

# Title

**qreg** — Quantile regression

## Syntax

*Quantile regression*

```
qreg depvar [indepvars] [if] [in] [weight] [, qreg_options]
```

*Interquantile range regression*

```
iqreg depvar [indepvars] [if] [in] [, iqreg_options]
```

*Simultaneous-quantile regression*

```
sqreg depvar [indepvars] [if] [in] [, sqreg_options]
```

*Bootstrapped quantile regression*

```
bsqreg depvar [indepvars] [if] [in] [, bsqreg_options]
```

*Internal estimation command for quantile regression*

```
_qreg [depvar [indepvars] [if] [in] [weight]] [, _qreg_options]
```

<i>qreg_options</i>	Description
Model	
<u>q</u> uantile(#)	estimate # quantile; default is <code>quantile(.5)</code>
SE	
<u>v</u> ce(iid, [ <i>vceopts</i> ])	technique used to estimate standard errors
Reporting	
<u>l</u> evel(#)	set confidence level; default is <code>level(95)</code>
<i>display_options</i>	control column formats and line width
Optimization	
<i>optimization_options</i>	control the optimization process; seldom used
<u>w</u> lsiter(#)	attempt # weighted least-squares iterations before doing linear programming iterations

<i>vceopts</i>	Description
<i>denmethod</i>	nonparametric density estimation technique
<i>bwidth</i>	bandwidth method used by the density estimator

<i>denmethod</i>	Description
<u>fitted</u>	use the empirical quantile function using fitted values; the default
<u>residual</u>	use the empirical residual quantile function
<u>kernel</u> [ ( <i>kernel</i> ) ]	use the nonparametric kernel density estimator; default is epanechnikov
<hr/>	
<i>bwidth</i>	Description
<u>hsheather</u>	Hall–Sheather’s bandwidth; the default
<u>bofinger</u>	Bofinger’s bandwidth
<u>chamberlain</u>	Chamberlain’s bandwidth
<hr/>	
<i>kernel</i>	Description
<u>epanechnikov</u>	Epanechnikov kernel function; the default
<u>epan2</u>	alternative Epanechnikov kernel function
<u>biweight</u>	biweight kernel function
<u>cosine</u>	cosine trace kernel function
<u>gaussian</u>	Gaussian kernel function
<u>parzen</u>	Parzen kernel function
<u>rectangle</u>	rectangle kernel function
<u>triangle</u>	triangle kernel function
<hr/>	
<i>iqreg_options</i>	Description
Model	
<u>quantiles</u> (# #)	interquantile range; default is <code>quantiles(.25 .75)</code>
<u>reps</u> (#)	perform # bootstrap replications; default is <code>reps(20)</code>
Reporting	
<u>level</u> (#)	set confidence level; default is <code>level(95)</code>
<u>nodots</u>	suppress display of the replication dots
<u>display_options</u>	control column formats and line width
<hr/>	
<i>sqreg_options</i>	Description
Model	
<u>quantiles</u> (# [ # [ # ... ] ])	estimate # quantiles; default is <code>quantiles(.5)</code>
<u>reps</u> (#)	perform # bootstrap replications; default is <code>reps(20)</code>
Reporting	
<u>level</u> (#)	set confidence level; default is <code>level(95)</code>
<u>nodots</u>	suppress display of the replication dots
<u>display_options</u>	control column formats and line width

<i>bsqreg_options</i>	Description
Model	
<code>quantile(#)</code>	estimate # quantile; default is <code>quantile(.5)</code>
<code>reps(#)</code>	perform # bootstrap replications; default is <code>reps(20)</code>
Reporting	
<code>level(#)</code>	set confidence level; default is <code>level(95)</code>
<code>display_options</code>	control column formats and line width
<i>_qreg_options</i>	Description
<code>quantile(#)</code>	estimate # quantile; default is <code>quantile(.5)</code>
<code>level(#)</code>	set confidence level; default is <code>level(95)</code>
<code>accuracy(#)</code>	relative accuracy required for linear programming algorithm; should not be specified
<code>optimization_options</code>	control the optimization process; seldom used

`by`, `mi estimate`, `rolling`, `statsby`, and `xi` are allowed by `qreg`, `iqreg`, `sqreg`, and `bsqreg`; `fracpoly`, `mfp`, `nestreg`, and `stepwise` are allowed only with `qreg`; see [U] 11.1.10 **Prefix commands**.

`qreg` and `_qreg` allow `fweights` and `aweight`s; see [U] 11.1.6 **weight**.

See [U] 20 **Estimation and postestimation commands** for more capabilities of estimation commands.

## Menu

### qreg

Statistics > Nonparametric analysis > Quantile regression

### iqreg

Statistics > Nonparametric analysis > Interquantile regression

### sqreg

Statistics > Nonparametric analysis > Simultaneous-quantile regression

### bsqreg

Statistics > Nonparametric analysis > Bootstrapped quantile regression

## Description

`qreg` fits quantile (including median) regression models, also known as least–absolute-value models (LAV or MAD) and minimum L1-norm models. The quantile regression models fit by `qreg` express the quantiles of the conditional distribution as linear functions of the independent variables.

`iqreg` estimates interquantile range regressions, regressions of the difference in quantiles. The estimated variance–covariance matrix of the estimators (VCE) is obtained via bootstrapping.

`sqreg` estimates simultaneous-quantile regression. It produces the same coefficients as `qreg` for each quantile. Reported standard errors will be similar, but `sqreg` obtains an estimate of the VCE via bootstrapping, and the VCE includes between-quantile blocks. Thus you can test and construct confidence intervals comparing coefficients describing different quantiles.

`bsqreg` is equivalent to `sqreg` with one quantile.

`_qreg` is the internal estimation command for quantile regression. `_qreg` is not intended to be used directly; see *Methods and formulas* below.

## Options for qreg

### Model

`quantile(#)` specifies the quantile to be estimated and should be a number between 0 and 1, exclusive. Numbers larger than 1 are interpreted as percentages. The default value of 0.5 corresponds to the median.

### SE

`vce(iid, [denmethod] [bwidth])` specifies the density estimation method and the bandwidth method to use in computing the variance–covariance of the estimator (VCE). As discussed in *Methods and formulas*, the estimator for the VCE uses a nonparametric density estimator for which there are several choices.

`denmethod` specifies the method to use for the nonparametric density estimator. Available methods are `fitted`, `residual`, or `kernel` [ (*kernel*) ], where the optional *kernel* must be one of the kernel choices listed below.

`fitted` and `residual` specify that the nonparametric density estimator use some of the structure imposed by quantile regression. The default `fitted` uses a function of the fitted values and `residual` uses a function of the residuals.

`kernel()` specifies that the nonparametric density estimator use a kernel method. The available kernel functions are `epanechnikov`, `epan2`, `biweight`, `cosine`, `gaussian`, `parzen`, `rectangle`, and `triangle`. The default is `epanechnikov`. See [R] **kdensity** for the kernel function forms.

`bwidth` specifies the bandwidth method to use by the nonparametric density estimator. Available methods are `hsheather`, `bofinger`, or `chamberlain`.

Some intuition and all the details for the choices of `denmethod` and `bwidth` are given in *Methods and formulas*.

### Reporting

`level(#)`; see [R] **estimation options**.

`display_options`: `cformat(%fmt)`, `pformat(%fmt)`, `sformat(%fmt)`, and `no stretch`; see [R] **estimation options**.

### Optimization

`optimization_options`: `iterate(#)`, `[no] log`, `trace`. `iterate()` specifies the maximum number of iterations; `log/no log` specifies whether to show the iteration log; and `trace` specifies that the iteration log should include the current parameter vector. These options are seldom used.

`wlsiter(#)` specifies the number of weighted least-squares iterations that will be attempted before the linear programming iterations are started. The default value is 1. If there are convergence problems, increasing this number should help.

## Options for iqreg

### Model

`quantiles(# #)` specifies the quantiles to be compared. The first number must be less than the second, and both should be between 0 and 1, exclusive. Numbers larger than 1 are interpreted as percentages. Not specifying this option is equivalent to specifying `quantiles(.25 .75)`, meaning the interquartile range.

`reps(#)` specifies the number of bootstrap replications to be used to obtain an estimate of the variance–covariance matrix of the estimators (standard errors). `reps(20)` is the default and is arguably too small. `reps(100)` would perform 100 bootstrap replications. `reps(1000)` would perform 1,000 replications.

### Reporting

`level(#)`; see [R] **estimation options**.

`nodots` suppresses display of the replication dots.

*display\_options*: `cformat(%fmt)`, `pformat(%fmt)`, `sformat(%fmt)`, and `nolstretch`; see [R] **estimation options**.

