the core Python language and compatible versions of the most popular scientific libraries. We can download and install the latest version of Anaconda in the following link: https://www.anaconda.com/distribution/download-section

Anaconda supplies a tool called conda to manage and upgrade your Anaconda packages. A good idea as first step after installing the software is to open an anaconda terminal as administrator and type "conda update anaconda", which will update the whole Anaconda distribution.

All the required packages (pandas, io, requests, and numpy) come with anaconda except for cvxopt. We can install it using the anaconda prompt and typing "conda install -c conda-forge cvxopt".

This should be enough to allow Stata run `qregsel` calling the integration with Python through the option py.