0.1 rq: Quantile Regression for Continuous Dependent Variables

Use a linear programming implementation of quantile regression to estimate a linear predictor of the τth conditional quantile of the population.

Syntax

```
> z.out <- zelig(Y ~ X1 + X2, model = "rq", data = mydata, tau = 0.5)
> x.out <- setx(z.out)
> s.out <- sim(z.out, x = x.out)
```

Additional Inputs

In addition to the standard inputs, `zelig()` takes the following additional options for quantile regression:

- `tau`: defaults to 0.5. Specifies the conditional quantile(s) that will be estimated. 0.5 corresponds to estimating the conditional median, 0.25 and 0.75 correspond to the conditional quartiles, etc. If `tau` is a vector, the conditional quantile function at each tau is estimated. If tau is set outside of the interval [0,1], zelig returns the solution for all possible conditional quantiles given the data, but does not support inference on this fit (`setx` and `sim` will fail).
- `se`: a string value that defaults to "nid". Specifies the method by which the covariance matrix of coefficients is estimated during the `sim` stage of analysis. `se` can take the following values, which are passed to the `summary.rq` function from the `quantreg` package. These descriptions are copied from the `summary.rq` documentation.
  - `"iid"` which presumes that the errors are iid and computes an estimate of the asymptotic covariance matrix as in KB(1978).
  - `"nid"` which presumes local (in `tau`) linearity (in `x`) of the the conditional quantile functions and computes a Huber sandwich estimate using a local estimate of the sparsity.
  - `"ker"` which uses a kernel estimate of the sandwich as proposed by Powell(1990).
- `...`: additional options passed to `rq` when fitting the model. See documentation for `rq` in the `quantreg` package for more information.

Examples

1. Basic Example with First Differences

   Attach sample data, in this case a dataset pertaining to the efficiency of plants that convert ammonia to nitric acid. The dependent variable, stack.loss, is 10 times the percentage of ammonia that escaped unconverted:

```
> data(stackloss)
```

Estimate model:

```
> z.out1 <- zelig(stack.loss ~ Air.Flow + Water.Temp + Acid.Conc.,
+     model = "rq", data = stackloss, tau = 0.5)
```

Summarize regression coefficients:

```
> summary(z.out1)
```

Set explanatory variables to their default (mean/mode) values, with high (80th percentile) and low (20th percentile) values for the water temperature variable (the variable that indiates the temperature of water in the plant's cooling coils):

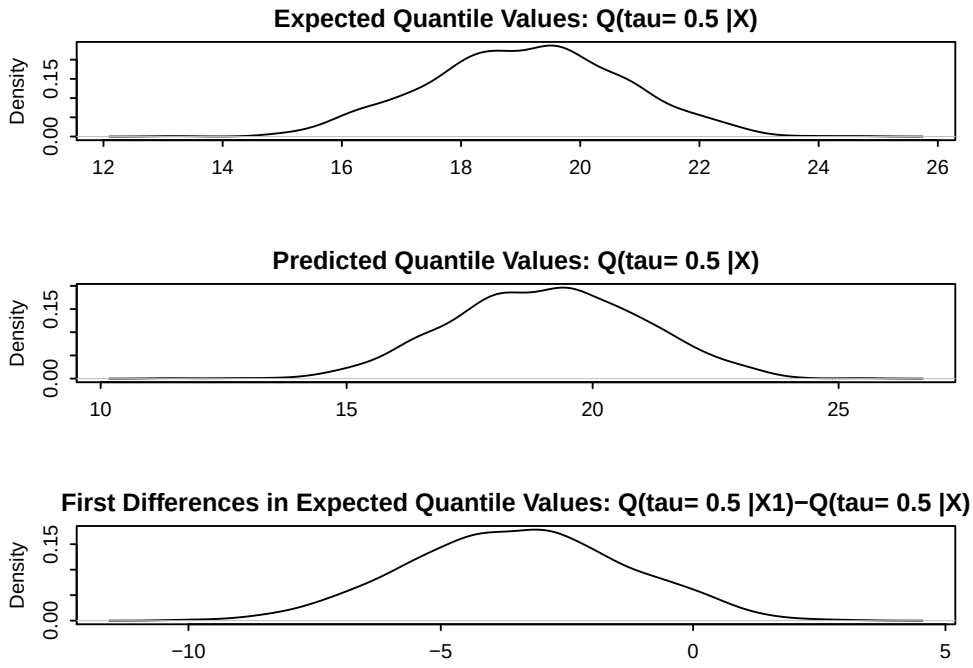
```
> x.high <- setx(z.out1, Water.Temp = quantile(stackloss$Water.Temp,
+     0.8))
> x.low <- setx(z.out1, Water.Temp = quantile(stackloss$Water.Temp,
+     0.2))
```

