Quantile regression through linear programming

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

Introduction

We can say that least squares linear regression corresponds to finding the mean of a single distribution. Similarly, quantile regression corresponds to finding quantiles of a single distribution. With quantile regression we obtain curves -- called “regression quantiles” -- that together with the least squares regression curve would give a more complete picture of the distributions (the *y*’s) corresponding to a set of *x*’s.

For a complete, interesting, and colorful introduction and justification to quantile regression see [2]. An introduction and description of the major properties of quantile regression is given in the Wikipedia entry [3].

In order to have a fast enough for practical purposes implementation of quantile regression we need to re-cast the quantile regression problem as linear programming problem. (Such a formulation is also discussed in [2].)

This document is mostly a guide for usage of the *Mathematica* package for quantile regression that is provided by the MathematicaForPrediction project at GitHub, see [1].

The second section provides theoretical background of the linear programming formulation of the quantile regression problem. The third section shows examples of finding regression quantiles using the function `QuantileRegression` provided by [1]. The last section describes profiling experiments and their results.

The motivational examples in the theoretical section, formulas (1) and (2), can be completed with more expansions and proofs. (Which will be done in the next version of the document.)

Theory

We can formulate the quantile regression problem in way analogous to the formulation of least squares (conditional mean) regression.

Consider a random variable Y having some distribution function F and a sample $\{y_i\}_{i=1}^n$ of Y. The median of the set of samples $\{y_i\}_{i=1}^n$ can be defined as the solution of the minimization problem

$$\min_{\beta} \sum_{i=1}^{n} |y_i - \beta|, \ \beta \in \mathbb{R} . \tag{1}$$

To see that the β which minimizes (1) is the median of $\{y_i\}_{i=1}^n$, consider two points $y_1 < y_2$. Then $|y_1 - \mu| + |\mu - y_2| = |y_1 - y_2|$, $\forall\ \mu \in [y_1, y_2]$, hence any $\mu \in [y_1, y_2]$ minimizes (1). Using the observation for two points we see that for three points $y_1 < y_2 < y_3$, $\mu = y_2$ minimizes (1). For four points $y_1 < y_2 < y_3 < y_4$ any $\mu \in [y_2, y_3]$ minimizes (1). We can generalize these observations and show that (1) gives the median for any set of points.

If we want to find θ-th sample quantile of $\{y_i\}_{i=1}^n$ then we need to change (1) into

$$\min_{\beta} \left(\sum_{y_i \geq \beta} \theta \, |y_i - \beta| + \sum_{y_i < \beta} (1 - \theta) \, |y_i - \beta| \right), \ \beta \in \mathbb{R}. \tag{2}$$

Consider a set of random variables Y_i, $i \in [1, n]$, $n \in \mathbb{N}$ that are paired with a set of x-coordinates $X = \{x_i\}_{i=1}^n$. We have data of pairs $\{x_i, y_i\}_{i=1}^n$, where y_i is a realization of Y_i.

The linear regression problem can be formulated as

$$\min_{\beta_0, \beta_1} \sum_{i=1}^{n} (y_i - (\beta_0 + \beta_1 x_i))^2. \tag{3}$$

Similarly, the median regression problem can be formulated as

$$\min_{\beta_0, \beta_1} \sum_{i=1}^{n} |y_i - (\beta_0 + \beta_1 x_i)| . \tag{4}$$

and the θ-th quantile regression problem as

$$\min_{\beta_0, \beta_1} \left(\sum_{i \in \{i : y_i \geq \beta_0 + \beta_1 x_i\}} \theta \, |y_i - (\beta_0 + \beta_1 x_i)| + \sum_{i \in \{i : y_i < \beta_0 + \beta_1 x_i\}} (1 - \theta) \, |y_i - (\beta_0 + \beta_1 x_i)| \right), \ \beta_0, \beta_1 \in \mathbb{R}. \tag{5}$$

