# Quantile regression robustness

Anton Antonov
*Mathematica* for Prediction at GitHub
*Mathematica* for Prediction blog
December 2013

## Introduction

This document shows examples of robustness of quantile regression. The quantile regression curves (called “regression quantiles”) are computed with the *Mathematica* package [1] hosted at the MathematicaForPrediction project at GitHub. Quantile regression was introduced by Koenker and Bassett in [3] and detailed theoretical descriptions and discussions are given the book “Quantile regression” by Koenker (see [4]).

This document extends the descriptions in [2]. More complicated data distributions can be used, but the data used in the document although based on a simple deterministic model has a simple heteroscedasticity (the variance varies with *x*, see [5]).

## Load

Load the package [1]:

In[53]:=
```
Get["~/MathFiles/MathematicaForPrediction/QuantileRegression.m"]
```

Here is the definition of the data generation function used below:

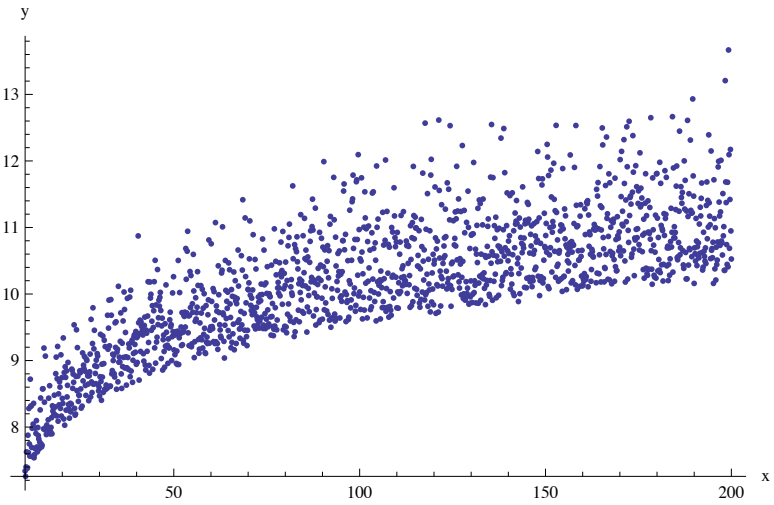
In[54]:=
```
Clear[LogarithmicCurveWithNoise]
LogarithmicCurveWithNoise[
   nPoints_Integer, start_?NumberQ, end_?NumberQ] :=
  Block[{data},
   data =
    Table[{t, 5 + Log[t] + RandomReal[SkewNormalDistribution[0, Log[t]/5, 12]]},
     {t, Rescale[Range[1, nPoints], {1, nPoints}, {start, end}]}];
   data
  ];
```

## Starting data and regression quantiles

Let us generate some data.

In[56]:=
```
data = LogarithmicCurveWithNoise[1500, 10, 200];
ListPlot[data, AxesLabel → {"x", "y"}, PlotRange → All, ImageSize → 400]
```

Out[57]=




We know that the model for the data is

$$y = \beta_0 + \beta_1 x + \beta_3 \log(x), \tag{1}$$

hence are going to fit the model functions:

In[58]:=
```
funcs = {1, x, Log[x]};
```

Here are the regression quantiles:

In[59]:=
```
qrFuncs = QuantileRegressionFit[data, funcs, x, qs];
TableForm[List /@ Chop[qrFuncs]]
```

Out[60]//TableForm=

```
5.01985 + 0.0000412499 x + 1.00452 Log[x]
4.97137 + 1.07237 Log[x]
4.99696 + 1.14097 Log[x]
5.01935 + 1.2262 Log[x]
4.95386 + 1.41213 Log[x]
```