## Options for sqreg

### Model

`quantiles(# [# [# ... ]])` specifies the quantiles to be estimated and should contain numbers between 0 and 1, exclusive. Numbers larger than 1 are interpreted as percentages. The default value of 0.5 corresponds to the median.

`reps(#)` specifies the number of bootstrap replications to be used to obtain an estimate of the variance–covariance matrix of the estimators (standard errors). `reps(20)` is the default and is arguably too small. `reps(100)` would perform 100 bootstrap replications. `reps(1000)` would perform 1,000 replications.

### Reporting

`level(#)`; see [R] **estimation options**.

`nodots` suppresses display of the replication dots.

*display\_options*: `cformat(%fmt)`, `pformat(%fmt)`, `sformat(%fmt)`, and `nolstretch`; see [R] **estimation options**.

## Options for bsqreg

### Model

`quantile(#)` specifies the quantile to be estimated and should be a number between 0 and 1, exclusive. Numbers larger than 1 are interpreted as percentages. The default value of 0.5 corresponds to the median.

`reps(#)` specifies the number of bootstrap replications to be used to obtain an estimate of the variance–covariance matrix of the estimators (standard errors). `reps(20)` is the default and is arguably too small. `reps(100)` would perform 100 bootstrap replications. `reps(1000)` would perform 1,000 replications.

#### Reporting

`level(#)`; see [R] [estimation options](#).

`display_options`: `cformat(%fmt)`, `pformat(%fmt)`, `sformat(%fmt)`, and `nolstretch`; see [R] [estimation options](#).

## Options for `_qreg`

`quantile(#)` specifies the quantile to be estimated and should be a number between 0 and 1, exclusive. The default value of 0.5 corresponds to the median.

`level(#)`; see [R] [estimation options](#).

`accuracy(#)` should not be specified; it specifies the relative accuracy required for the linear programming algorithm. If the potential for improving the sum of weighted deviations by deleting an observation from the basis is less than this on a percentage basis, the algorithm will be said to have converged. The default value is  $10^{-10}$ .

`optimization_options`: `iterate(#)`, `[no]log`, `trace`. `iterate()` specifies the maximum number of iterations; `log/nolog` specifies whether to show the iteration log; and `trace` specifies that the iteration log should include the current parameter vector. These options are seldom used.

## Remarks

Remarks are presented under the following headings:

*Median regression*

*Quantile regression*

*Estimated standard errors*

*Interquantile and simultaneous-quantile regression*

*What are the parameters?*

## Median regression

`qreg` fits quantile regression models. The default form is median regression, where the objective is to estimate the median of the dependent variable, conditional on the values of the independent variables. This method is similar to ordinary regression, where the objective is to estimate the conditional mean of the dependent variable. Simply put, median regression finds a line through the data that minimizes the sum of the *absolute* residuals rather than the sum of the *squares* of the residuals, as in ordinary regression. Equivalently, median regression expresses the median of the conditional distribution of the dependent variable as a linear function of the conditioning (independent) variables. Cameron and Trivedi (2010, chap. 7) provide a nice introduction to quantile regression using Stata.

## ▷ Example 1

Consider a two-group experimental design with 5 observations per group:

```
. use http://www.stata-press.com/data/r12/twogrp
. list
```

	x	y
1.	0	0
2.	0	1
3.	0	3
4.	0	4
5.	0	95
6.	1	14
7.	1	19
8.	1	20
9.	1	22
10.	1	23

```
. qreg y x
Iteration 1: WLS sum of weighted deviations = 121.88268
Iteration 1: sum of abs. weighted deviations = 111
Iteration 2: sum of abs. weighted deviations = 110
Median regression
Raw sum of deviations      157 (about 14)
Min sum of deviations      110
Number of obs = 10
Pseudo R2 = 0.2994
```

	y	Coef.	Std. Err.	t	P> t	[95% Conf. Interval]
	x	17	18.23213	0.93	0.378	-25.04338 59.04338
	_cons	3	12.89207	0.23	0.822	-26.72916 32.72916

We have estimated the equation

$$y_{\text{median}} = 3 + 17x$$

We look back at our data.  $x$  takes on the values 0 and 1, so the median for the  $x = 0$  group is 3, whereas for  $x = 1$  it is  $3 + 17 = 20$ . The output reports that the raw sum of absolute deviations about 14 is 157; that is, the sum of  $|y - 14|$  is 157. Fourteen is the unconditional median of  $y$ , although in these data, any value between 14 and 19 could also be considered an unconditional median (we have an even number of observations, so the median is bracketed by those two values). In any case, the raw sum of deviations of  $y$  about the median would be the same no matter what number we choose between 14 and 19. (With a “median” of 14, the raw sum of deviations is 157. Now think of choosing a slightly larger number for the median and recalculating the sum. Half the observations will have larger negative residuals, but the other half will have smaller positive residuals, resulting in no net change.)

We turn now to the actual estimated equation. The sum of the absolute deviations about the solution  $y_{\text{median}} = 3 + 17x$  is 110. The pseudo- $R^2$  is calculated as  $1 - 110/157 \approx 0.2994$ . This result is based on the idea that the median regression is the maximum likelihood estimate for the double-exponential distribution.

## □ Technical note

`qreg` is an alternative to regular regression or robust regression—see [R] `regress` and [R] `rreg`. Let's compare the results:

```
. regress y x
```

Source	SS	df	MS	Number of obs = 10		
Model	2.5	1	2.5	F( 1, 8) =	0.00	
Residual	6978.4	8	872.3	Prob > F =	0.9586	
Total	6980.9	9	775.655556	R-squared =	0.0004	
				Adj R-squared =	-0.1246	
				Root MSE =	29.535	

  

y	Coef.	Std. Err.	t	P> t	[95% Conf. Interval]	
x	-1	18.6794	-0.05	0.959	-44.07477	42.07477
_cons	20.6	13.20833	1.56	0.157	-9.858465	51.05847

Unlike `qreg`, `regress` fits ordinary linear regression and is concerned with predicting the mean rather than the median, so both results are, in a technical sense, correct. Putting aside those technicalities, however, we tend to use either regression to describe the central tendency of the data, of which the mean is one measure and the median another. Thus we can ask, “which method better describes the central tendency of these data?”

Means—and therefore ordinary linear regression—are sensitive to outliers, and our data were purposely designed to contain two such outliers: 95 for  $x = 0$  and 14 for  $x = 1$ . These two outliers dominated the ordinary regression and produced results that do not reflect the central tendency well—you are invited to enter the data and graph  $y$  against  $x$ .

Robust regression attempts to correct the outlier-sensitivity deficiency in ordinary regression:

```
. rreg y x, genwt(wt)
```

```
Huber iteration 1: maximum difference in weights = .7311828
Huber iteration 2: maximum difference in weights = .17695779
Huber iteration 3: maximum difference in weights = .03149585
Biweight iteration 4: maximum difference in weights = .1979335
Biweight iteration 5: maximum difference in weights = .23332905
Biweight iteration 6: maximum difference in weights = .09960067
Biweight iteration 7: maximum difference in weights = .02691458
Biweight iteration 8: maximum difference in weights = .0009113
```

```
Robust regression
```

y	Coef.	Std. Err.	t	P> t	[95% Conf. Interval]	
x	18.16597	2.023114	8.98	0.000	13.50066	22.83128
_cons	2.000003	1.430558	1.40	0.200	-1.298869	5.298875

Here `rreg` discarded the first outlier completely. (We know this because we included the `genwt()` option on `rreg` and, after fitting the robust regression, examined the weights.) For the other “outlier”, `rreg` produced a weight of 0.47.

In any case, the answers produced by `qreg` and `rreg` to describe the central tendency are similar, but the standard errors are different. In general, robust regression will have smaller standard errors because it is not as sensitive to the exact placement of observations near the median. You are welcome to try removing the first outlier in the `qreg` estimation to observe an improvement in the standard errors by typing



```
. qreg y x if _n!=5
```

Also, some authors (Rousseeuw and Leroy 1987, 11) have noted that quantile regression, unlike the unconditional median, may be sensitive to even one outlier if its leverage is high enough. Rousseeuw and Leroy (1987) discuss estimators that are more robust to perturbations to the data than either mean regression or quantile regression.