In order to convert (5) into a linear programming problem, let us introduce the non-negative variables u_i and v_i for which the following equations are true:

$$\begin{aligned} &y_i - (\beta_0 + \beta_1 x_i) + u_i = 0, \ i \in \{i : y_i \geq \beta_0 + \beta_1 x_i\}, \\ &u_i = 0, \ i \notin \{i : y_i \geq \beta_0 + \beta_1 x_i\}, \end{aligned} \tag{6}$$

$$\begin{aligned} &(\beta_0 + \beta_1 x_i) - y_i + v_i = 0, \ i \in \{i : y_i < \beta_0 + \beta_1 x_i\}, \\ &v_i = 0, \ i \notin \{i : y_i < \beta_0 + \beta_1 x_i\}. \end{aligned} \tag{7}$$

Since u_i and v_i are greater than 0 on complementary sets, we can re-write (6) and (7) simply as

$$y_i - (\beta_0 + \beta_1 x_i) + u_i - v_i = 0, \ u_i \geq 0, \ v_i \geq 0, \ i \in [1, n]. \tag{8}$$

Then (5) expressed with u_i and v_i becomes

$$\min_{u_i, v_i, \beta_0, \beta_1} \left(\sum_{i \in \{i : y_i \geq \beta_0 + \beta_1 x_i\}} \theta \, u_i + \sum_{i \in \{i : y_i < \beta_0 + \beta_1 x_i\}} (1 - \theta) \, v_i \right). \tag{9}$$

Since $u_i \geq 0$ and $v_i \geq 0$ the minimization function (9) can simply be written as

$$\min_{u_i, v_i, \beta_0, \beta_1} \left(\sum_{i=1}^{n} \theta \, u_i + \sum_{i=1}^{n} (1 - \theta) \, v_i \right). \tag{10}$$

The equations (8) and formula (10) are the linear programming formulation of the quantile regression problem (5).

Note that $u_i \, v_i = 0, \ \forall \, i \in [1, n]$.

The quantile regression formulations (5), and (8) and (10) can be done for any model of Y_i that is a linear combination of functions over X not just for the linear model $\beta_0 + \beta_1 X$.

Examples of usage

Package load

Load the package [1]:

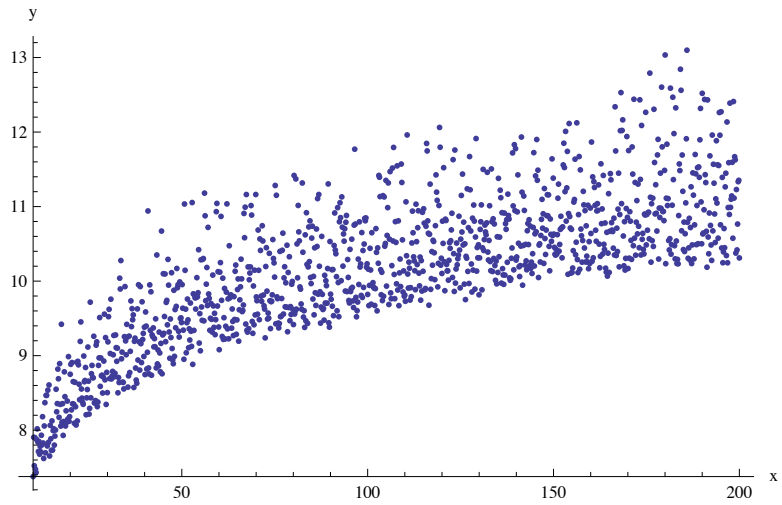
In[1]:=
```
Get["~/MathFiles/MathematicaForPrediction/QuantileRegression.m"]
```

Logarithmic curve with noise

Let us generate some data.

In[2]:=
```
Block[{n = 1200, start = 10, end = 200},
 data = Table[{t, 5 + Log[t] +
       RandomReal[SkewNormalDistribution[0, Log[t] / 5, 12]]},
    {t, Rescale[Range[1, n], {1, n}, {start, end}]}];
]
ListPlot[data, AxesLabel → {"x", "y"},
 PlotRange → All, ImageSize → 400]
```

Out[3]=



Consider the following quantiles:

In[4]:=
```
qs = {0.05, 0.25, 0.5, 0.75, 0.95};
```

We want to find curves that separate the data according the quantiles. Those curves are called “regression quantiles”.