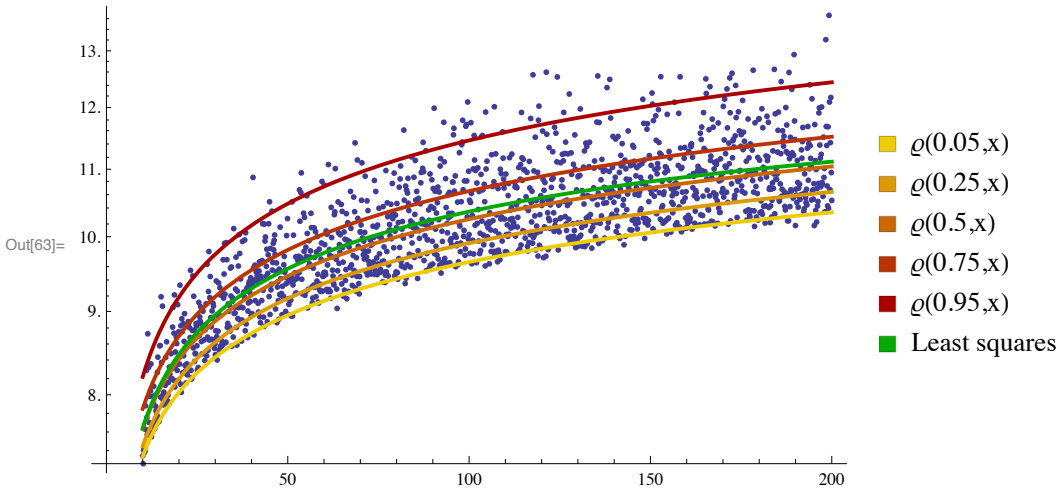
Here is the least regression fit:

In[61]:=
```
fFunc = Fit[data, funcs, x]
```

Out[61]=
```
4.83463 - 0.000888869 x + 1.21936 Log[x]
```

Here is a plot of the data and all fitted functions:

Out[63]=

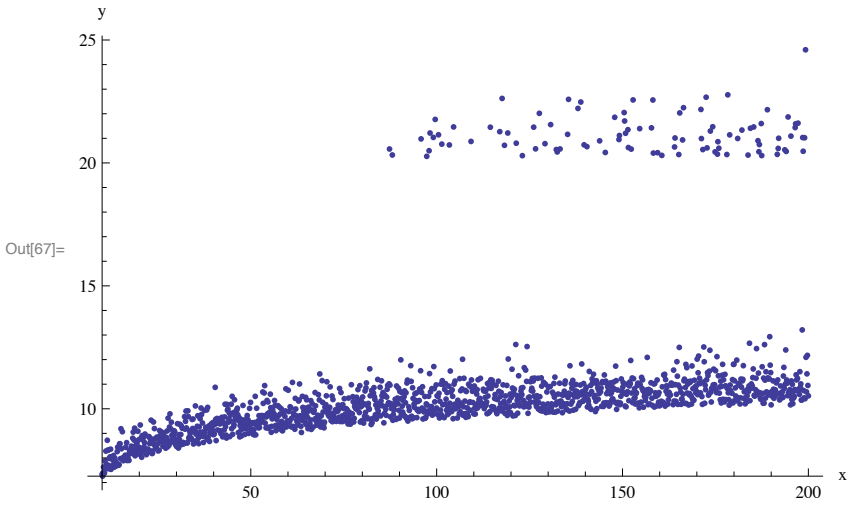



## Outliers above a constant

Let us demonstrate the robustness of the regression quantiles with the data of the previous example. Suppose that for some reason 50% of the data *y*-values greater than 11.25 are altered by multiplying them with a some greater than 1 factor, say, $\alpha = 1.8$ . Then the altered data looks like this:

In[65]:=
```
α = 1.8;
dataAlt =
  Map[If[RandomInteger[{0, 1}] == 1 && #[[2]] > 11.25, {#[[1]], α #[[2]]}, #] &, data];
ListPlot[dataAlt, AxesLabel → {"x", "y"}, PlotRange → All, ImageSize → 400]
```

Out[67]=




Let us compute the regression quantiles for the altered data:

In[68]:=
```
qrFuncsAlt = QuantileRegressionFit[dataAlt, funcs, x, qs];
TableForm[List /@ Chop[qrFuncs]]
```

Out[69]//TableForm=

5.01985 + 0.0000412499 x + 1.00452 Log[x]
4.97137 + 1.07237 Log[x]
4.99696 + 1.14097 Log[x]
5.01935 + 1.2262 Log[x]
4.95386 + 1.41213 Log[x]

and let us also compute the least squares fit of the model (1):

