Understanding and estimating conditional parametric quantile models

Matteo Bottai, ScD

Karolinska Institutet
Stockholm, Sweden

Acknowledgments

Nonlinear parametric quantile models

1. Bottai M, Cilluffo G. Nonlinear parametric quantile models. *Statistical Methods in Medical Research*, 2020
2. Bottai M, Orsini N. qmodel: a command for fitting parametric quantile models. *The Stata Journal*, 2019

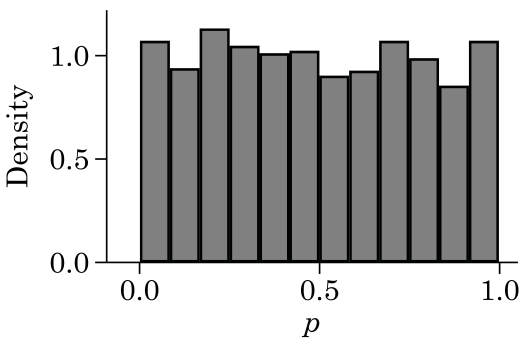
Linear parametric quantile models

3. Frumento P, Bottai M, Fernndez-Val I. Parametric modeling of quantile regression coefficient functions with longitudinal data. *Journal of the American Statistical Association*, 2021
4. Sottile G, Frumento P, Chiodi M, Bottai M. A penalized approach to covariate selection through quantile regression coefficient models. *Statistical Modelling*, 2019
5. Frumento P, Bottai M. Parametric modeling of quantile regression coefficient functions with censored and truncated data. *Biometrics*, 2017
6. Frumento P, Bottai M. Parametric modeling of quantile regression coefficient functions. *Biometrics*, 2016

A Uniform Variable

We generate 1,000 values from a uniform distribution.

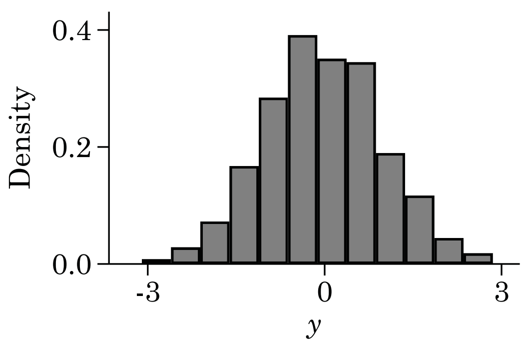
```
. quietly set obs 1000
. generate p = runiform()
. quietly hist p
```



A Standard Normal Variable

We generate 1,000 values from a standard normal distribution.

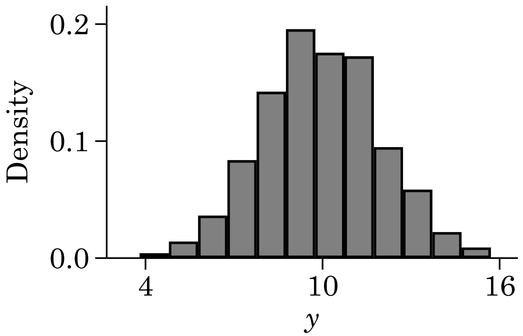
```
. generate p = runiform()
. generate y = invnormal(p)
. quietly hist y
```



A Normal Variable

We generate 1,000 values from a normal distribution.

```
. generate p = runiform()
. generate y = 10 + 2 * invnormal(p)
. quietly hist y
```



A Parametric Model with `qmodel`

We observe a random sample of n values $y_1, \ldots, y_n$.
We assume a parametric model,

$$y_i = \mu + \sigma z_i$$

with z_i a standard normal variable.

We estimate the parameters with `qmodel`.

```
. generate p = runiform()
. generate y =  10  +    2    * invnormal(p)
. qmodel   y = {mu} + {sigma} * invnormal(p)
Parametric Quantile Model                          Number of obs =      1,000
                                                   AIC           =   10.34585
Loss function = 570.1219037                        BIC           =   20.16136
------------------------------------------------------------------------------
           y |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
          mu |   9.946733   .0651033   152.78   0.000     9.819133    10.07433
       sigma |   2.007513   .0502369    39.96   0.000      1.90905    2.105975
------------------------------------------------------------------------------
```


Built-in Functions

The `qmodel` command has a number of built-in functions.

```
. generate p = runiform()
. generate y =  10 + 2 * invnormal(p)
. qmodel y = _normal
Parametric Quantile Model                          Number of obs =      1,000
                                                   AIC           =   10.34585
Loss function = 570.1219037                        BIC           =   20.16136
------------------------------------------------------------------------------
           y |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
normal       |
        mean |   9.946733   .0651033   152.78   0.000     9.819133    10.07433
     log(sd) |   .6968965   .0250245    27.85   0.000     .6478494    .7459436
------------------------------------------------------------------------------
```

The `_normal` function estimates $\log(\sigma)$ because $\sigma > 0$.
This generally improves stability of the estimation algorithm.