Pretending that we do not know how the data is generated, just by looking at the plot we assume that the model for the data is

$$y = \beta_0 + \beta_1 x + \beta_2 \sqrt{x} + \beta_3 \log(x). \tag{11}$$

Let us put the model functions for the regression fit in the variable `funcs`:

In[5]:=
```
funcs = {1, x, √x , Log[x]};
```

Here we find the regression quantiles:

In[6]:=
```
qrFuncs = QuantileRegression[data, funcs, x, qs];
TableForm[List /@ qrFuncs]
```

Out[7]//TableForm=

$$5.0162 + 0.00727537\ \sqrt{x} + 1.70048 \times 10^{-12}\ x + 0.989398\ \text{Log}[x]$$

$$5.0659 + 5.7025 \times 10^{-12}\ \sqrt{x} + 5.79087 \times 10^{-13}\ x + 1.04655\ \text{Log}[x]$$

$$5.03547 + 3.97074 \times 10^{-14}\ \sqrt{x} + 5.20547 \times 10^{-12}\ x + 1.12658\ \text{Log}[x]$$

$$5.1561 + 4.77482 \times 10^{-12}\ \sqrt{x} + 0.000454095\ x + 1.18638\ \text{Log}[x]$$

$$4.77513 + 4.35188 \times 10^{-13}\ \sqrt{x} + 3.51738 \times 10^{-12}\ x + 1.44937\ \text{Log}[x]$$

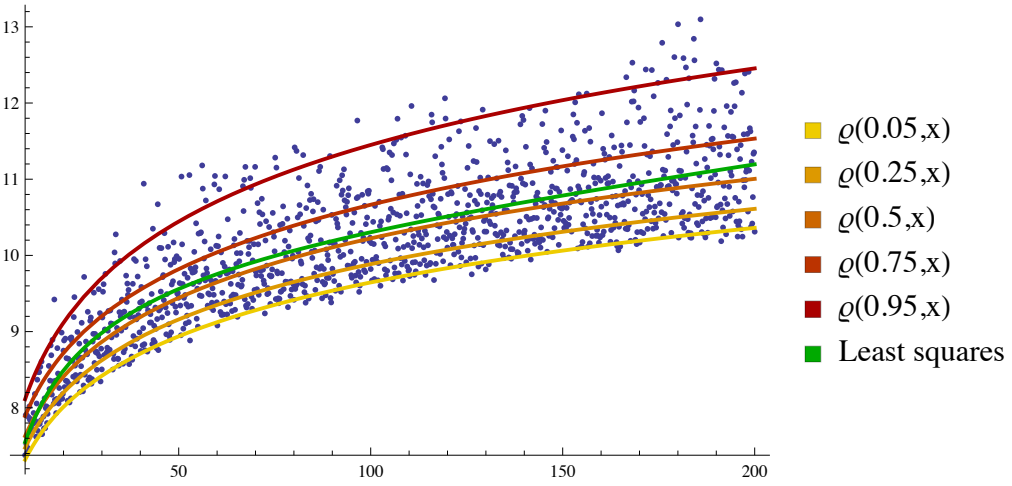
We also apply `Fit` to the data and the model functions in order to compare the regression quantiles with the least-squares regression fit:

In[8]:=
```
fFunc = Fit[data, funcs, x]
```

Out[8]=
$$4.189 - 0.484254\ \sqrt{x} + 0.0146854\ x + 2.06049\ \text{Log}[x]$$

Here is a plot that combines the found regression quantiles and least squares fit:

Out[10]=



Let us check how good the regression quantiles are for separating the data according to the quantiles they were computed for:

In[12]:=
```
tbl = Table[
    {qs[[i]], Length[Select[data, #[[2]] ≥ (qrFuncs[[i]] /. x → #[[1]]) &]] /
       Length[data] // N}, {i, Length[qs]}];
TableForm[tbl, TableHeadings → {None,
    {"quantile", "fraction\nabove"}}]
```