Generate first differences for the effect of high versus low water temperature on stack loss:

```
> s.out1 <- sim(z.out1, x = x.high, x1 = x.low)
```

```
> summary(s.out1)
```

```
> plot(s.out1)
```



2. Using Dummy Variables

   We can estimate a model of unemployment as a function of macroeconomic indicators and fixed effects for each country (see Section **??** for help with dummy variables). Note that you do not need to create dummy variables, as the program will automatically parse the unique values in the selected variable into discrete levels.

```
> data(macro)
> z.out2 <- zelig(unem ~ gdp + trade + capmob + as.factor(country),
+     model = "rq", tau = 0.5, data = macro)
```

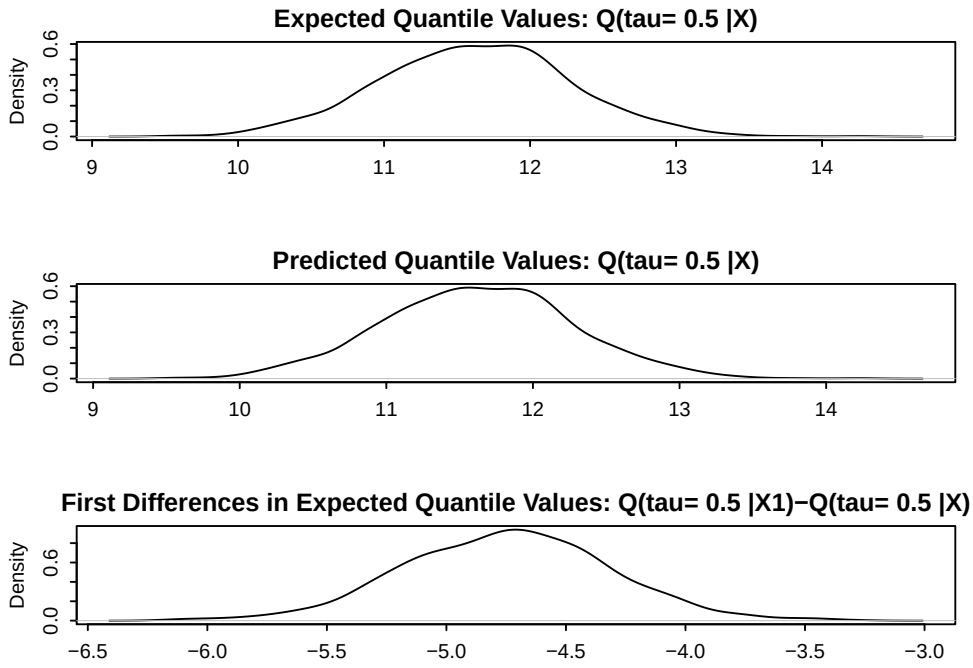
   Set values for the explanatory variables, using the default mean/mode values, with country set to the United States and Japan, respectively:

```
> x.US <- setx(z.out2, country = "United States")
> x.Japan <- setx(z.out2, country = "Japan")
```

   Simulate quantities of interest:

```
> s.out2 <- sim(z.out2, x = x.US, x1 = x.Japan)


> plot(s.out2)
```