In the end, quantile regression may be more useful for the interpretation of the parameters that it estimates than for its robustness to perturbations to the data. □

## ▷ Example 2

Let's now consider a less artificial example using the automobile data described in [U] **1.2.2 Example datasets**. Using median regression, we will regress each car's price on its weight and length and whether it is of foreign manufacture:

```
. use http://www.stata-press.com/data/r12/auto, clear
(1978 Automobile Data)

. qreg price weight length foreign
Iteration 1: WLS sum of weighted deviations = 112795.66
Iteration 1: sum of abs. weighted deviations = 111901
Iteration 2: sum of abs. weighted deviations = 110529.43
Iteration 3: sum of abs. weighted deviations = 109524.57
Iteration 4: sum of abs. weighted deviations = 109468.3
Iteration 5: sum of abs. weighted deviations = 109105.27
note: alternate solutions exist
Iteration 6: sum of abs. weighted deviations = 108931.02
Iteration 7: sum of abs. weighted deviations = 108887.4
Iteration 8: sum of abs. weighted deviations = 108822.59

Median regression                                Number of obs =          74
  Raw sum of deviations  142205 (about 4934)
  Min sum of deviations 108822.6                Pseudo R2      =      0.2347
```

price	Coef.	Std. Err.	t	P> t	[95% Conf. Interval]	
weight	3.933588	1.328718	2.96	0.004	1.283543	6.583632
length	-41.25191	45.46469	-0.91	0.367	-131.9284	49.42456
foreign	3377.771	885.4198	3.81	0.000	1611.857	5143.685
_cons	344.6489	5182.394	0.07	0.947	-9991.31	10680.61

The estimated equation is

$$\text{price}_{\text{median}} = 3.93 \text{ weight} - 41.25 \text{ length} + 3377.8 \text{ foreign} + 344.65$$

The output may be interpreted in the same way as linear regression output; see [R] **regress**. The variables `weight` and `foreign` are significant, but `length` is not significant. The median price of the cars in these data is \$4,934. This value is a median (one of the two center observations), not the median, which would typically be defined as the midpoint of the two center observations. ◀

## Quantile regression

Quantile regression is similar to median regression in that it estimates an equation expressing a quantile of the conditional distribution, albeit one that generally differs from the 0.5 quantile that is the median. For example, specifying `quantile(.25)` estimates the parameters that describe the 25th percentile (first quartile) of the conditional distribution.

Quantile regression allows for effects of the independent variables to differ over the quantiles. For example, Chamberlain (1994) finds that union membership has a larger effect on the lower quantiles than on the higher quantiles of the conditional distribution of U.S. wages. That the effects of the independent variables may vary over quantiles of the conditional distribution is an important advantage of quantile regression over mean regression.

### ► Example 3

Returning to real data, the equation for the 25th percentile of price conditional on weight, length, and foreign in our automobile data is

```
. use http://www.stata-press.com/data/r12/auto
(1978 Automobile Data)

. qreg price weight length foreign, quantile(.25)
Iteration 1: WLS sum of weighted deviations = 98938.466
Iteration 1: sum of abs. weighted deviations = 99457.766
Iteration 2: sum of abs. weighted deviations = 91339.779
Iteration 3: sum of abs. weighted deviations = 86833.291
Iteration 4: sum of abs. weighted deviations = 83894.441
Iteration 5: sum of abs. weighted deviations = 82186.049
Iteration 6: sum of abs. weighted deviations = 75246.848
Iteration 7: sum of abs. weighted deviations = 71442.907
Iteration 8: sum of abs. weighted deviations = 70452.616
Iteration 9: sum of abs. weighted deviations = 69646.639
Iteration 10: sum of abs. weighted deviations = 69603.554

.25 Quantile regression
Raw sum of deviations 83825.5 (about 4187)      Number of obs =      74
Min sum of deviations 69603.55                Pseudo R2      =      0.1697
```

price	Coef.	Std. Err.	t	P> t	[95% Conf. Interval]	
weight	1.831789	.6328903	2.89	0.005	.5695289	3.094049
length	2.84556	21.65558	0.13	0.896	-40.34514	46.03626
foreign	2209.925	421.7401	5.24	0.000	1368.791	3051.059
_cons	-1879.775	2468.46	-0.76	0.449	-6802.963	3043.413

Compared with our previous median regression, the coefficient on length now has a positive sign, and the coefficients on foreign and weight are reduced. The actual lower quantile is \$4,187, substantially less than the median \$4,934.

We can also estimate the upper quartile as a function of the same three variables:

```
. qreg price weight length foreign, quantile(.75)
Iteration 1: WLS sum of weighted deviations = 110931.48
Iteration 1: sum of abs. weighted deviations = 111305.91
Iteration 2: sum of abs. weighted deviations = 105989.57
Iteration 3: sum of abs. weighted deviations = 100378.89
Iteration 4: sum of abs. weighted deviations = 99796.49
Iteration 5: sum of abs. weighted deviations = 98796.212
Iteration 6: sum of abs. weighted deviations = 98483.669
Iteration 7: sum of abs. weighted deviations = 98395.935
```

```
.75 Quantile regression
Raw sum of deviations 159721.5 (about 6342)
Min sum of deviations 98395.94
Number of obs = 74
Pseudo R2 = 0.3840
```

price	Coef.	Std. Err.	t	P> t	[95% Conf. Interval]	
weight	9.22291	1.785767	5.16	0.000	5.66131	12.78451
length	-220.7833	61.10352	-3.61	0.001	-342.6504	-98.91616
foreign	3595.133	1189.984	3.02	0.004	1221.785	5968.482
_cons	20242.9	6965.02	2.91	0.005	6351.61	34134.2

This result tells a different story: `weight` is much more important, and `length` is now significant—with a negative coefficient! The prices of high-priced cars seem to be determined by factors different from those affecting the prices of low-priced cars.

◀

## □ Technical note

One explanation for having substantially different regression functions for different quantiles is that the data are heteroskedastic, as we will demonstrate below. The following statements create a sharply heteroskedastic set of data:

```
. drop _all
. set obs 10000
obs was 0, now 10000
. set seed 50550
. gen x = .1 + .9 * runiform()
. gen y = x * runiform()^2
```

Let's now fit the regressions for the 5th and 95th quantiles:

```
. qreg y x, quantile(.05)
Iteration 1: WLS sum of weighted deviations = 1080.7273
Iteration 1: sum of abs. weighted deviations = 1078.3192
Iteration 2: sum of abs. weighted deviations = 282.73545
Iteration 3: sum of abs. weighted deviations = 182.46921
Iteration 4: sum of abs. weighted deviations = 182.25456
Iteration 5: sum of abs. weighted deviations = 182.2527
Iteration 6: sum of abs. weighted deviations = 182.25247
Iteration 7: sum of abs. weighted deviations = 182.25246
Iteration 8: sum of abs. weighted deviations = 182.25245
Iteration 9: sum of abs. weighted deviations = 182.25244
.05 Quantile regression
Raw sum of deviations 182.357 (about .0009234)
Min sum of deviations 182.2524
Number of obs = 10000
Pseudo R2 = 0.0006
```

y	Coef.	Std. Err.	t	P> t	[95% Conf. Interval]	
x	.002601	.0004576	5.68	0.000	.001704	.003498
_cons	-.0001393	.0002782	-0.50	0.617	-.0006846	.000406

```

. qreg y x, quantile(.95)
Iteration 1: WLS sum of weighted deviations = 1237.5569
Iteration 1: sum of abs. weighted deviations = 1238.0014
Iteration 2: sum of abs. weighted deviations = 456.65044
Iteration 3: sum of abs. weighted deviations = 338.45497
Iteration 4: sum of abs. weighted deviations = 338.43897
Iteration 5: sum of abs. weighted deviations = 338.4389

.95 Quantile regression          Number of obs =    10000
Raw sum of deviations 554.6889 (about .61326343)
Min sum of deviations 338.4389          Pseudo R2      =    0.3899

```

y	Coef.	Std. Err.	t	P> t	[95% Conf. Interval]	
x	.8898259	.0090984	97.80	0.000	.8719912	.9076605
_cons	.0021514	.0055307	0.39	0.697	-.00869	.0129927

The coefficient on  $x$ , in particular, differs markedly between the two estimates. For the mathematically inclined, it is not too difficult to show that the theoretical lines are  $y = 0.0025x$  for the 5th percentile and  $y = 0.9025x$  for the 95th, numbers in close agreement with our numerical results.

The estimator for the standard errors computed by `qreg` assumes that the sample is independent and identically distributed (i.i.d.); see *Estimated standard errors* and *Methods and formulas* for details. Because the data are conditionally heteroskedastic, we should have used `bsqreg` to consistently estimate the standard errors using a bootstrap method.

□

## Estimated standard errors

The variance–covariance matrix of the estimator (VCE) depends on the reciprocal of the density of the dependent variable evaluated at the quantile of interest. This function, known as the “sparsity function”, is hard to estimate.