Out[13]//TableForm=

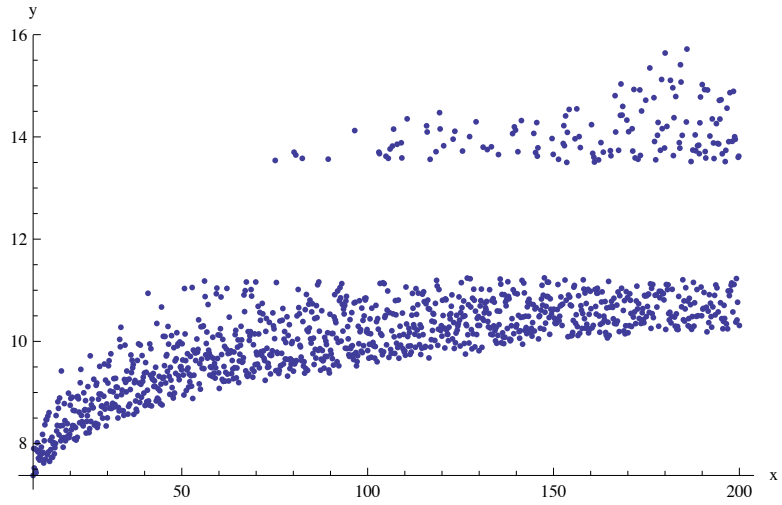
quantile	fraction above
0.05	0.949167
0.25	0.749167
0.5	0.500833
0.75	0.249167
0.95	0.0491667

Robustness

Let us demonstrate the robustness of the regression quantiles with the data of the previous example. Suppose that for some reason all the data *y*-values greater than 11.25 are altered by multiplying them with a some greater than 1 factor, say, $\alpha = 1.2$. Then the altered data looks like this:

In[14]:=
```
α = 1.2;
dataAlt = Map[If[#[[2]] > 11.25, {#[[1]], α #[[2]]}, #] &, data];
ListPlot[dataAlt, AxesLabel → {"x", "y"},
 PlotRange → All, ImageSize → 400]
```

Out[16]=



Let us compute the regression quantiles for the altered data:

In[17]:=
```
qrFuncsAlt = QuantileRegression[dataAlt, funcs, x, qs];
TableForm[List /@ qrFuncs]
```

Out[18]//TableForm=

$$5.0162 + 0.00727537\sqrt{x} + 1.70048\times10^{-12}\,x + 0.989398\,\text{Log}[x]$$

$$5.0659 + 5.7025\times10^{-12}\sqrt{x} + 5.79087\times10^{-13}\,x + 1.04655\,\text{Log}[x]$$

$$5.03547 + 3.97074\times10^{-14}\sqrt{x} + 5.20547\times10^{-12}\,x + 1.12658\,\text{Log}[x]$$

$$5.1561 + 4.77482\times10^{-12}\sqrt{x} + 0.000454095\,x + 1.18638\,\text{Log}[x]$$

$$4.77513 + 4.35188\times10^{-13}\sqrt{x} + 3.51738\times10^{-12}\,x + 1.44937\,\text{Log}[x]$$

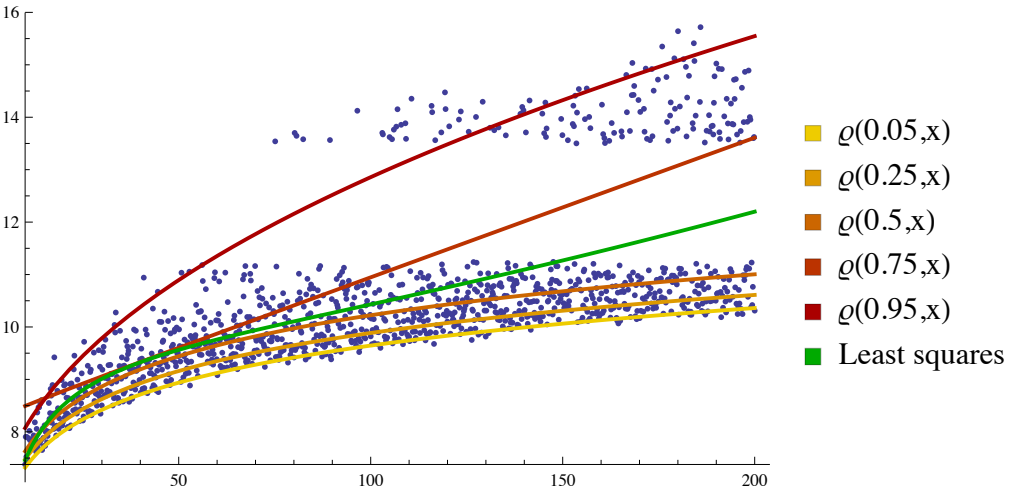
and let us also compute the least squares fit of the model (11):

