Description

`qreg` fits quantile (including median) regression models, also known as least absolute value, minimum absolute deviation, or minimum L1-norm value. 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.

Quick start

Quantile regression

Median regression of `y` on `x1` and `x2`

```
qreg y x1 x2
```

Add categorical covariate `a` using factor-variable syntax

```
qreg y x1 x2 i.a
```

As above, but with standard errors using a biweight kernel for the nonparametric density estimator

```
qreg y x1 x2 i.a, vce(, kernel(biweight))
```

Quantile regression of the 75th percentile of `y` on `x1`, `x2`, and `a`

```
qreg y x1 x2 i.a, quantile(.75)
```

Interquantile range regression

Difference between the 90th and 10th quantiles of `y` on `x1`, `x2`, and `a` with bootstrap standard errors

```
iqreg y x1 x2 i.a, quantiles(.1 .9)
```
Simultaneous-quantile regression

Simultaneous estimation of quantile regressions for the 10th and 90th quantiles of $y$ with bootstrap standard errors

```
sqreg y x1 x2 i.a, quantiles(.1 .9)
```

As above, but for the 25th, 50th, and 75th quantiles of $y$

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

As above, but increase the number of bootstrap replications to 500

```
sqreg y x1 x2 i.a, quantiles(.25 .5 .75) reps(500)
```

Bootstrapped quantile regression

Single quantile regression for the 25th quantile with bootstrap standard errors

```
bsqreg y x1 x2 i.a, quantile(.25)
```

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
Syntax

Quantile regression

\texttt{qreg \{ depvar \} \{ indepvars \} \{ if \} \{ in \} \{ weight \} \{ , qreg\_options \}}

Interquantile range regression

\texttt{iqreg \{ depvar \} \{ indepvars \} \{ if \} \{ in \} \{ , iqreg\_options \}}

Simultaneous-quantile regression

\texttt{sqreg \{ depvar \} \{ indepvars \} \{ if \} \{ in \} \{ , sqreg\_options \}}

Bootstrapped quantile regression

\texttt{bsqreg \{ depvar \} \{ indepvars \} \{ if \} \{ in \} \{ , bsqreg\_options \}}

\texttt{qreg\_options}

\begin{tabular}{ll}
\hline
\textbf{Model} & \textbf{Description} \\
\hline
\texttt{quantile(#)\hspace{1cm}} & \texttt{estimate # quantile; default is quantile(.5)} \\
\hline
\texttt{vce([vcetype], [vceopts])\hspace{1cm}} & \texttt{technique used to estimate standard errors} \\
\hline
\texttt{level(#)\hspace{1cm}} & \texttt{set confidence level; default is level(95)} \\
\texttt{display\_options\hspace{1cm}} & \texttt{control columns and column formats, row spacing, line width, display of omitted variables and base and empty cells, and factor-variable labeling} \\
\hline
\texttt{optimization\_options\hspace{1cm}} & \texttt{control the optimization process; seldom used} \\
\texttt{wlsiter(#)} & \texttt{attempt # weighted least-squares iterations before doing linear programming iterations} \\
\hline
\texttt{vcetype\hspace{1cm}} & \texttt{Description} \\
\hline
\texttt{iid\hspace{1cm}} & \texttt{compute the VCE assuming the residuals are i.i.d.} \\
\texttt{robust\hspace{1cm}} & \texttt{compute the robust VCE} \\
\hline
\end{tabular}
vceopts

**denmethod**
nonparametric density estimation technique

**bwidth**
bandwidth method used by the density estimator

---

**denmethod**

- **fitted**
  use the empirical quantile function using fitted values; the default
- **residual**
  use the empirical residual quantile function
- **kernel (kernel)**
  use a nonparametric kernel density estimator; default is `epanechnikov`

---

**bwidth**

- **hsheather**
  Hall–Sheather’s bandwidth; the default
- **bofinger**
  Bofinger’s bandwidth
- **chamberlain**
  Chamberlain’s bandwidth

---

**kernel**

- **epanechnikov**
  Epanechnikov kernel function; the default
- **epan2**
  alternative Epanechnikov kernel function
- **biweight**
  biweight kernel function
- **cosine**
  cosine trace kernel function
- **gaussian**
  Gaussian kernel function
- **parzen**
  Parzen kernel function
- **rectangle**
  rectangle kernel function
- **triangle**
  triangle kernel function

---

**iqreg_options**

- **quantiles(#) #)**
  interquantile range; default is `quantiles(.25 .75)`
- **reps(#)**
  perform # bootstrap replication replications; default is `reps(20)`

**Reporting**

- **level(#)**
  set confidence level; default is `level(95)`
- **nodots**
  suppress display of the replication dots
- **display_options**
  control columns and column formats, row spacing, line width, display of omitted variables and base and empty cells, and factor-variable labeling
Model

**quantiles(#[[#...]])** estimate # quantiles; default is quantiles(.5)

**reps(#)** perform # bootstrap replications; default is reps(20)

**level(#)** set confidence level; default is level(95)

**nodots** suppress display of the replication dots

**display_options** control columns and column formats, row spacing, line width, display of omitted variables and base and empty cells, and factor-variable labeling

**bsqreg_options**

Model

**quantile(#)** estimate # quantile; default is quantile(.5)

**reps(#)** perform # bootstrap replications; default is reps(20)

**level(#)** set confidence level; default is level(95)

**display_options** control columns and column formats, row spacing, line width, display of omitted variables and base and empty cells, and factor-variable labeling

**indepvars** may contain factor variables; see [U] 11.4.3 Factor variables.

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

qreg allows fweights, iweights, and pweights; see [U] 11.1.6 weight.