The default method, which uses the fitted values for the predicted quantiles, generally performs well, but other methods may be preferred in larger samples. The `vce()` suboptions *denmethod* and *bwidth* provide other estimators of the sparsity function, the details of which are described in *Methods and formulas*.

## Interquantile and simultaneous-quantile regression

Consider a quantile regression model where the  $q$ th quantile is given by

$$Q_q(y) = a_q + b_{q,1}x_1 + b_{q,2}x_2$$

For instance, the 75th and 25th quantiles are given by

$$Q_{0.75}(y) = a_{0.75} + b_{0.75,1}x_1 + b_{0.75,2}x_2$$

$$Q_{0.25}(y) = a_{0.25} + b_{0.25,1}x_1 + b_{0.25,2}x_2$$

The difference in the quantiles is then

$$Q_{0.75}(y) - Q_{0.25}(y) = (a_{0.75} - a_{0.25}) + (b_{0.75,1} - b_{0.25,1})x_1 + (b_{0.75,2} - b_{0.25,2})x_2$$

`qreg` fits models such as  $Q_{0.75}(y)$  and  $Q_{0.25}(y)$ . `iqreg` fits interquantile models, such as  $Q_{0.75}(y) - Q_{0.25}(y)$ . The relationships of the coefficients estimated by `qreg` and `iqreg` are exactly as shown: `iqreg` reports coefficients that are the difference in coefficients of two `qreg` models, and, of course, `iqreg` reports the appropriate standard errors, which it obtains by bootstrapping.

`sqreg` is like `qreg` in that it estimates the equations for the quantiles

$$Q_{0.75}(y) = a_{0.75} + b_{0.75,1}x_1 + b_{0.75,2}x_2$$

$$Q_{0.25}(y) = a_{0.25} + b_{0.25,1}x_1 + b_{0.25,2}x_2$$

The coefficients it obtains are the same that would be obtained by estimating each equation separately using `qreg`. `sqreg` differs from `qreg` in that it estimates the equations simultaneously and obtains an estimate of the entire variance–covariance matrix of the estimators by bootstrapping. Thus you can perform hypothesis tests concerning coefficients both within and across equations.

For example, to fit the above model, you could type

```
. qreg y x1 x2, quantile(.25)
. qreg y x1 x2, quantile(.75)
```

Doing this, you would obtain estimates of the parameters, but you could not test whether  $b_{0.25,1} = b_{0.75,1}$  or, equivalently,  $b_{0.75,1} - b_{0.25,1} = 0$ . If your interest really is in the difference of coefficients, you could type

```
. iqreg y x1 x2, quantiles(.25 .75)
```

The “coefficients” reported would be the difference in quantile coefficients. You could also estimate both quantiles simultaneously and then test the equality of the coefficients:

```
. sqreg y x1 x2, quantiles(.25 .75)
. test [q25]x1 = [q75]x1
```

Whether you use `iqreg` or `sqreg` makes no difference for this test. `sqreg`, however, because it estimates the quantiles simultaneously, allows you to test other hypotheses. `iqreg`, by focusing on quantile differences, presents results in a way that is easier to read.

Finally, `sqreg` can estimate quantiles singly,

```
. sqreg y x1 x2, quantiles(.5)
```

and can thereby be used as a substitute for the slower `bsqreg`. (Gould [1997b] presents timings demonstrating that `sqreg` is faster than `bsqreg`.) `sqreg` can also estimate more than two quantiles simultaneously:

```
. sqreg y x1 x2, quantiles(.25 .5 .75)
```

## ▷ Example 4

In demonstrating `qreg`, we performed quantile regressions using the automobile data. We discovered that the regression of `price` on `weight`, `length`, and `foreign` produced vastly different coefficients for the 0.25, 0.5, and 0.75 quantile regressions. Here are the coefficients that we obtained:

Variable	25th percentile	50th percentile	75th percentile
<code>weight</code>	1.83	3.93	9.22
<code>length</code>	2.85	−41.25	−220.8
<code>foreign</code>	2209.9	3377.8	3595.1
<code>_cons</code>	−1879.8	344.6	20242.9

All we can say, having estimated these equations separately, is that price seems to depend differently on the weight, length, and foreign variables depending on the portion of the price distribution we examine. We cannot be more precise because the estimates have been made separately. With `sqreg`, however, we can estimate all the effects simultaneously:

```
. use http://www.stata-press.com/data/r12/auto, clear
(1978 Automobile Data)
. set seed 1001
. sqreg price weight length foreign, q(.25 .5 .75) reps(100)
(fitting base model)
(bootstrapping .....
> .....)
```

Simultaneous quantile regression  
bootstrap(100) SEs

Number of obs =	74
.25 Pseudo R2 =	0.1697
.50 Pseudo R2 =	0.2347
.75 Pseudo R2 =	0.3840

	price	Coef.	Bootstrap Std. Err.	t	P> t	[95% Conf. Interval]	
<hr/>							
q25							
	weight	1.831789	1.574777	1.16	0.249	-1.309005	4.972583
	length	2.84556	38.63523	0.07	0.941	-74.20998	79.9011
	foreign	2209.925	1008.521	2.19	0.032	198.494	4221.357
	_cons	-1879.775	3665.184	-0.51	0.610	-9189.753	5430.203
<hr/>							
q50							
	weight	3.933588	2.529541	1.56	0.124	-1.111423	8.978599
	length	-41.25191	68.62258	-0.60	0.550	-178.1153	95.61151
	foreign	3377.771	1017.422	3.32	0.001	1348.586	5406.956
	_cons	344.6489	6199.257	0.06	0.956	-12019.38	12708.68
<hr/>							
q75							
	weight	9.22291	2.483676	3.71	0.000	4.269374	14.17645
	length	-220.7833	86.17422	-2.56	0.013	-392.6524	-48.91421
	foreign	3595.133	1147.216	3.13	0.003	1307.083	5883.184
	_cons	20242.9	9414.242	2.15	0.035	1466.79	39019.02

The coefficient estimates above are the same as those previously estimated, although the standard error estimates are a little different. `sqreg` obtains estimates of variance by bootstrapping. The important thing here, however, is that the full covariance matrix of the estimators has been estimated and stored, and thus it is now possible to perform hypothesis tests. Are the effects of weight the same at the 25th and 75th percentiles?

```
. test [q25]weight = [q75]weight
( 1) [q25]weight - [q75]weight = 0
      F( 1, 70) = 8.97
      Prob > F = 0.0038
```

It appears that they are not. We can obtain a confidence interval for the difference by using `lincom`:

```
. lincom [q75]weight-[q25]weight
( 1) - [q25]weight + [q75]weight = 0
```

	price	Coef.	Std. Err.	t	P> t	[95% Conf. Interval]	
<hr/>							
	(1)	7.391121	2.467548	3.00	0.004	2.469752	12.31249

Indeed, we could test whether the `weight` and `length` sets of coefficients are equal at the three quantiles estimated:

```
. quietly test [q25]weight = [q50]weight
. quietly test [q25]weight = [q75]weight, accumulate
. quietly test [q25]length = [q50]length, accumulate
. test [q25]length = [q75]length, accumulate
( 1) [q25]weight - [q50]weight = 0
( 2) [q25]weight - [q75]weight = 0
( 3) [q25]length - [q50]length = 0
( 4) [q25]length - [q75]length = 0
      F( 4, 70) = 2.43
      Prob > F = 0.0553
```

`iqreg` focuses on one quantile comparison but presents results that are more easily interpreted:

```
. set seed 1001
. iqreg price weight length foreign, q(.25 .75) reps(100) nolog
.75-.25 Interquantile regression          Number of obs = 74
      bootstrap(100) SEs                  .75 Pseudo R2 = 0.3840
                                          .25 Pseudo R2 = 0.1697
```

price	Coef.	Bootstrap Std. Err.	t	P> t	[95% Conf. Interval]	
weight	7.391121	2.467548	3.00	0.004	2.469752	12.31249
length	-223.6288	83.09868	-2.69	0.009	-389.3639	-57.89376
foreign	1385.208	1193.557	1.16	0.250	-995.2672	3765.683
_cons	22122.68	9009.159	2.46	0.017	4154.478	40090.88

Looking only at the 0.25 and 0.75 quantiles (the interquartile range), the `iqreg` command output is easily interpreted. Increases in `weight` correspond significantly to increases in price dispersion. Increases in `length` correspond to decreases in price dispersion. The `foreign` variable does not significantly change price dispersion.

Do not make too much of these results; the purpose of this example is simply to illustrate the `sqreg` and `iqreg` commands and to do so in a context that suggests why analyzing dispersion might be of interest.