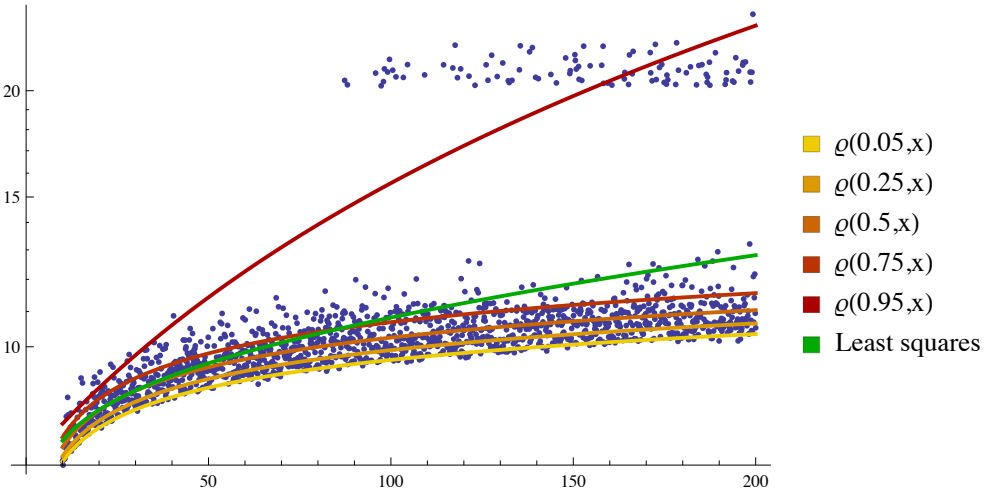
In[70]:=
```
fFunc = Fit[data, funcs, x]
fFuncAlt = Fit[dataAlt, funcs, x]
```

Out[70]= 4.83463 − 0.000888869 x + 1.21936 Log[x]

Out[71]= 5.86084 + 0.0146416 x + 0.759538 Log[x]

Here is a plot of the altered data and all fitted functions:

Out[73]=




We can see that the new regression quantiles computed for 0.05, 0.25, and 0.5 have not changed significantly:

In[75]:=
```
Grid[List /@ Flatten[Transpose[{Chop[qrFuncs], Chop[qrFuncsAlt]}]],
 Alignment → Left, Dividers →
  {{True, True}, Flatten@{Table[{True, False}, {Length[qrFuncs]}], True}}]
```

Out[75]=

| | |
|---|---|
| 5.01985 + 0.0000412499 x + 1.00452 Log[x] | 5.01985 + 0.0000412455 x + 1.00452 Log[x] |
| 4.97137 + 1.07237 Log[x] | 4.97137 + 1.07237 Log[x] |
| 4.99696 + 1.14097 Log[x] | 4.99696 + 1.14097 Log[x] |
| 5.01935 + 1.2262 Log[x] | 4.97944 + 0.000172092 x + 1.23582 Log[x] |
| 4.95386 + 1.41213 Log[x] | 7.29262 + 0.0828048 x |

ant that they are still good for separating the un-altered data:

In[76]:=
```
tbl = Table[{qs[[i]], Length[Select[data, #[[2]] ≥ (qrFuncsAlt[[i]] /. x → #[[1]]) &]] /
      Length[data] // N}, {i, Length[qs]}];
TableForm[tbl, TableHeadings → {None, {"quantile", "fraction\nabove"}}]
```

Out[76]//TableForm=

| quantile | fraction above |
|---|---|
| 0.05 | 0.950667 |
| 0.25 | 0.749333 |
| 0.5 | 0.499333 |
| 0.75 | 0.241333 |
| 0.95 | 0.012 |

Also we can see that the least squares fit of (1) has significantly changed:

In[77]:=
```
fFunc
```

Out[77]=
```
4.83463 - 0.000888869 x + 1.21936 Log[x]
```

In[78]:=
```
fFuncAlt
```

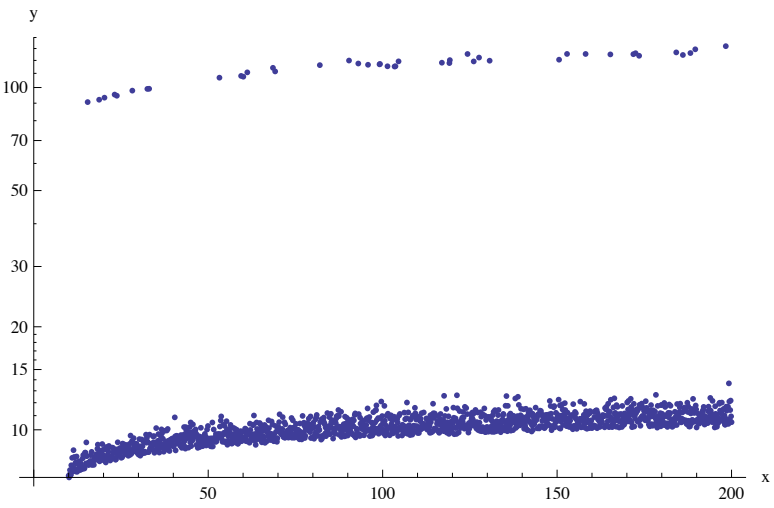
Out[78]=
```
5.86084 + 0.0146416 x + 0.759538 Log[x]
```

## Outliers above a chosen regression quantile

Let us demonstrate the robustness of the regression quantiles with different type of data alternation. Suppose that for some reason 70 % of the data *y*-values above the altered by multiplying them with a some greater than 1 factor, say, $\alpha = 10$ . Then the altered data looks like this:

In[79]:=
```
α = 10.;
dataAlt = Map[If[RandomReal[{0, 1}] > 0.3 && (#[[2]] > (qrFuncs[[5]] /. x -> #[[1]])),
      {#[[1]], α #[[2]]}, #] &, data];
ListLogPlot[dataAlt, AxesLabel → {"x", "y"}, PlotRange → All, ImageSize → 400]
```

Out[81]=




Let us compute the regression quantiles for the altered data:

In[82]:=
```
qrFuncsAlt = QuantileRegressionFit[dataAlt, funcs, x, qs];
TableForm[List /@ Chop[qrFuncs]]
```

Out[83]//TableForm=

$5.01985 + 0.0000412499\, x + 1.00452 \operatorname{Log}[x]$
$4.97137 + 1.07237 \operatorname{Log}[x]$
$4.99696 + 1.14097 \operatorname{Log}[x]$
$5.01935 + 1.2262 \operatorname{Log}[x]$
$4.95386 + 1.41213 \operatorname{Log}[x]$

and let us also compute the least squares fit of the model (1):

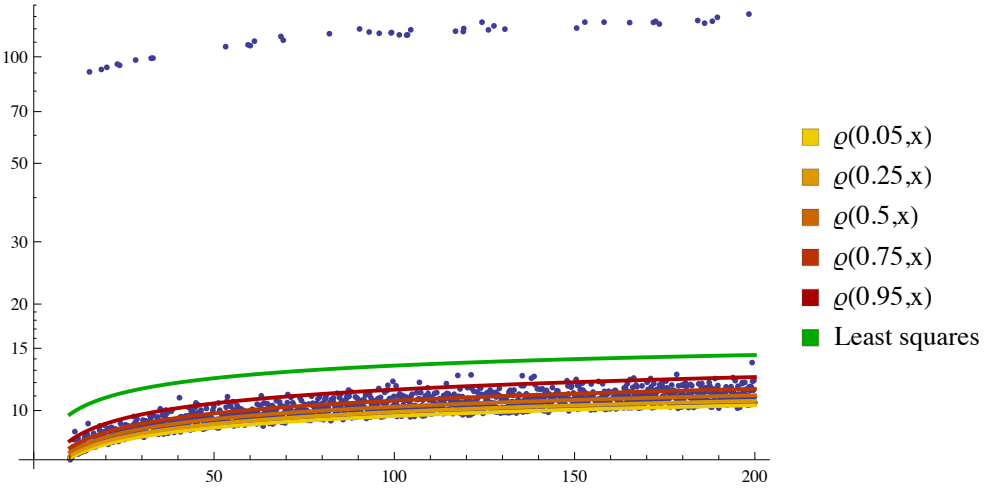
In[84]:=
```
fFunc = Fit[data, funcs, x]
fFuncAlt = Fit[dataAlt, funcs, x]
```