In[19]:=
```
fFuncAlt = Fit[dataAlt, funcs, x]
```

Out[19]=
$$3.50155 - 1.4017\sqrt{x} + 0.0520594\,x + 3.41777\,\text{Log}[x]$$

Here is a plot that combines the functions found over the altered data:

Out[21]=



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

In[23]:=
```
qrFuncs[[1 ;; 3]]
```

Out[23]=
$$\left\{5.0162 + 0.00727537\sqrt{x} + 1.70048\times10^{-12}\,x + 0.989398\,\text{Log}[x],\right.$$
$$5.0659 + 5.7025\times10^{-12}\sqrt{x} + 5.79087\times10^{-13}\,x + 1.04655\,\text{Log}[x],$$
$$\left.5.03547 + 3.97074\times10^{-14}\sqrt{x} + 5.20547\times10^{-12}\,x + 1.12658\,\text{Log}[x]\right\}$$

In[24]:=
```
qrFuncsAlt[[1 ;; 3]]
```

Out[24]=
$$\left\{5.0162 + 0.00727522\sqrt{x} + 4.39848\times10^{-9}\,x + 0.989399\,\text{Log}[x],\right.$$
$$5.0659 + 1.28596\times10^{-11}\sqrt{x} + 2.08693\times10^{-12}\,x + 1.04655\,\text{Log}[x],$$
$$\left.5.03547 + 1.30156\times10^{-9}\sqrt{x} + 5.5865\times10^{-9}\,x + 1.12658\,\text{Log}[x]\right\}$$