`lincom` after `sqreg` produced the same  $t$  statistic for the interquartile range of `weight`, as did the `iqreg` command above. In general, they will not agree exactly because of the randomness of bootstrapping, unless the random-number seed is set to the same value before estimation (as was done here).

◀

Gould (1997a) presents simulation results showing that the coverage—the actual percentage of confidence intervals containing the true value—for `iqreg` is appropriate.

## What are the parameters?

In this section, we use a specific data-generating process (DGP) to illustrate the interpretation of the parameters estimated by `qreg`. If simulation experiments are not intuitive to you, skip this section.

In general, quantile regression parameterizes the quantiles of the distribution of  $y$  conditional on the independent variables  $\mathbf{x}$  as  $\mathbf{x}\beta$ , where  $\beta$  is a vector of estimated parameters. In our example, we include a constant term and a single independent variable, and we express quantiles of the distribution of  $y$  conditional on  $x$  as  $\beta_0 + \beta_1 x$ .

We use simulated data to illustrate what we mean by a conditional distribution and how to interpret the parameters  $\beta$  estimated by `qreg`. We also note how we could change our example to illustrate a DGP for which the estimator in `qreg` would be misspecified.

We suppose that the distribution of  $y$  conditional on  $x$  has a Weibull form. If  $y$  has a Weibull distribution, the distribution function is  $F(y) = 1 - \exp\{-(y/\lambda)^k\}$ , where the scale parameter  $\lambda > 0$  and the shape parameter  $k > 0$ . We can make  $y$  have a Weibull distribution function conditional on  $x$  by making the scale parameter or the shape parameter functions of  $x$ . In our example, we specify a particular DGP by supposing that  $\lambda = (1 + \alpha x)$ ,  $\alpha = 1.5$ ,  $x = 1 + \sqrt{\nu}$ , and that  $\nu$  has a  $\chi^2(1)$  distribution. For the moment, we leave the parameter  $k$  as is so that we can discuss how this decision relates to model specification.

Plugging in for  $\lambda$  yields the functional form for the distribution of  $y$  conditional on  $x$ , which is known as the conditional distribution function and is denoted  $F(y|x)$ .  $F(y|x)$  is the distribution for  $y$  for each given value of  $x$ .

Some algebra yields that  $F(y|x) = 1 - \exp[-\{y/(1 + \alpha x)\}^k]$ . Letting  $\tau = F(y|x)$  implies that  $0 \leq \tau \leq 1$ , because probabilities must be between 0 and 1.

To obtain the  $\tau$  quantile of the distribution of  $y$  conditional on  $x$ , we solve

$$\tau = 1 - \exp[-\{y/(1 + \alpha x)\}^k]$$

for  $y$  as a function of  $\tau$ ,  $x$ ,  $\alpha$ , and  $k$ . The solution is

$$y = (1 + \alpha x)\{-\ln(1 - \tau)\}^{(1/k)} \tag{1}$$

For any value of  $\tau \in (0, 1)$ , expression (1) gives the  $\tau$  quantile of the distribution of  $y$  conditional on  $x$ . To use `qreg`, we must rewrite (1) as a function of  $x$ ,  $\beta_0$ , and  $\beta_1$ . Some algebra yields that (1) can be rewritten as

$$y = \beta_0 + \beta_1 * x$$

where  $\beta_0 = \{-\ln(1 - \tau)\}^{(1/k)}$  and  $\beta_1 = \alpha\{-\ln(1 - \tau)\}^{(1/k)}$ . We can express the conditional quantiles as linear combinations of  $x$ , which is a property of the estimator implemented in `qreg`.

If we parameterize  $k$  as a nontrivial function of  $x$ , the conditional quantiles will not be linear in  $x$ . If the conditional quantiles cannot be represented as linear functions of  $x$ , we cannot estimate the true parameters of the DGP. This restriction illustrates the limits of the estimator implemented in `qreg`.

We set  $k = 2$  for our example.

Conditional quantile regression allows the coefficients to change with the specified quantile. For our DGP, the coefficients  $\beta_0$  and  $\beta_1$  increase as  $\tau$  gets larger. Substituting in for  $\alpha$  and  $k$  yields that  $\beta_0 = \sqrt{-\ln(1 - \tau)}$  and  $\beta_1 = 1.5\sqrt{-\ln(1 - \tau)}$ . Table 1 presents the true values for  $\beta_0$  and  $\beta_1$  implied by our DGP when  $\tau \in \{0.25, 0.5, 0.8\}$ .



Table 1: True values for  $\beta_0$  and  $\beta_1$ 

$\tau$	$\beta_0$	$\beta_1$
0.25	0.53636	0.80454
0.5	0.8325546	1.248832
0.8	1.268636	1.902954

We can also use (1) to generate data from the specified distribution of  $y$  conditional on  $x$  by plugging in random uniform numbers for  $\tau$ . Each random uniform number substituted in for  $\tau$  in (1) yields a draw from the conditional distribution of  $y$  given  $x$ .

### ▷ Example 5

In this example, we generate 100,000 observations from our specified DGP by substituting random uniform numbers for  $\tau$  in (1), with  $\alpha = 1.5$ ,  $k = 2$ ,  $x = 1 + \sqrt{\nu}$ , and  $\nu$  coming from a  $\chi^2(1)$  distribution.

We begin by executing the code that implements this method; below we discuss each line of the output produced.

```
. clear                                // drop existing variables
. set seed 1234571                      // set random-number seed
. set obs 100000                         // set number of observations
obs was 0, now 100000
. generate double tau = runiform()       // generate uniform variate
. generate double x   = 1 + sqrt(rchi2(1)) // generate values for x
. generate double lambda = 1 + 1.5*x     // lambda is 1 + alpha*x
. generate double k   = 2                // fix value of k
.                                     // generate random values for y
.                                     // given x
. generate double y   = lambda*((-ln(1-tau))^(1/k))
```

Although the comments at the end of each line briefly describe what each line is doing, we provide a more careful description. The first line drops any variables in memory. The second sets the seed of the random-number generator so that we will always get the same sequence of random uniform numbers. The third line sets the sample size to 100,000 observations, and the fourth line reports the change in sample size.

The fifth line substitutes random uniform numbers for  $\tau$ . This line is the key to the algorithm. This standard method, known as inverse-probability transforms, for computing random numbers is discussed by Cameron and Trivedi (2010, 126–127), among others.

Lines 6–8 generate  $x$ ,  $\lambda$ , and  $k$  per our specified DGP. Lines 9–11 implement (1) using the previously generated  $\lambda$ ,  $x$ , and  $k$ .

At the end, we have 100,000 observations on  $y$  and  $x$ , with  $y$  coming from the conditional distribution that we specified above.

◀

### ▷ Example 6

In the example below, we use `qreg` to estimate  $\beta_1$  and  $\beta_0$ , the parameters from the conditional quantile function, for the 0.5 quantile from our simulated data.

```

. qreg y x, quantile(.5)
Iteration 1: WLS sum of weighted deviations = 137951.04
Iteration 1: sum of abs. weighted deviations = 137950.65
Iteration 2: sum of abs. weighted deviations = 137687.92
Iteration 3: sum of abs. weighted deviations = 137259.28
Iteration 4: sum of abs. weighted deviations = 137252.76
Iteration 5: sum of abs. weighted deviations = 137251.32
Iteration 6: sum of abs. weighted deviations = 137251.31
Iteration 7: sum of abs. weighted deviations = 137251.31

Median regression
Raw sum of deviations 147681 (about 2.944248)
Min sum of deviations 137251.3
Number of obs = 100000
Pseudo R2 = 0.0706

```

	y	Coef.	Std. Err.	t	P> t	[95% Conf. Interval]	
	x	1.228536	.0118791	103.42	0.000	1.205253	1.251819
	_cons	.8693355	.0225288	38.59	0.000	.8251793	.9134917

In the `qreg` output, the results for `x` correspond to the estimate of  $\beta_1$ , and the results for `_cons` correspond to the estimate of  $\beta_0$ . The reported estimates are close to their true values of 1.248832 and 0.8325546, which are given in table 1.

The intuition in this example comes from the ability of `qreg` to recover the true parameters of our specified DGP. As we increase the number of observations in our sample size, the `qreg` estimates will get closer to the true values.