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

### Options for qreg

**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.

**vce([vcetype],[vceopts])** specifies the type of VCE to compute and the density estimation method to use in computing the VCE.

**vcetype** specifies the type of VCE to compute. Available types are iid and robust.

- **vce(iid)**, the default, computes the VCE under the assumption that the residuals are independent and identically distributed (i.i.d.).

- **vce(robust)** computes the robust VCE under the assumption that the residual density is continuous and bounded away from 0 and infinity at the specified quantile(); see Koenker (2005, sec. 4.2).
vceopts consists of available *denmethod* and *bwidth* options.

*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. *vce(robust, residual)* is not allowed.

*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* for the Hall–Sheather bandwidth, *bofinger* for the Bofinger bandwidth, and *chamberlain* for the Chamberlain bandwidth.

See Koenker (2005, sec. 3.4 and 4.10) for a description of the sparsity estimation techniques and the Hall–Sheather and Bofinger bandwidth formulas. See Chamberlain (1994, eq. 2.2) for the Chamberlain bandwidth.

---

**Options for iqreg**

*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 interquantile 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.
Options for \texttt{sqreg}

\begin{verbatim}
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.

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

Options for \texttt{bsqreg}

\begin{verbatim}
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.

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

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 (2022, chap. 15) provide a nice introduction to quantile regression using Stata.

Example 1: Estimating the conditional median

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

```
. use https://www.stata-press.com/data/r17/twogrp
. list
```

<table>
<thead>
<tr>
<th></th>
<th>x</th>
<th>y</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2.</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>3.</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>4.</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>5.</td>
<td>0</td>
<td>95</td>
</tr>
<tr>
<td>6.</td>
<td>1</td>
<td>14</td>
</tr>
<tr>
<td>7.</td>
<td>1</td>
<td>19</td>
</tr>
<tr>
<td>8.</td>
<td>1</td>
<td>20</td>
</tr>
<tr>
<td>9.</td>
<td>1</td>
<td>22</td>
</tr>
<tr>
<td>10.</td>
<td>1</td>
<td>23</td>
</tr>
</tbody>
</table>

```
. qreg y x
```

```
Iteration 1: WLS sum of weighted deviations = 60.941342
Iteration 1: sum of abs. weighted deviations = 55.5
Iteration 2: sum of abs. weighted deviations = 55

Median regression

|      | Coefficient | 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 78.5; that is, the sum of \( |y - 14| \) is 78.5. 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 78.5. 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 55. The pseudo-\( R^2 \) is calculated as \( 1 - 55/78.5 \approx 0.2994 \). This result is based on the idea that the median regression is the maximum likelihood estimate for the double-exponential distribution.

Technical note

\texttt{qreg} is an alternative to regular regression or robust regression—see \texttt{[R] regress} and \texttt{[R] rreg}. Let’s compare the results:

\begin{verbatim}
. regress y x

Source | SS      df | MS
------ | -------- | ---
Model  | 2.5      | 1  | 2.5
Residual | 6978.4   | 8  | 872.3
Total  | 6980.9   | 9  | 775.655556

Number of obs = 10
F(1, 8) = 0.00
Prob > F = 0.9586
R-squared = 0.0004
Adj R-squared = -0.1246
Root MSE = 29.535

y | Coefficient 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       

\end{verbatim}

Unlike \texttt{qreg}, \texttt{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 = .1769779
Huber iteration 3: maximum difference in weights = .03149585
Biweight iteration 4: maximum difference in weights = .1979335
Biweight iteration 5: maximum difference in weights = .2332905
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

|         | Coefficient | 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: Median regression

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 https://www.stata-press.com/data/r17/auto, clear
(1978 automobile data)
. qreg price weight length foreign
Iteration 1: WLS sum of weighted deviations = 56397.829
Iteration 1: sum of abs. weighted deviations = 55950.5
Iteration 2: sum of abs. weighted deviations = 55264.718
Iteration 3: sum of abs. weighted deviations = 54762.283
Iteration 4: sum of abs. weighted deviations = 54734.152
Iteration 5: sum of abs. weighted deviations = 54552.638
note: alternate solutions exist.
Iteration 6: sum of abs. weighted deviations = 54465.511
Iteration 7: sum of abs. weighted deviations = 54443.699
Iteration 8: sum of abs. weighted deviations = 54411.294
Median regression Number of obs = 74
Raw sum of deviations 71102.5 (about 4934)
Min sum of deviations 54411.29 Pseudo R2 = 0.2347

|        | Coefficient | 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: Estimating quantiles other than the median

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

```stata
use https://www.stata-press.com/data/r17/auto
qreg price weight length foreign, quantile(.25)
```

The output shows the iterative process of minimizing the sum of absolute weighted deviations until convergence.