3. Estimating Multiple Quantiles

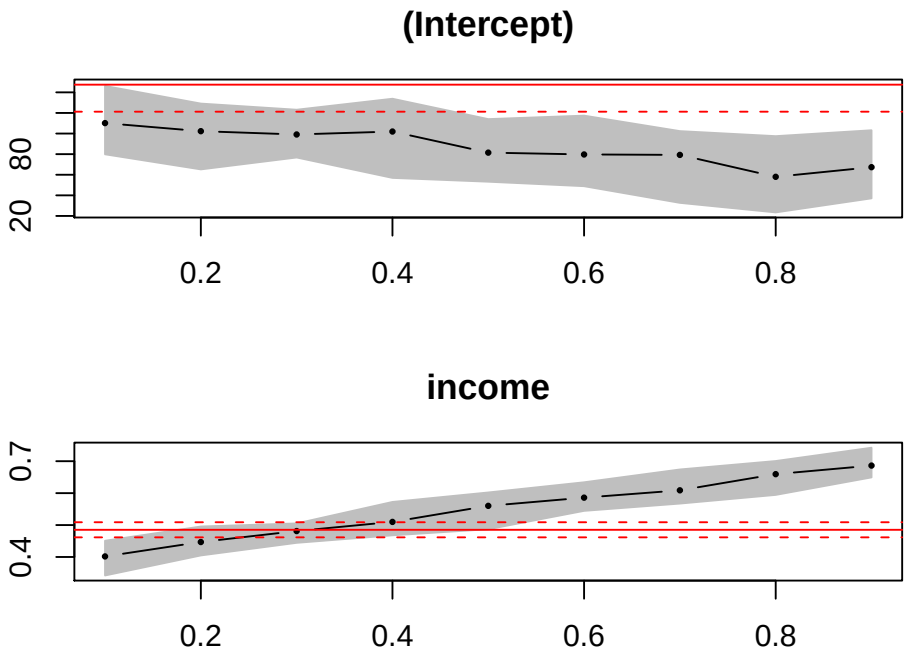
   Using the Engel dataset on food expenditure as a function of income, we can use the "rq" model to estimate multiple conditional quantiles:

```
> data(engel)
> z.out3 <- zelig(foodexp ~ income, model = "rq", tau = seq(0.1,
+     0.9, by = 0.1), data = engel)
```

   We can summarize the coefficient fits, or plot them to compare them to the least squares conditional mean estimator.

```
> summary(z.out3)
```

```
> plot(summary(z.out3))
```



Set the value of income to the top quartile and the bottom quartile of the income distribution for each fit:

```
> x.bottom <- setx(z.out3, income = quantile(engel$income,
+     0.25))
> x.top <- setx(z.out3, income = quantile(engel$income, 0.75))
```

Simulate quantities of interest for each fit simultaneously:

```
> s.out3 <- sim(z.out3, x = x.bottom, x1 = x.top)
```

Summary

```
> summary(s.out3)
```

Model

The quantile estimator is best introduced by considering the sample median estimator and comparing it to the sample mean estimator. To find the mean of a sample, we solve for the quantity μ which minimizes the sum squared residuals:

$$\mu \quad = \quad \arg\min_{\mu} \sum_{i} (y_i - \mu)^2$$

Estimating a quantile is similar, but we solve for ξ which minimizes the sum absolute residuals:

$$\xi = \arg\min_{\xi} \sum_i |y_i - \xi|$$

One can confirm the equivalence of these optimization problems and the standard mean and median operators by taking the derivative with respect to the argument and setting it to zero.

The relationship between quantile regression and ordinary least squares regression is analogous to the relationship between the sample median and the sample mean, except we are now solving for the conditional median or conditional mean given covariates and a linear functional form. The optimization problems for the sample mean and median are then easily generalized to optimization problems for estimating *conditional* means or medians by replacing μ or ξ with a linear combination of covariates $X'\beta$:

$$\begin{aligned} \hat{\beta}_{\text{mean}} &= \arg\min_{\beta} \sum_i (Y_i - X_i'\beta)^2 \\ \hat{\beta}_{\text{median}} &= \arg\min_{\beta} \sum_i |Y_i - X_i'\beta| \end{aligned} \tag{1}$$

Equation 1 can be generalized to provide any quantile of the conditional distribution, not just the median. We do this by weighting the aboslute value function asymmetrically in proportion to the requested τth quantile:

$$\begin{aligned} \hat{\beta}_\tau &= \arg\max_{\beta} \sum \rho(Y_i - X_i'\beta) \\ \rho &= \tau(1 - I(Y - X_i'\beta > 0)) + (1 - \tau) I(Y - X_i'\beta > 0) \end{aligned} \tag{2}$$

We call the asymmetric absolute value function a "check function". This optimization problem has no closed form solution and is solved using linear programming, so unlike most `Zelig` models, it is not straightforward to specify a systematic and stochastic component for conditional quantile estimates. However, the following systematic and stochastic components do emerge asymptotically in the large-n limit.