◀

## ▷ Example 7

In the example below, we estimate the parameters of the conditional quantile function for the 0.25 quantile and compare them with the true values.

```

. qreg y x, quantile(.25)
Iteration 1: WLS sum of weighted deviations = 130994.87
Iteration 1: sum of abs. weighted deviations = 130984.72
Iteration 2: sum of abs. weighted deviations = 120278.95
Iteration 3: sum of abs. weighted deviations = 99999.586
Iteration 4: sum of abs. weighted deviations = 99998.958
Iteration 5: sum of abs. weighted deviations = 99998.93
Iteration 6: sum of abs. weighted deviations = 99998.93

.25 Quantile regression
Raw sum of deviations 104029.6 (about 1.857329)
Min sum of deviations 99998.93
Number of obs = 100000
Pseudo R2 = 0.0387

```

	y	Coef.	Std. Err.	t	P> t	[95% Conf. Interval]	
	x	.7844305	.0107092	73.25	0.000	.7634405	.8054204
	_cons	.5633285	.0203102	27.74	0.000	.5235209	.6031362

As above, `qreg` reports the estimates of  $\beta_1$  and  $\beta_0$  in the output table for `x` and `_cons`, respectively. The reported estimates are close to their true values of 0.80454 and 0.53636, which are given in table 1. As expected, the estimates are close to their true values. Also as expected, the estimates for the 0.25 quantile are smaller than the estimates for the 0.5 quantile.

◀

## ► Example 8

We finish this section by estimating the parameters of the conditional quantile function for the 0.8 quantile and comparing them with the true values.

```
. qreg y x, quantile(.8)
Iteration 1: WLS sum of weighted deviations = 132664.83
Iteration 1: sum of abs. weighted deviations = 132668.09
Iteration 2: sum of abs. weighted deviations = 120153.29
Iteration 3: sum of abs. weighted deviations = 105178.38
Iteration 4: sum of abs. weighted deviations = 104681.92
Iteration 5: sum of abs. weighted deviations = 104525.01
Iteration 6: sum of abs. weighted deviations = 104498.61
Iteration 7: sum of abs. weighted deviations = 104490.25
Iteration 8: sum of abs. weighted deviations = 104490.21
Iteration 9: sum of abs. weighted deviations = 104490.16
Iteration 10: sum of abs. weighted deviations = 104490.15
Iteration 11: sum of abs. weighted deviations = 104490.15
Iteration 12: sum of abs. weighted deviations = 104490.15
Iteration 13: sum of abs. weighted deviations = 104490.15
Iteration 14: sum of abs. weighted deviations = 104490.15
Iteration 15: sum of abs. weighted deviations = 104490.15

.8 Quantile regression                                Number of obs =    100000
Raw sum of deviations 120186.7 (about 4.7121822)
Min sum of deviations 104490.1                       Pseudo R2      =    0.1306
```

y	Coef.	Std. Err.	t	P> t	[95% Conf. Interval]	
x	1.889702	.0146895	128.64	0.000	1.860911	1.918493
_cons	1.293773	.0278587	46.44	0.000	1.23917	1.348375

As above, `qreg` reports the estimates of  $\beta_1$  and  $\beta_0$  in the output table for `x` and `_cons`, respectively. The reported estimates are close to their true values of 1.902954 and 1.268636, which are given in table 1. As expected, the estimates are close to their true values. Also as expected, the estimates for the 0.8 quantile are larger than the estimates for the 0.5 quantile.

◀

## Saved results

qreg saves the following in `e()`:

### Scalars

<code>e(N)</code>	number of observations
<code>e(df_m)</code>	model degrees of freedom
<code>e(df_r)</code>	residual degrees of freedom
<code>e(q)</code>	quantile requested
<code>e(q_v)</code>	value of the quantile
<code>e(sum_adev)</code>	sum of absolute deviations
<code>e(sum_rdev)</code>	sum of raw deviations
<code>e(f_r)</code>	density estimate
<code>e(sparsity)</code>	sparsity estimate
<code>e(bwidth)</code>	bandwidth
<code>e(kbwidth)</code>	kernel bandwidth
<code>e(rank)</code>	rank of <code>e(V)</code>
<code>e(convcode)</code>	0 if converged; otherwise, return code for why nonconvergence

### Macros

<code>e(cmd)</code>	<code>qreg</code>
<code>e(cmdline)</code>	command as typed
<code>e(depvar)</code>	name of dependent variable
<code>e(bwmethod)</code>	bandwidth method
<code>e(denmethod)</code>	density estimation method
<code>e(kernel)</code>	kernel function, if <code>vce(iid, kernel)</code>
<code>e(wtype)</code>	weight type
<code>e(wexp)</code>	weight expression
<code>e(properties)</code>	<code>b V</code>
<code>e(predict)</code>	program used to implement <code>predict</code>
<code>e(marginsnotok)</code>	predictions disallowed by <code>margins</code>

### Matrices

<code>e(b)</code>	coefficient vector
<code>e(V)</code>	variance–covariance matrix of the estimators

### Functions

<code>e(sample)</code>	marks estimation sample
------------------------	-------------------------

`iqreg` saves the following in `e()`:

Scalars

<code>e(N)</code>	number of observations
<code>e(df_r)</code>	residual degrees of freedom
<code>e(q0)</code>	lower quantile requested
<code>e(q1)</code>	upper quantile requested
<code>e(reps)</code>	number of replications
<code>e(sumrdev0)</code>	lower quantile sum of raw deviations
<code>e(sumrdev1)</code>	upper quantile sum of raw deviations
<code>e(sumadev0)</code>	lower quantile sum of absolute deviations
<code>e(sumadev1)</code>	upper quantile sum of absolute deviations
<code>e(rank)</code>	rank of <code>e(V)</code>
<code>e(convcode)</code>	0 if converged; otherwise, return code for why nonconvergence

Macros

<code>e(cmd)</code>	<code>iqreg</code>
<code>e(cmdline)</code>	command as typed
<code>e(depvar)</code>	name of dependent variable
<code>e(vctype)</code>	title used to label Std. Err.
<code>e(properties)</code>	<code>b V</code>
<code>e(predict)</code>	program used to implement <code>predict</code>
<code>e(marginsnotok)</code>	predictions disallowed by <code>margins</code>

Matrices

<code>e(b)</code>	coefficient vector
<code>e(V)</code>	variance–covariance matrix of the estimators

Functions

<code>e(sample)</code>	marks estimation sample
------------------------	-------------------------

`sqreg` saves the following in `e()`:

Scalars

<code>e(N)</code>	number of observations
<code>e(df_r)</code>	residual degrees of freedom
<code>e(n_q)</code>	number of quantiles requested
<code>e(q#)</code>	the quantiles requested
<code>e(reps)</code>	number of replications
<code>e(sumrdv#)</code>	sum of raw deviations for <code>q#</code>
<code>e(sumadv#)</code>	sum of absolute deviations for <code>q#</code>
<code>e(rank)</code>	rank of <code>e(V)</code>
<code>e(convcode)</code>	0 if converged; otherwise, return code for why nonconvergence

Macros

<code>e(cmd)</code>	<code>sqreg</code>
<code>e(cmdline)</code>	command as typed
<code>e(depvar)</code>	name of dependent variable
<code>e(eqnames)</code>	names of equations
<code>e(vctype)</code>	title used to label Std. Err.
<code>e(properties)</code>	<code>b V</code>
<code>e(predict)</code>	program used to implement <code>predict</code>
<code>e(marginsnotok)</code>	predictions disallowed by <code>margins</code>

Matrices

<code>e(b)</code>	coefficient vector
<code>e(V)</code>	variance–covariance matrix of the estimators

Functions

<code>e(sample)</code>	marks estimation sample
------------------------	-------------------------

`bsqreg` saves the following in `e()`:

Scalars

<code>e(N)</code>	number of observations
<code>e(df_r)</code>	residual degrees of freedom
<code>e(q)</code>	quantile requested
<code>e(q_v)</code>	value of the quantile
<code>e(reps)</code>	number of replications
<code>e(sum_adev)</code>	sum of absolute deviations
<code>e(sum_rdev)</code>	sum of raw deviations
<code>e(rank)</code>	rank of <code>e(V)</code>
<code>e(convcode)</code>	0 if converged; otherwise, return code for why nonconvergence

Macros

<code>e(cmd)</code>	<code>bsqreg</code>
<code>e(cmdline)</code>	command as typed
<code>e(depvar)</code>	name of dependent variable
<code>e(properties)</code>	<code>b V</code>
<code>e(predict)</code>	program used to implement <code>predict</code>
<code>e(marginsnotok)</code>	predictions disallowed by <code>margins</code>

Matrices

<code>e(b)</code>	coefficient vector
<code>e(V)</code>	variance–covariance matrix of the estimators

Functions

<code>e(sample)</code>	marks estimation sample
------------------------	-------------------------

`_qreg` saves the following in `r()`:

Scalars

<code>r(N)</code>	number of observations
<code>r(df_m)</code>	model degrees of freedom
<code>r(q)</code>	quantile requested
<code>r(q_v)</code>	value of the quantile
<code>r(sum_w)</code>	sum of the weights
<code>r(sum_adev)</code>	sum of absolute deviations
<code>r(sum_rdev)</code>	sum of raw deviations
<code>r(f_r)</code>	residual density estimate
<code>r(ic)</code>	number of iterations
<code>r(convcode)</code>	1 if converged, 0 otherwise

## Methods and formulas

`qreg`, `iqreg`, `sqreg`, and `bsqreg` are implemented as ado-files.