```
.25 Quantile regression
Number of obs = 74
Raw sum of deviations 41912.75 (about 4187)
Min sum of deviations 34801.78  Pseudo R2 = 0.1697

|        | Coefficient | Std. err. | t    | P>|t|   | [95% conf. interval] |
|--------|-------------|-----------|------|-------|----------------------|
| price  |             |           |      |       |                      |
| 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 = 55466.741
Iteration 1: sum of abs. weighted deviations = 55652.957
Iteration 2: sum of abs. weighted deviations = 52994.785
Iteration 3: sum of abs. weighted deviations = 50189.446
Iteration 4: sum of abs. weighted deviations = 49898.245
Iteration 5: sum of abs. weighted deviations = 49398.106
Iteration 6: sum of abs. weighted deviations = 49241.835
Iteration 7: sum of abs. weighted deviations = 49197.967

.75 Quantile regression
Number of obs = 74

Raw sum of deviations 79860.75 (about 6342)
Min sum of deviations 49197.97 Pseudo R2 = 0.3840

|         | Coefficient | Std. err. | t   | P>|t| | [95% conf. interval] |
|---------|-------------|-----------|-----|-----|---------------------|
| price   |             |           |     |     |                     |
| 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
Number of observations (_N) was 0, now 10,000.
set seed 50550
generate x = .1 + .9 * runiform()
generate 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 = 555.44181
Iteration 1: sum of abs. weighted deviations = 555.25622
Iteration 2: sum of abs. weighted deviations = 115.02628
Iteration 3: sum of abs. weighted deviations = 89.617883
Iteration 4: sum of abs. weighted deviations = 89.61679

.05 Quantile regression
Number of obs = 10,000
Raw sum of deviations 89.68001 (about .00105493)
Min sum of deviations 89.61679 Pseudo R2 = 0.0007

|   y | Coefficient | Std. err. | t     | P>|t| | [95% conf. interval] |
|-----|-------------|-----------|-------|-----|---------------------|
| x   | .0028667    | .0004395  | 6.52  | 0.000 | .0020052 .0037283   |
| _cons | -.0001135 | .0002661  | -0.43 | 0.670 | -.0006352 .0004081  |
```

```
. qreg y x, quantile(.95)
Iteration 1: WLS sum of weighted deviations = 624.91903
Iteration 1: sum of abs. weighted deviations = 621.88928
Iteration 2: sum of abs. weighted deviations = 182.03243
Iteration 3: sum of abs. weighted deviations = 170.42588
Iteration 4: sum of abs. weighted deviations = 169.05915
Iteration 5: sum of abs. weighted deviations = 169.05911

.95 Quantile regression
Number of obs = 10,000
Raw sum of deviations 275.9779 (about .60579139)
Min sum of deviations 169.0591 Pseudo R2 = 0.3874

|   y | Coefficient | Std. err. | t     | P>|t| | [95% conf. interval] |
|-----|-------------|-----------|-------|-----|---------------------|
| x   | .9010814    | .008758   | 102.89| 0.000 | .883914 .9182488    |
| _cons | -.0004053 | .0053028  | -0.08 | 0.939 | -.0107999 .0099893  |
```

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.

**Example 4: Obtaining robust standard errors**

Example 2 of qreg on real data above was a median regression of price on weight, length, and foreign using auto.dta. Suppose, after investigation, we are convinced that car price observations are not independent. We decide that standard errors robust to non-i.i.d. errors would be appropriate and use the option vce(robust).

```
. use https://www.stata-press.com/data/r17/auto, clear
   (1978 automobile data)
. qreg price weight length foreign, vce(robust)
Iteration 1: WLS sum of weighted deviations = 56397.829
Iteration 1: sum of abs. weighted deviations = 55950.5
Iteration 2: sum of abs. weighted deviations = 55264.718
Iteration 3: sum of abs. weighted deviations = 54762.283
Iteration 4: sum of abs. weighted deviations = 54734.152
Iteration 5: sum of abs. weighted deviations = 54552.638
    note: alternate solutions exist.
Iteration 6: sum of abs. weighted deviations = 54465.511
Iteration 7: sum of abs. weighted deviations = 54443.699
Iteration 8: sum of abs. weighted deviations = 54411.294
Median regression Number of obs = 74
Raw sum of deviations 71102.5 (about 4934)
Min sum of deviations 54411.29 Pseudo R2 = 0.2347

| price         | Coefficient | Robust std. err. | t    | P>|t|   | [95% conf. interval] |
|---------------|-------------|------------------|------|-------|----------------------|
| weight        | 3.933588    | 1.694477         | 2.32 | 0.023 | 0.55406              | 7.313116  |
| length        | -41.25191   | 51.73571         | -0.80| 0.428 | -144.4355            | 61.93171  |
| foreign       | 3377.771    | 728.5115         | 4.64 | 0.000 | 1924.801             | 4830.741  |
| _cons         | 344.6489    | 5096.528         | 0.07 | 0.946 | -9820.055            | 10509.35  |
```

We see that the robust standard error for weight increases, making it less significant in modifying the median automobile price. The standard error for length also increases, but the standard error for the foreign indicator decreases.
For comparison, we repeat the estimation using bootstrap standard errors:

```
. use https://www.stata-press.com/data/r17/auto, clear
   (1978 automobile data)
. set seed 1001
. bsqreg price weight length foreign
   (fitting base model)
Bootstrap replications (20)
   1 2 3 4 5
............................
Median regression, bootstrap(20) SEs Number of obs = 74
   Raw sum of deviations  71102.5 (about 4934)
   Min sum of deviations  54411.29 Pseudo R2 = 0.2347

|     | Coefficient | Std. err. | t     | P>|t|  | [95% conf. interval] |
|-----|-------------|-----------|-------|------|---------------------|
| price |            |           |       |      |                     |
| weight| 3.933588    | 2.941839  | 1.34  | 0.186| -1.933726          | 9.800901 |
| length| -41.25191   | 73.47105  | -0.56 | 0.576| -187.7853          | 105.2815 |
| foreign| 3377.771    | 1352.518  | 2.50  | 0.015| 680.2582           | 6075.284 |
| _cons | 344.6489    | 5927.045  | 0.06  | 0.954| -11476.47          | 12165.77 |
```

The coefficient estimates are the same—indeed, they are obtained using the same technique. Only the standard errors differ. Therefore, the $t$ statistics, significance levels, and confidence intervals also differ.

Because `bsqreg` (as well as `sqreg` and `iqreg`) obtains standard errors by randomly resampling the data, the standard errors it produces will not be the same from run to run unless we first set the random-number seed to the same number; see [R] `set seed`. 
By default, `bsqreg`, `sqreg`, and `iqreg` use 20 replications. We can control the number of replications by specifying the `reps()` option:

```
. bsqreg price weight length i.foreign, reps(1000)
(fitting base model)
```

**Bootstrap replications (1,000)**

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>..................................................</td>
<td>50</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>..................................................</td>
<td>100</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>..................................................</td>
<td>150</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>..................................................</td>
<td>200</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>..................................................</td>
<td>250</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>..................................................</td>
<td>300</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>..................................................</td>
<td>350</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>..................................................</td>
<td>400</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>..................................................</td>
<td>450</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>..................................................</td>
<td>500</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>..................................................</td>
<td>550</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>..................................................</td>
<td>600</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>..................................................</td>
<td>650</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>..................................................</td>
<td>700</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>..................................................</td>
<td>750</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>..................................................</td>
<td>800</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>..................................................</td>
<td>850</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>..................................................</td>
<td>900</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>..................................................</td>
<td>950</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>..................................................</td>
<td>1,000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Median regression, bootstrap(1000) SEs

<table>
<thead>
<tr>
<th>Number of obs = 74</th>
</tr>
</thead>
<tbody>
<tr>
<td>Raw sum of deviations</td>
</tr>
<tr>
<td>Min sum of deviations</td>
</tr>
<tr>
<td>Pseudo R2 = 0.2347</td>
</tr>
</tbody>
</table>

| price | Coefficient | Std. err. | t | P>|t| | [95% conf. interval] |
|-------|-------------|-----------|---|-------|---------------------|
| weight | 3.933588 | 2.58771 | 1.52 | 0.133 | -1.227437 to 9.094613 |
| length | -41.25191 | 68.02626 | -0.61 | 0.546 | -176.926 to 94.42219 |

<table>
<thead>
<tr>
<th>Variable</th>
<th>qreg</th>
<th>qreg vce(robust)</th>
<th>bsqreg reps(20)</th>
<th>bsqreg reps(1000)</th>
</tr>
</thead>
<tbody>
<tr>
<td>weight</td>
<td>1.329</td>
<td>1.694</td>
<td>2.942</td>
<td>2.588</td>
</tr>
<tr>
<td>length</td>
<td>45.46</td>
<td>51.74</td>
<td>73.47</td>
<td>68.03</td>
</tr>
<tr>
<td>1.foreign</td>
<td>885.4</td>
<td>728.5</td>
<td>1353.</td>
<td>1071.</td>
</tr>
<tr>
<td>_cons</td>
<td>5182.</td>
<td>5097.</td>
<td>5927.</td>
<td>5863.</td>
</tr>
</tbody>
</table>

A comparison of the standard errors is informative.

The results shown above are typical for models with heteroskedastic errors. (Our dependent variable is `price`; if our model had been in terms of `ln(price)`, the standard errors estimated by `qreg` and `bsqreg` would have been nearly identical.) Also, even for heteroskedastic errors, 20 replications is generally sufficient for hypothesis tests against 0.
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

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

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

\texttt{sqreg} is like \texttt{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 \texttt{qreg}. \texttt{sqreg} differs from \texttt{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

\begin{verbatim}
. qreg y x1 x2, quantile(.25)
. qreg y x1 x2, quantile(.75)
\end{verbatim}

By 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

\begin{verbatim}
. iqreg y x1 x2, quantiles(.25 .75)
\end{verbatim}

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:

\begin{verbatim}
. sqreg y x1 x2, quantiles(.25 .75)
. test [q25]x1 = [q75]x1
\end{verbatim}

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

Finally, \texttt{sqreg} can estimate quantiles singly,

\begin{verbatim}
. sqreg y x1 x2, quantiles(.5)
\end{verbatim}

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

\begin{verbatim}
. sqreg y x1 x2, quantiles(.25 .5 .75)
\end{verbatim}
Example 5: Simultaneous quantile estimation

In demonstrating qreg, we performed quantile regressions using auto.dta. 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:

<table>
<thead>
<tr>
<th>Variable</th>
<th>25th percentile</th>
<th>50th percentile</th>
<th>75th percentile</th>
</tr>
</thead>
<tbody>
<tr>
<td>weight</td>
<td>1.83</td>
<td>3.93</td>
<td>9.22</td>
</tr>
<tr>
<td>length</td>
<td>2.85</td>
<td>-41.25</td>
<td>-220.8</td>
</tr>
<tr>
<td>foreign</td>
<td>2209.9</td>
<td>3377.8</td>
<td>3595.1</td>
</tr>
<tr>
<td>_cons</td>
<td>-1879.8</td>
<td>344.6</td>
<td>20242.9</td>
</tr>
</tbody>
</table>

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 https://www.stata-press.com/data/r17/auto, clear
(1978 automobile data)
. set seed 1001
. sqreg price weight length foreign, q(.25 .5 .75) reps(100)
(fitting base model)
```

```
Bootstrap replications (100)
.................................................. 50
.................................................. 100
Simultaneous quantile regression

Bootstrap price Coefficient std. err. t P>|t| [95% conf. interval]

q25

<table>
<thead>
<tr>
<th>Variable</th>
<th>weight</th>
<th>1.831789</th>
<th>1.206151</th>
<th>1.52</th>
<th>0.133</th>
<th>-.573803</th>
<th>4.237381</th>
</tr>
</thead>
<tbody>
<tr>
<td>length</td>
<td>2.84556</td>
<td>26.775</td>
<td>0.11</td>
<td>0.916</td>
<td>-50.55549</td>
<td>56.24661</td>
<td></td>
</tr>
<tr>
<td>foreign</td>
<td>2209.925</td>
<td>946.9585</td>
<td>2.33</td>
<td>0.022</td>
<td>321.2762</td>
<td>4098.575</td>
<td></td>
</tr>
<tr>
<td>_cons</td>
<td>-1879.775</td>
<td>3243.182</td>
<td>-0.58</td>
<td>0.564</td>
<td>-8348.097</td>
<td>4588.548</td>
<td></td>
</tr>
</tbody>
</table>

q50

<table>
<thead>
<tr>
<th>Variable</th>
<th>weight</th>
<th>3.933588</th>
<th>2.34028</th>
<th>1.68</th>
<th>0.097</th>
<th>-.7339531</th>
<th>8.601129</th>
</tr>
</thead>
<tbody>
<tr>
<td>length</td>
<td>-41.25191</td>
<td>59.58361</td>
<td>-0.69</td>
<td>0.491</td>
<td>-160.0877</td>
<td>77.58386</td>
<td></td>
</tr>
<tr>
<td>foreign</td>
<td>3377.771</td>
<td>1081.475</td>
<td>3.12</td>
<td>0.003</td>
<td>1220.836</td>
<td>5534.706</td>
<td></td>
</tr>
<tr>
<td>_cons</td>
<td>344.6489</td>
<td>5062.804</td>
<td>0.07</td>
<td>0.946</td>
<td>-9752.795</td>
<td>10442.09</td>
<td></td>
</tr>
</tbody>
</table>

q75

<table>
<thead>
<tr>
<th>Variable</th>
<th>weight</th>
<th>9.22291</th>
<th>2.631084</th>
<th>3.51</th>
<th>0.001</th>
<th>3.975378</th>
<th>14.47044</th>
</tr>
</thead>
<tbody>
<tr>
<td>length</td>
<td>-220.7833</td>
<td>39.54754</td>
<td>-2.47</td>
<td>0.016</td>
<td>-399.3802</td>
<td>-42.18635</td>
<td></td>
</tr>
<tr>
<td>foreign</td>
<td>3595.133</td>
<td>1208.769</td>
<td>2.97</td>
<td>0.004</td>
<td>1184.319</td>
<td>6005.947</td>
<td></td>
</tr>
<tr>
<td>_cons</td>
<td>20242.9</td>
<td>9682.24</td>
<td>2.09</td>
<td>0.040</td>
<td>932.2846</td>
<td>39553.52</td>
<td></td>
</tr>
</tbody>
</table>
```

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?
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 | Coefficient Std. err.    t  P>|t|     [95% conf. interval]
-----------|----------------------|--|--|-----------|---------------------|
          (1)          7.391121       2.524626   2.93  0.005   2.355914    12.42633
```

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) =  2.25
    Prob > F =  0.0727
```