Log-Normal Distribution

We consider a log-normal distribution.

$$\log(y_i) = \mu + \sigma z_i \quad \text{or} \quad y_i = \exp(\mu + \sigma z_i)$$

with z_i a standard normal variable.

```
. generate p = runiform()
. generate  y = exp(invnormal(p))
. // qmodel y = exp(_normal)
. qmodel log(y) = _normal
Parametric Quantile Model                          Number of obs =      1,000
                                                   AIC           =   9.652212
Loss function =  284.921086                        BIC           =   19.46772
------------------------------------------------------------------------------
      log(y) |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
normal       |
        mean |  -.0265395   .0325531    -0.82   0.415    -.0903425    .0372634
     log(sd) |   .0031389   .0252176     0.12   0.901    -.0462867    .0525646
------------------------------------------------------------------------------
```

Location-Shift Models

A general location-shift model is

$$s_0(y_i) = s_1(x_i, \beta) + s_2(x_i, \sigma)Q(p_i)$$

with s_0, s_1, s_2 functions, Q a quantile function, and p_i uniform.

For example, the accelerated failure time model is

$$\log(y_i) = x_i'\beta + \sigma \log[p_i/(1-p_i)]$$

and the proportional hazard model is

$$\log H_0(y_i) = x_i'\beta + \log[-\log(1-p_i)]$$

Quantile Regression Coefficient Models

We consider a quantile regression coefficient model,

$$y_i = \mu + \sigma z_i + \alpha \log(p_i)x_i + (\beta_0 + \beta_1 p_i)w_i$$

with z_i and w_i standard normal, x_i Bernoulli, and p_i uniform.

```
. generate z = rnormal()
. generate x = rbinomial(1,.5)
. generate w = rnormal()
. generate p = runiform()
. generate         y = 10 + exp(.5)*z +  log(p)  * x +    (1-p)    * w
. quietly qmodel   y =    _(_normal)_  + _(_log)_ * x + _(_linear)_ * w
```


Quantile Regression Coefficient Models

We consider a quantile regression coefficient model,

$$y_i = \mu + \sigma z_i + \alpha \log(p_i)x_i + (\beta_0 + \beta_1 p_i)w_i$$

```
. qmodel   y =   _(_normal)_  + _(_log)_ * x + _(_linear)_ * w
Parametric Quantile Model                        Number of obs =      1,000
                                                           AIC =   16.25148
Loss function = 518.7776441                                BIC =   40.79025
```

y	Coef.	Std. Err.	z	P>|z|	[95% Conf. Interval]	
normal						
mean	9.880561	.0677105	145.92	0.000	9.747851	10.01327
log(sd)	.416787	.0345554	12.06	0.000	.3490597	.4845143
log						
log(p)	.7585247	.1025104	7.40	0.000	.5576079	.9594414
linear						
constant	.5857826	.0611316	9.58	0.000	.4659669	.7055983
p	.1198039	.1491004	0.80	0.422	-.1724275	.4120353

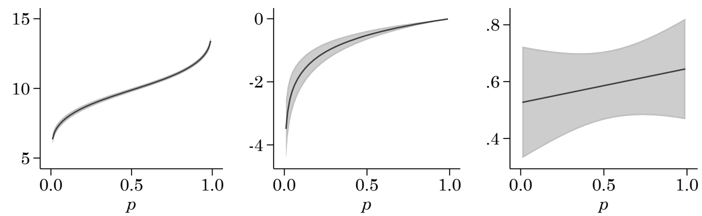
Post-estimation Commands: `qmodel_plot`

We consider a quantile regression coefficient model,

$$y_i = \mu + \sigma z_i + \alpha \log(p_i)x_i + (\beta_0 + \beta_1 p_i)w_i$$

We plot the estimated regression coefficients.

```
. qmodel_plot, ci
```



Post-estimation Commands: predict

We consider a quantile regression coefficient model,

$$y_i = \mu + \sigma z_i + \alpha \log(p_i) x_i + (\beta_0 + \beta_1 p_i) w_i$$

We estimate the regression coefficients at different proportions.

```
. range proportion .01 .99 99
(901 missing values generated)
. predict base beta_x beta_w, p(proportion)
(901 missing values generated)
. list proportion base beta_x beta_w if inlist(_n,10,25,50,75,90), clean noobs abbrev:
    proportion        base       beta_x      beta_w
            .1    7.936345    -1.746568     .537861
           .25    8.857306    -1.051538    .5558316
            .5    9.880561    -.5257692    .5857826
           .75    10.90382    -.2182139    .6157336
            .9    11.82478    -.0799186    .6337042
```


Post-estimation Commands: qmodel_quantile

We consider a quantile regression coefficient model,

$$y_i = \mu + \sigma z_i + \alpha \log(p_i) x_i + (\beta_0 + \beta_1 p_i) w_i$$

We estimate the 90th percentile at specified covariate values.

```
. qmodel_quantile .9, at(x=0 w=1.5) noheader
Quantile p=.9 at x=0 w=1.5
```

y	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
quantile	12.77533	.1345677	94.94	0.000	12.51158	13.03908

Final Notes

Parametric quantile models expand modeling possibilities.

The `qmodel` command

- ✓ allows any parametric quantile model
- ✓ has convenient built-in functions
- ✓ has useful post-estimation commands
- ✓ has help documentation and a Stata Journal article
- × is computationally slow
- × is sensitive to model misspecifications

Future work may aim to

- implement censoring, truncation, and random effects
- improve computational speed