Methods and formulas are presented under the following headings:

*Introduction*

*Linear programming formulation of quantile regression*

*Standard errors when residuals are i.i.d.*

*Pseudo- $R^2$*

*Interquantile and simultaneous-quantile regression*

## Introduction

According to Stuart and Ord (1991, 1084), the method of minimum absolute deviations was first proposed by Boscovich in 1757 and was later developed by Laplace; Stigler (1986, 39–55) and Hald (1998, 97–103, 112–116) provide historical details. According to Bloomfield and Steiger (1980), Harris (1950) later observed that the problem of minimum absolute deviations could be turned into the

linear programming problem that was first implemented by Wagner (1959). Interest has grown in this method because robust methods and extreme value modeling have become more popular. Statistical and computational properties of minimum absolute deviation estimators are surveyed by Narula and Wellington (1982). Cameron and Trivedi (2005), Hao and Naiman (2007), and Wooldridge (2010) provide excellent introductions to quantile regression methods, while Koenker (2005) gives an in-depth review of the topic.

## Linear programming formulation of quantile regression

Define  $\tau$  as the quantile to be estimated; the median is  $\tau = 0.5$ . For each observation  $i$ , let  $\varepsilon_i$  be the residual

$$\varepsilon_i = y_i - \mathbf{x}'_i \widehat{\boldsymbol{\beta}}_\tau$$

The objective function to be minimized is

$$\begin{aligned} c_\tau(\varepsilon_i) &= (\tau \mathbf{1}\{\varepsilon_i \geq 0\} + (1 - \tau) \mathbf{1}\{\varepsilon_i < 0\}) |\varepsilon_i| \\ &= (\tau \mathbf{1}\{\varepsilon_i \geq 0\} - (1 - \tau) \mathbf{1}\{\varepsilon_i < 0\}) \varepsilon_i \\ &= (\tau - \mathbf{1}\{\varepsilon_i < 0\}) \varepsilon_i \end{aligned} \quad (2)$$

where  $\mathbf{1}\{\cdot\}$  is the indicator function. This function is sometimes referred to as the check function because it resembles a check mark (Wooldridge 2010, 450); the slope of  $c_\tau(\varepsilon_i)$  is  $\tau$  when  $\varepsilon_i > 0$  and is  $\tau - 1$  when  $\varepsilon_i < 0$ , but is undefined for  $\varepsilon_i = 0$ . Choosing the  $\widehat{\boldsymbol{\beta}}_\tau$  that minimize  $c_\tau(\varepsilon_i)$  is equivalent to finding the  $\widehat{\boldsymbol{\beta}}_\tau$  that make  $\mathbf{x}\widehat{\boldsymbol{\beta}}_\tau$  best fit the quantiles of the distribution of  $y$  conditional on  $\mathbf{x}$ .

This minimization problem is set up as a linear programming problem and is solved with linear programming techniques, as suggested by Armstrong, Frome, and Kung (1979) and described in detail by Koenker (2005). Here  $2n$  slack variables,  $\mathbf{u}_{n \times 1}$  and  $\mathbf{v}_{n \times 1}$ , are introduced, where  $u_i \geq 0$ ,  $v_i \geq 0$ , and  $u_i \times v_i = 0$ , reformulating the problem as

$$\min_{\boldsymbol{\beta}_\tau, \mathbf{u}, \mathbf{v}} \{ \tau \mathbf{1}'_n \mathbf{u} + (1 - \tau) \mathbf{1}'_n \mathbf{v} \mid \mathbf{y} - \mathbf{X}\boldsymbol{\beta}_\tau = \mathbf{u} - \mathbf{v} \}$$

where  $\mathbf{1}_n$  is a vector of 1s. This is a linear objective function on a polyhedral constraint set with  $\binom{n}{k}$  vertices, and our goal is to find the vertex that minimizes (2). Each step in the search is described by a set of  $k$  observations through which the regression plane passes, called the *basis*. A step is taken by replacing a point in the basis if the linear objective function can be improved. If this occurs, a line is printed in the iteration log. The definition of convergence is exact in the sense that no amount of added iterations could improve the objective function.

A series of weighted least-squares (WLS) regressions is used to identify a set of observations as a starting basis. The WLS algorithm for  $\tau = 0.5$  is taken from Schlossmacher (1973) with a generalization for  $0 < \tau < 1$  implied from Hunter and Lange (2000).

## Standard errors when residuals are i.i.d.

The estimator for the VCE implemented in `qreg` assumes that the errors of the model are independent and identically distributed (i.i.d.). When the errors are i.i.d., the large-sample VCE is

$$\text{cov}(\boldsymbol{\beta}_\tau) = \frac{\tau(1 - \tau)}{f_Y^2(\xi_\tau)} \{ \mathbf{E}(\mathbf{x}_i \mathbf{x}'_i) \}^{-1} \quad (3)$$

where  $\xi_\tau = F_Y^{-1}(\tau)$  and  $F_Y(y)$  is the distribution function of  $Y$  with density  $f_Y(y)$ . See Koenker (2005, 73) for this result. From (3), we see that the regression precision depends on the inverse of the density function, termed the sparsity function,  $s_\tau = 1/f_Y(\xi_\tau)$ .

While  $1/n \sum_{i=1}^n \mathbf{x}_i \mathbf{x}_i'$  estimates  $\mathbf{E}(\mathbf{x}_i \mathbf{x}_i')$ , estimating the sparsity function is more difficult. `qreg` provides several methods to estimate the sparsity function. The different estimators are specified through the suboptions of `vce(iid, denmethod bwidth)`. The suboption `denmethod` specifies the functional form for the sparsity estimator. The default is `fitted`.

Here we outline the logic underlying the `fitted` estimator. Because  $F_Y(y)$  is the distribution function for  $Y$ , we have  $f_Y(y) = \{dF_Y(y)\}/dy$ ,  $\tau = F_Y(\xi_\tau)$ , and  $\xi_\tau = F_Y^{-1}(\tau)$ . When differentiating the identity  $F_Y\{F_Y^{-1}(\tau)\} = \tau$ , the sparsity function can be written as  $s_\tau = \{F_Y^{-1}(\tau)\}/dt$ . Numerically, we can approximate the derivative using the centered difference,

$$\frac{F_Y^{-1}(\tau)}{dt} \approx \frac{F_Y^{-1}(\tau + h) - F_Y^{-1}(\tau - h)}{2h} = \frac{\xi_{\tau+h} - \xi_{\tau-h}}{2h} = \widehat{s}_\tau \quad (4)$$

where  $h$  is the bandwidth.

The empirical quantile function is computed by first estimating  $\beta_{\tau+h}$  and  $\beta_{\tau-h}$ , and then computing  $\widehat{F}_Y^{-1}(\tau + h) = \bar{\mathbf{x}}' \widehat{\beta}_{\tau+h}$  and  $\widehat{F}_Y^{-1}(\tau - h) = \bar{\mathbf{x}}' \widehat{\beta}_{\tau-h}$ , where  $\bar{\mathbf{x}}$  is the sample mean of the independent variables  $\mathbf{x}$ . These quantities are then substituted into (4).

Alternatively, as the option suggests, `vce(iid, residual)` specifies that `qreg` use the empirical quantile function of the residuals to estimate the sparsity. Here we substitute  $F_\epsilon$ , the distribution of the residuals, for  $F_Y$ , which only differ by their first moments.

The  $k$  residuals associated with the linear programming basis will be zero, where  $k$  is the number of regression coefficients. These zero residuals are removed before computing the  $\tau + h$  and  $\tau - h$  quantiles,  $\varepsilon_{(\tau+h)} = \widehat{F}_\epsilon^{-1}(\tau + h)$  and  $\varepsilon_{(\tau-h)} = \widehat{F}_\epsilon^{-1}(\tau - h)$ . The  $\widehat{F}_\epsilon^{-1}$  estimates are then substituted for  $F_Y^{-1}$  in (4).

Each of the estimators for the sparsity function depends on a bandwidth. The `vce()` suboption `bwidth` specifies the bandwidth method to use. The three bandwidth options and their citations are `hsheather` (Hall and Sheather 1988), `bofinger` (Bofinger 1975), and `chamberlain` (Chamberlain 1994).