`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 | Coefficient std. err.    t  P>|t|     [95% conf. interval]
-----------|----------------------|--|--|-----------|---------------------|
    weight  7.391121       2.524626   2.93  0.005   2.355914    12.42633
    length -223.6288       84.21504  -2.66  0.010  -391.5904   -55.66724
   foreign 1385.208       1321.832   1.05  0.298  -1251.103   4021.519
     _cons  22122.68      9329.178   2.37  0.020   3516.219   40729.14
```

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 \( x \) as \( x \beta \), 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

\[
F(y; \lambda, k) = 1 - \exp\left\{ -\left(\frac{y}{\lambda}\right)^k \right\},
\]

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

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

(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>
<thead>
<tr>
<th>$\tau$</th>
<th>$\beta_0$</th>
<th>$\beta_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.25</td>
<td>0.53636</td>
<td>0.80454</td>
</tr>
<tr>
<td>0.5</td>
<td>0.8325546</td>
<td>1.248832</td>
</tr>
<tr>
<td>0.8</td>
<td>1.268636</td>
<td>1.902954</td>
</tr>
</tbody>
</table>

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 6**

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
Number of observations (_N) was 0, now 100,000.

. 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 (2022, 220–221), 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 7

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 = 68573.243
Iteration 1: sum of abs. weighted deviations = 68571.918
Iteration 2: sum of abs. weighted deviations = 68308.342
Iteration 3: sum of abs. weighted deviations = 68241.17
Iteration 4: sum of abs. weighted deviations = 68232.043
Iteration 5: sum of abs. weighted deviations = 68230.304
Iteration 6: sum of abs. weighted deviations = 68229.643
Iteration 7: sum of abs. weighted deviations = 68229.532
Iteration 8: sum of abs. weighted deviations = 68229.514
Iteration 9: sum of abs. weighted deviations = 68229.508
Iteration 10: sum of abs. weighted deviations = 68229.506
Iteration 11: sum of abs. weighted deviations = 68229.505
Median regression  Number of obs = 100,000
Raw sum of deviations 73861.64 (about 2.9443724)
Min sum of deviations 68229.51 Pseudo R2 = 0.0763

    y       Coefficient  Std. err.     t    P>|t|     [95% conf. interval]
-------------  -------------  --------  ------  --------------  -------------
     x       1.266062     .0117759   107.51    0.000    1.242981    1.289143
 _cons     .8083315     .0222972    36.25    0.000     .7646291    .8520338
```

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 8

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 = 65395.359
Iteration 1:  sum of abs. weighted deviations = 65397.892
Iteration 2:  sum of abs. weighted deviations = 52640.481
Iteration 3:  sum of abs. weighted deviations = 50706.508
Iteration 4:  sum of abs. weighted deviations = 49767.356
Iteration 5:  sum of abs. weighted deviations = 49766.98
Iteration 6:  sum of abs. weighted deviations = 49765.818
Iteration 7:  sum of abs. weighted deviations = 49765.589
Iteration 8:  sum of abs. weighted deviations = 49765.549
Iteration 9:  sum of abs. weighted deviations = 49765.533
Iteration 10: sum of abs. weighted deviations = 49765.528
Iteration 11: sum of abs. weighted deviations = 49765.527
Iteration 12: sum of abs. weighted deviations = 49765.527
Iteration 13: sum of abs. weighted deviations = 49765.527

.25 Quantile regression  Number of obs =  100,000
Raw sum of deviations 51945.91 (about 1.8560913)
Min sum of deviations 49765.53  Pseudo R2 = 0.0420

|     | Coefficient | Std. err. | t    | P>|t| | [95% conf. interval] |
|-----|-------------|-----------|------|------|----------------------|
| x   | .8207143    | .0106425  | 77.12| 0.000| .799855 .8415735     |
| _cons | .5075988   | .0201512  | 25.19| 0.000| .4681026 .547095     |
```

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 9

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 = 66126.751
Iteration 1:  sum of abs. weighted deviations = 66130.001
Iteration 2:  sum of abs. weighted deviations = 55084.287
Iteration 3:  sum of abs. weighted deviations = 52914.276
Iteration 4:  sum of abs. weighted deviations = 52101.59
Iteration 5:  sum of abs. weighted deviations = 51899.426
Iteration 6:  sum of abs. weighted deviations = 51898.269
Iteration 7:  sum of abs. weighted deviations = 51898.268
Iteration 8:  sum of abs. weighted deviations = 51898.267
```

```
.8 Quantile regression
Number of obs = 100,000
Raw sum of deviations 60129.76 (about 4.7060381)
Min sum of deviations 51898.27 Pseudo R2 = 0.1369
```

| Coefficient Std. err. t P>|t| [95% conf. interval] |
|---|---|---|---|---|
| x | 1.911771 | .014834 | 128.88 | 0.000 | 1.882697 | 1.940846 |
| _cons | 1.254583 | .0280877 | 44.67 | 0.000 | 1.199531 | 1.309634 |

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.
qreg stores the following in e():

Scalars:

- e(N): number of observations
- e(df_m): model degrees of freedom
- e(df_r): residual degrees of freedom
- e(q): quantile requested
- e(q_v): value of the quantile
- e(r2_p): pseudo-$R^2$
- e(sum_adev): sum of absolute deviations
- e(sum_rdev): sum of raw deviations
- e(sum_w): sum of weights
- e(f_r): density estimate
- e(sparcity): sparsity estimate
- e(bwidth): bandwidth
- e(kbwidth): kernel bandwidth
- e(rank): rank of e(V)
- e(convcode): 0 if converged; otherwise, return code for why nonconvergence