Let Q_τ be the true conditional quantile for a given set of covariates X_i, and $\hat{Q}_\tau$ be the conditional quantile estimator. Then the *stochastic component* is described by a density with mean $\hat{Q}_\tau$ and variance σ^2:

$$Q_\tau = \mathcal{N}(\hat{Q}_\tau, \sigma^2)$$

The *systematic component* models the mean and variance of the above density as:

$$\begin{aligned} \hat{Q}_\tau &= x'\hat{\beta}_\tau \\ \sigma^2 &= \frac{\tau(1-\tau)}{nf^2} \end{aligned}$$

Where $\hat{\beta}$ is the vector that solves equation 2, n is the number of datapoints, and f is the true population density at the τth conditional quantile. Zelig uses this asymptotic approximation of stochastic and systematic components in simulation and numerically estimates the population density to derive σ^2. The simulation results should thus be treated with caution when using small datasets as both this asymptotic approximation and the population density approximation can break down.

Quantities of Interest

- The expected value (`qi$ev`) is the mean of simulations from the stochastic component,

$$E(\hat{Q}_\tau) = x_i \beta_\tau,$$

  given a draw of β_τ from its sampling distribution. Variation in the expected value distribution comes from estimation uncertainty of β_τ.

- The predicted value (`qi$pr`) is the result of a single draw from the stochastic component given a draw of β_τ from its sampling distribution. The distribution of predicted values should be centered around the same place as the expected values but have larger variance because it includes both estimation uncertainty and fundamental uncertainty.

- This model does not support conditional prediction.

Output Values

The output of each Zelig command contains useful information which you may view. For example, if you run `z.out <- zelig(y ~ x, model = "ls", data)`, then you may examine the available information in `z.out` by using `names(z.out)`, see the `coefficients` by using `z.out$coefficients`, and a default summary of information through `summary(z.out)`. Other elements available through the `$` operator are listed below.

- `zelig()` will return an object `z.out`, which is of class `rq` (when `tau` specifies a single quantile), `rqs` (when `tau` specifies multiple quantiles), or `rq.process` (when `tau` is a value outside of $[0, 1]$. The `rq` and `rqs` objects are supported by Zelig's prediction utilities, but `rq.process` is not. These objects maintain the same functionality that they have in the `quantreg` package – for example, one can call the `summary` or `plot` methods on them directly. See documentation for `rq` from package `quantreg` for details. The following information can be extracted directly from the `z.out` object:
  - `coefficients`: parameter estimates for the explanatory variables.
  - `residuals`: vector of differences between Y and $X'\beta_\tau$.
  - `fitted.values`: fitted values given by $X'\beta_\tau$.
  - `zelig.data`: the input data frame if `save.data = TRUE`.

- When `zelig` was called with a single `tau` value, `summary(z.out)` returns a `summary.rq` object. When `zelig` was called with τ as a vector, `summary(z.out)` returns a list of `summary.rq` objects. From each `summary.rq` object you may extract:
  - `coefficients`: the parameter estimates with their 95 percent confidence intervals. The user can also obtain standard errors and p-values by specifying the `se` argument to `summary`. See documentation for `summary.rq` in the `quantreg` package for details.
  - `rdf`: the residual degrees of freedom.
  - `cov`: a $k \times k$ matrix of unscaled covariances. To obtain this attribute, the user must specify `cov=TRUE` as an argument to `summary`. See documentation for `summary.rq` in the `quantreg` package for details.
- From the `sim()` output object `s.out`, you may extract quantities of interest arranged as matrices indexed by simulation $\times$ `x`-observation (for more than one `x`-observation). Available quantities are:
  - `qi$ev`: the simulated expected values for the specified values of `x`.
  - `qi$pr`: the simulated predicted values for the specified values of `x`.
  - `qi$fd`: the simulated first differences (or differences in expected values) for the specified values of `x` and `x1`.

How to Cite

To cite Zelig as a whole, please reference these two sources:

Kosuke Imai, Gary King, and Olivia Lau. 2007. "Zelig: Everyone's Statistical Software," `http://GKing.harvard.edu/zelig`.

Imai, Kosuke, Gary King, and Olivia Lau. (2008). "Toward A Common Framework for Statistical Analysis and Development." Journal of Computational and Graphical Statistics, Vol. 17, No. 4 (December), pp. 892-913.

See also

The quantile regression package `quantreg` by Richard Koenker. In addition, advanced users may wish to refer to `help(rq)`, `help(summary.rq)` and `help(rq.object)`.

Bibliography