Their formulas are

$$\begin{aligned} h_s &= n^{-1/3} \Phi^{-1} \left( 1 - \frac{\alpha}{2} \right)^{2/3} \left[ \frac{3}{2} \times \frac{\phi\{\Phi^{-1}(\tau)\}^4}{2\Phi^{-1}(\tau)^2 + 1} \right]^{1/3} \\ h_b &= n^{-1/5} \left[ \frac{\frac{9}{2} \phi\{2\Phi^{-1}(\tau)\}^4}{\{2\Phi^{-1}(\tau)^2 + 1\}^2} \right]^{1/5} \\ h_c &= \Phi^{-1} \left( 1 - \frac{\alpha}{2} \right) \sqrt{\frac{\tau(1-\tau)}{n}} \end{aligned}$$

where  $h_s$  is the Hall–Sheather bandwidth,  $h_b$  is the Bofinger bandwidth,  $h_c$  is the Chamberlain bandwidth,  $\Phi()$  and  $\phi()$  are the standard normal distribution and density functions,  $n$  is the sample size, and  $100(1 - \alpha)$  is the confidence level set by the `level()` option. Koenker (2005) discusses the derivation of the Hall–Sheather and the Bofinger bandwidth formulas. You should avoid modifying the confidence level when replaying estimates that use the Hall–Sheather or Chamberlain bandwidths because these methods use the confidence level to estimate the coefficient standard errors.



Finally, the `vce()` suboption `kernel(kernel)` specifies that `qreg` use one of several kernel-density estimators to estimate the sparsity function. `kernel` allows you to choose which kernel function to use, where the default is the Epanechnikov kernel. See [R] `kdensity` for the functional form of the eight kernels.

The kernel bandwidth is computed using an adaptive estimate of scale

$$h_k = \min\left(\hat{\sigma}, \frac{r_q}{1.34}\right) \times \{\Phi^{-1}(\tau + h) - \Phi^{-1}(\tau - h)\}$$

where  $h$  is one of  $h_s$ ,  $h_b$ , or  $h_c$ ;  $r_q$  is the interquartile range; and  $\hat{\sigma}$  is the standard deviation of  $\mathbf{y}$ ; see Silverman (1992, 47) and Koenker (2005, 81) for discussions. Let  $\hat{f}_\epsilon(\varepsilon_i)$  be the kernel density estimate for the  $i$ th residual, and then the kernel estimator for the sparsity function is

$$\hat{s}_\tau = \frac{nh_k}{\sum_{i=1}^n \hat{f}_\epsilon(\varepsilon_i)}$$

Finally, substituting your choice of sparsity estimate into (3) results in the i.i.d. variance–covariance matrix

$$\mathbf{V}_n = \hat{s}_\tau^2 \tau(1 - \tau) \left( \sum_{i=1}^n \mathbf{x}_i \mathbf{x}_i' \right)^{-1}$$

## Pseudo- $R^2$

The pseudo- $R^2$  is calculated as

$$1 - \frac{\text{sum of weighted deviations about estimated quantile}}{\text{sum of weighted deviations about raw quantile}}$$

This is based on the likelihood for a double-exponential distribution  $e^{v_i|\varepsilon_i|}$ , where  $v_i$  are multipliers

$$v_i = \begin{cases} 2\tau & \text{if } \varepsilon_i > 0 \\ 2(1 - \tau) & \text{otherwise} \end{cases}$$

Minimizing the objective function (2) with respect to  $\beta_\tau$  also minimizes  $\sum_i |\varepsilon_i| v_i$ , the sum of weighted least absolute deviations. For example, for the 50th percentile  $v_i = 1$ , for all  $i$ , and we have median regression. If we want to estimate the 75th percentile, we weight the negative residuals by 0.50 and the positive residuals by 1.50. It can be shown that the criterion is minimized when 75% of the residuals are negative.

## References

- Angrist, J. D., and J.-S. Pischke. 2009. *Mostly Harmless Econometrics: An Empiricist's Companion*. Princeton, NJ: Princeton University Press.
- Armstrong, R. D., E. L. Frome, and D. S. Kung. 1979. Algorithm 79-01: A revised simplex algorithm for the absolute deviation curve fitting problem. *Communications in Statistics, Simulation and Computation* 8: 175–190.
- Bloomfield, P., and W. Steiger. 1980. Least absolute deviations curve-fitting. *SIAM Journal on Scientific Computing* 1: 290–301.

- Bofinger, E. 1975. Estimation of a density function using order statistics. *Australian Journal of Statistics* 17: 1–17.
- Cameron, A. C., and P. K. Trivedi. 2005. *Microeconometrics: Methods and Applications*. New York: Cambridge University Press.
- . 2010. *Microeconometrics Using Stata*. Rev. ed. College Station, TX: Stata Press.
- Chamberlain, G. 1994. Quantile regression, censoring, and the structure of wages. In *Advances in Econometrics, Vol. 1: Sixth World Congress*, ed. C. A. Sims, 171–209. Cambridge: Cambridge University Press.
- Frölich, M., and B. Melly. 2010. Estimation of quantile treatment effects with Stata. *Stata Journal* 10: 423–457.
- Gould, W. W. 1997a. crc46: Better numerical derivatives and integrals. *Stata Technical Bulletin* 35: 3–5. Reprinted in *Stata Technical Bulletin Reprints*, vol. 6, pp. 8–12. College Station, TX: Stata Press.
- . 1997b. sg70: Interquantile and simultaneous-quantile regression. *Stata Technical Bulletin* 38: 14–22. Reprinted in *Stata Technical Bulletin Reprints*, vol. 7, pp. 167–176. College Station, TX: Stata Press.
- Gould, W. W., and W. H. Rogers. 1994. Quantile regression as an alternative to robust regression. In *1994 Proceedings of the Statistical Computing Section*. Alexandria, VA: American Statistical Association.
- Hald, A. 1998. *A History of Mathematical Statistics from 1750 to 1930*. New York: Wiley.
- Hall, P., and S. Sheather. 1988. On the distribution of the Studentized quantile. *Journal of the Royal Statistical Society, Series B* 50: 381–391.
- Hao, L., and D. Q. Naiman. 2007. *Quantile Regression*. Thousand Oaks, CA: Sage.
- Harris, T. 1950. Regression using minimum absolute deviations. *American Statistician* 4: 14–15.
- Huber, P. J. 1981. *Robust Statistics*. New York: Wiley.
- Hunter, D. R., and K. Lange. 2000. Quantile regression via an MM algorithm. *Journal of Computational and Graphical Statistics* 9: 60–77.
- Jolliffe, D., B. Krushelnytskyy, and A. Semykina. 2000. sg153: Censored least absolute deviations estimator: CLAD. *Stata Technical Bulletin* 58: 13–16. Reprinted in *Stata Technical Bulletin Reprints*, vol. 10, pp. 240–244. College Station, TX: Stata Press.
- Koenker, R. 2005. *Quantile Regression*. New York: Cambridge University Press.
- Koenker, R., and K. Hallock. 2001. Quantile regression. *Journal of Economic Perspectives* 15: 143–156.
- Narula, S. C., and J. F. Wellington. 1982. The minimum sum of absolute errors regression: A state of the art survey. *International Statistical Review* 50: 317–326.
- Rousseeuw, P. J., and A. M. Leroy. 1987. *Robust Regression and Outlier Detection*. New York: Wiley.
- Schlossmacher, E. J. 1973. An iterative technique for absolute deviations curve fitting. *Journal of the American Statistical Association* 68: 857–859.
- Silverman, B. W. 1992. *Density Estimation for Statistics and Data Analysis*. London: Chapman & Hall.
- Stigler, S. M. 1986. *The History of Statistics: The Measurement of Uncertainty before 1900*. Cambridge, MA: Belknap Press.
- Stuart, A., and J. K. Ord. 1991. *Kendall’s Advanced Theory of Statistics: Distribution Theory, Vol I*. 5th ed. New York: Oxford University Press.
- Wagner, H. M. 1959. Linear programming techniques for regression analysis. *Journal of the American Statistical Association* 54: 206–212.
- Wooldridge, J. M. 2010. *Econometric Analysis of Cross Section and Panel Data*. 2nd ed. Cambridge, MA: MIT Press.

---

## Also see

- [R] **qreg postestimation** — Postestimation tools for qreg, iqreg, sqreg, and bsqreg
- [R] **bootstrap** — Bootstrap sampling and estimation
- [R] **regress** — Linear regression
- [R] **rreg** — Robust regression
- [MI] **estimation** — Estimation commands for use with mi estimate
- [U] **20 Estimation and postestimation commands**