Macros:

- e(cmd): qreg
- e(cmdline): command as typed
- e(depvar): name of dependent variable
- e(bwmethod): bandwidth method; hsheather, bofinger, or chamberlain
- e(demmethod): density estimation method; fitted, residual, or kernel
- e(kernel): kernel function
- e(wtype): weight type
- e(wexp): weight expression
- e(vcetype): vcetype specified in vce()
- e(properties): b V
- e(predict): program used to implement predict
- e(marginsnotok): predictions disallowed by margins
- e(asbalanced): factor variables fvset as asbalanced
- e(asobserved): factor variables fvset as asobserved

Matrices:

- e(b): coefficient vector
- e(V): variance–covariance matrix of the estimators

Functions:

- e(sample): marks estimation sample

In addition to the above, the following is stored in r():

Matrices:

- r(table): matrix containing the coefficients with their standard errors, test statistics, p-values, and confidence intervals

Note that results stored in r() are updated when the command is replayed and will be replaced when any r-class command is run after the estimation command.
iqreg stores the following in e():

Scalars
- `e(N)` number of observations
- `e(df_r)` residual degrees of freedom
- `e(q0)` lower quantile requested
- `e(q1)` upper quantile requested
- `e(reps)` number of replications
- `e(r2_p_q0)` lower quantile pseudo-$R^2$
- `e(r2_p_q1)` upper quantile pseudo-$R^2$
- `e(sumrdev0)` lower quantile sum of raw deviations
- `e(sumrdev1)` upper quantile sum of raw deviations
- `e(sumadev0)` lower quantile sum of absolute deviations
- `e(sumadev1)` upper quantile sum of absolute deviations
- `e(rank)` rank of `e(V)`
- `e(convcode)` 0 if converged; otherwise, return code for why nonconvergence

Macros
- `e(cmd)` iqreg
- `e(cmdline)` command as typed
- `e(depvar)` name of dependent variable
- `e(vcetype)` title used to label Std. err.
- `e(properties)` b V
- `e(predict)` program used to implement predict
- `e(marginsnotok)` predictions disallowed by margins
- `e(asbalanced)` factor variables fvset as asbalanced
- `e(asobserved)` factor variables fvset as asobserved

Matrices
- `e(b)` coefficient vector
- `e(V)` variance–covariance matrix of the estimators

Functions
- `e(sample)` marks estimation sample

In addition to the above, the following is stored in `r()`:

Matrices
- `r(table)` matrix containing the coefficients with their standard errors, test statistics, $p$-values, and confidence intervals

Note that results stored in `r()` are updated when the command is replayed and will be replaced when any r-class command is run after the estimation command.

sqreg stores the following in e():

Scalars
- `e(N)` number of observations
- `e(df_r)` residual degrees of freedom
- `e(n_q)` number of quantiles requested
- `e(q#)` the quantiles requested
- `e(reps)` number of replications
- `e(r2_p_q#)` pseudo-$R^2$ for q#
- `e(sumrdev#)` sum of raw deviations for q#
- `e(sumadev#)` sum of absolute deviations for q#
- `e(rank)` rank of `e(V)`
- `e(convcode)` 0 if converged; otherwise, return code for why nonconvergence

Macros
- `e(cmd)` sqreg
- `e(cmdline)` command as typed
- `e(depvar)` name of dependent variable
- `e(eqnames)` names of equations
- `e(vcetype)` title used to label Std. err.
- `e(properties)` b V
- `e(predict)` program used to implement predict
Prediction disallowed by margins

Factor variables fvset as asbalanced

Factor variables fvset as asobserved

Matrices

- \( e(b) \): coefficient vector
- \( e(V) \): variance–covariance matrix of the estimators

Functions

- \( e(sample) \): marks estimation sample

In addition to the above, the following is stored in \( r() \):

Matrices

- \( r(table) \): matrix containing the coefficients with their standard errors, test statistics, \( p \)-values, and confidence intervals

Note that results stored in \( r() \) are updated when the command is replayed and will be replaced when any \( r \)-class command is run after the estimation command.

bsqreg stores the following in \( e() \):

Scalars

- \( e(N) \): number of observations
- \( e(df_r) \): residual degrees of freedom
- \( e(q) \): quantile requested
- \( e(q_v) \): value of the quantile
- \( e(reps) \): number of replications
- \( e(r2_p) \): pseudo-\( R^2 \)
- \( e(sum_adev) \): sum of absolute deviations
- \( e(sum_rdev) \): sum of raw deviations
- \( e(rank) \): rank of \( e(V) \)
- \( e(convcode) \): 0 if converged; otherwise, return code for why nonconvergence

Macros

- \( e(cmd) \): bsqreg
- \( e(cmdline) \): command as typed
- \( e(depvar) \): name of dependent variable
- \( e(properties) \): b V
- \( e(predict) \): program used to implement predict
- \( e(marginsnotok) \): predictions disallowed by margins
- \( e(asbalanced) \): factor variables fvset as asbalanced
- \( e(asobserved) \): factor variables fvset as asobserved