Out[84]= $4.83463 - 0.000888869\, x + 1.21936 \operatorname{Log}[x]$

Out[85]= $5.95578 - 0.0018861\, x + 1.6569 \operatorname{Log}[x]$

Here is a plot of the altered data and all fitted functions:

Out[87]=




We can see that the new regression quantiles computed for 0.05, 0.25, 0.5, 0.75, and 0.95 have not changed significantly:

In[89]:=
```
Grid[List /@ Flatten[Transpose[{Chop[qrFuncs], Chop[qrFuncsAlt]}]],
 Alignment → Left, Dividers →
  {{True, True}, Flatten@{Table[{True, False}, {Length[qrFuncs]}], True}}]
```

Out[89]=

| | |
|---|---|
| $5.01985 + 0.0000412499\, x + 1.00452 \operatorname{Log}[x]$ | $5.01985 + 0.0000412489\, x + 1.00452 \operatorname{Log}[x]$ |
| $4.97137 + 1.07237 \operatorname{Log}[x]$ | $4.97137 + 1.07237 \operatorname{Log}[x]$ |
| $4.99696 + 1.14097 \operatorname{Log}[x]$ | $4.99707 + 1.14095 \operatorname{Log}[x]$ |
| $5.01935 + 1.2262 \operatorname{Log}[x]$ | $5.01935 + 1.2262 \operatorname{Log}[x]$ |
| $4.95386 + 1.41213 \operatorname{Log}[x]$ | $4.95386 + 3.10588 \times 10^{-10}\, x + 1.41213 \operatorname{Log}[x]$ |

ant that they are still good for separating the un-altered data:

In[90]:=
```
tbl = Table[{qs[[i]], Length[Select[data, #[[2]] ≥ (qrFuncsAlt[[i]] /. x → #[[1]]) &]] /
      Length[data] // N}, {i, Length[qs]}];
TableForm[tbl, TableHeadings → {None, {"quantile", "fraction\nabove"}}]
```

Out[90]//TableForm=

| quantile | fraction above |
|---|---|
| 0.05 | 0.95 |
| 0.25 | 0.75 |
| 0.5 | 0.5 |
| 0.75 | 0.25 |
| 0.95 | 0.05 |

Also we can see that the least squares fit of (1) has significantly changed:

In[91]:=
```
fFunc
```

Out[91]=
```
4.83463 - 0.000888869 x + 1.21936 Log[x]
```

In[92]:=
```
fFuncAlt
```

Out[92]=
```
5.95578 - 0.0018861 x + 1.6569 Log[x]
```

## Conclusions

The examples considerd clearly demonstrate the robustness of quantile regression when compared to the least squares fit. As in the single distribution case, computing quantiles can be very useful for identifying outliers. For example, we can do the regression analogue of standardizing the data by subtracting the median and dividing by the interquartile distances, and declare any point outside of specified range as an outlier.

## References

[1] Anton Antonov, Quantile regression *Mathematica* package, source code at GitHub, https://github.-com/antononcube/MathematicaForPrediction, package QuantileRegression.m, (2013).

[2] Anton Antonov, “Quantile regeression through linear programming”, a package usage guide at GitHub, https://github.com/antononcube/MathematicaForPrediction, (2013).
URL: https://github.com/antononcube/MathematicaForPrediction/blob/master/Documentation/Quantile%20 regression %20 through %20 linear %20 programming.pdf .

[3] Roger Koenker, Gilbert Bassett Jr., “Regression Quantiles”, Econometrica, 46(1), 1978, pp. 33-50. JSTOR URL: http://links.jstor.org/sici?sici=0012-9682%28197801 %2946 %3 A1 %3 C33 %3 ARQ %3 E2 .0.CO%3 B2-J .

[4] Roger Koenker, Quantile regression, Econometric Society Monographs (Book 38), Cambridge University Press, (2005). Google URL: http://books.google.com/books/about/Quantile_Regression.html?id=hdkt7V4NXsgC .

[5] Wikipedia entry, Heteroscedasticity, http://en.wikipedia.org/wiki/Heteroscedasticity .