Matrices

- \( e(b) \): coefficient vector
- \( e(V) \): variance–covariance matrix of the estimators

Functions

- \( e(sample) \): marks estimation sample

In addition to the above, the following is stored in \( r() \):

Matrices

- \( r(table) \): matrix containing the coefficients with their standard errors, test statistics, \( p \)-values, and confidence intervals

Note that results stored in \( r() \) are updated when the command is replayed and will be replaced when any \( r \)-class command is run after the estimation command.
Methods and formulas

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-R²

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

\[
\varepsilon_i = y_i - x_i' \hat{\beta}_\tau
\]

The objective function to be minimized is

\[
c_\tau(\varepsilon_i) = (\tau 1 \{\varepsilon_i \geq 0\} + (1 - \tau) 1 \{\varepsilon_i < 0\}) |\varepsilon_i| = (\tau 1 \{\varepsilon_i \geq 0\} - (1 - \tau) 1 \{\varepsilon_i < 0\}) \varepsilon_i \tag{2}
\]

where \( 1 \{\cdot\} \) 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 \( \hat{\beta}_\tau \) that minimize \( c_\tau(\varepsilon_i) \) is equivalent to finding the \( \hat{\beta}_\tau \) that make \( x \hat{\beta}_\tau \) best fit the quantiles of the distribution of \( y \) conditional on \( x \).

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, \( u_{n \times 1} \) and \( v_{n \times 1} \), are introduced, where \( u_i \geq 0, v_i \geq 0, \) and \( u_i \times v_i = 0 \), reformulating the problem as

\[
\min_{\beta_{\tau, u, v}} \{ \tau 1_n' u + (1 - \tau) 1_n' v \mid y - X \beta_{\tau} = u - v \}
\]

where \( 1_n \) 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 \texttt{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

\[
\text{cov}(\beta_{\tau}) = \frac{\tau(1 - \tau)}{f_Y^2(\xi_{\tau})} \left\{ E(x_i x'_i) \right\}^{-1}
\]  

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

Here we outline the logic underlying the \texttt{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,

\[
\frac{F_Y^{-1}(\tau)}{dt} \approx \frac{F_Y^{-1}(\tau + h) - F_Y^{-1}(\tau - h)}{2h} = \frac{\xi_{\tau+h} - \xi_{\tau-h}}{2h} = s_{\tau}
\]  

(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 \( \hat{F}_\psi^{-1}(\tau + h) = \bar{x}' \hat{\beta}_{\tau+h} \) and \( \hat{F}_\psi^{-1}(\tau - h) = \bar{x}' \hat{\beta}_{\tau-h} \), where \( \bar{x} \) is the sample mean of the independent variables \( x \). These quantities are then substituted into (4).

Alternatively, as the option suggests, \texttt{vce(iid, residual)} specifies that \texttt{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, \( \epsilon(\tau+h) = \hat{F}_\psi^{-1}(\tau + h) \) and \( \epsilon(\tau-h) = \hat{F}_\psi^{-1}(\tau - h) \). The \( \hat{F}_\psi^{-1} \) estimates are then substituted for \( F_Y^{-1} \) in (4).

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

\[ h_s = n^{-1/3} \Phi^{-1} \left( 1 - \frac{\alpha}{2} \right)^{2/3} \left[ \frac{3}{2} \times \frac{\phi\{\Phi^{-1}(\tau)\}^2}{2\Phi^{-1}(\tau)^2 + 1} \right]^{1/3} \]

\[ h_b = n^{-1/5} \left[ \frac{9}{2} \phi\{\Phi^{-1}(\tau)\}^4}{\{2\Phi^{-1}(\tau)^2 + 1\}^2} \right]^{1/5} \]

\[ h_c = \Phi^{-1} \left( 1 - \frac{\alpha}{2} \right) \sqrt{\frac{\tau(1 - \tau)}{n}} \]

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 \texttt{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 \texttt{vce()} suboption \texttt{kernel(kernel)} specifies that \texttt{qreg} use one of several kernel-density estimators to estimate the sparsity function. \texttt{kernel} allows you to choose which kernel function to use, where the default is the Epanechnikov kernel. See \texttt{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 \( y \); see Silverman (1986, 47) and Koenker (2005, 81) for discussions. Let \( \hat{f}_\epsilon(\epsilon_i) \) be the kernel density estimate for the \( i \)th residual, and then the kernel estimator for the sparsity function is

\[ \hat{s}_\tau = \frac{n h_k}{\sum_{i=1}^{n} \hat{f}_\epsilon(\epsilon_i)} \]

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

\[ V_n = \hat{s}_\tau^2 \tau(1 - \tau) \left( \sum_{i=1}^{n} x_i x_i' \right)^{-1} \]

**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 \( e^{v_i|\epsilon_i|} \), where \( v_i \) are multipliers

\[ v_i = \begin{cases} \tau & \text{if } \epsilon_i > 0 \\ (1 - \tau) & \text{otherwise} \end{cases} \]
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 0.25 and the positive residuals by 0.75. It can be shown that the criterion is minimized when 75% of the residuals are negative.

References


**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