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

In[25]:=
```
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[25]//TableForm=

quantile	fraction above
0.05	0.950833
0.25	0.75
0.5	0.499167
0.75	0.150833
0.95	0.0116667

Also we can see that the least squares fit of (11) has significantly changed:

In[26]:=
```
fFunc
```

Out[26]=
$$4.189 - 0.484254 \sqrt{x} + 0.0146854\, x + 2.06049\, \text{Log}[x]$$

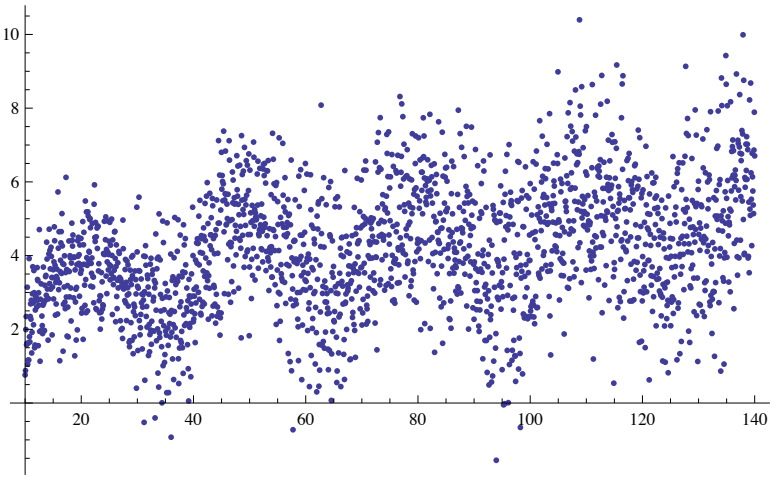
In[27]:=
```
fFuncAlt
```

Out[27]=
$$3.50155 - 1.4017 \sqrt{x} + 0.0520594\, x + 3.41777\, \text{Log}[x]$$

Data generated with Sin and noise

In[28]:=
```
With[{n = 2000, start = 10, end = 140},
 data = Table[{t,
    Sin[10 + t π / 15] + RandomReal[NormalDistribution[0, Log[t] / 3]]},
   {t, Rescale[Range[1, n], {1, n}, {start, end}]}];
 data[[All, 2]] = data[[All, 2]] + Log[data[[All, 1]]];
]
ListPlot[data, PlotRange → All, ImageSize → 400]
```

Out[29]=



We are going to use again the quantiles:

In[30]:=
```
qs = {0.05, 0.25, 0.5, 0.75, 0.95};
```

We can derive a guess about the data model by observing that we have peaks at $x = 50$, $x = 80$, $x = 110$. With these observations we make equations for Solve and take the second solution:

In[31]:=
```
Solve[{a > 0, Sin[x α] == Sin[α (x + 60)], Sin[x α] == Sin[α (x + 30)]}, α]
```

Out[31]=
```
Solve[{a > 0, Sin[1.2 x] == Sin[1.2 (60 + x)],
  Sin[1.2 x] == Sin[1.2 (30 + x)]}, 1.2]
```

In[32]:=
```
π / 15 // N
```

Out[32]=
```
0.20944
```

Next we need to find a guess for the phase. Again we use the second solution provided by Solve:

In[33]:=
```
Solve[{ϕ > 0, Sin[ϕ + 50 π / 15] == 1}, ϕ]
```

Out[33]=
$$\left\{\left\{\phi \to \text{ConditionalExpression}\left[\frac{1}{6}\,(-5\,\pi - 12\,\pi\,C[1]),\ C[1] \in \text{Integers}\ \&\&\ C[1] \le -1\right]\right\},\ \left\{\phi \to \text{ConditionalExpression}\left[\frac{1}{6}\,(7\,\pi - 12\,\pi\,C[1]),\ C[1] \in \text{Integers}\ \&\&\ C[1] \le 0\right]\right\}\right\}$$

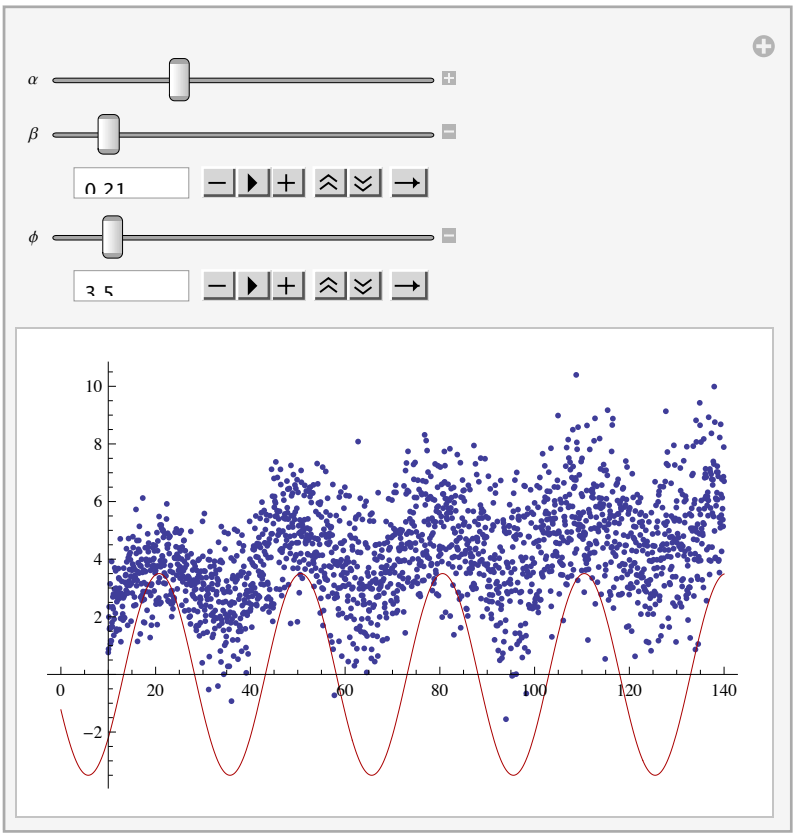
In[34]:=
```
1/6 7 π // N
```

Out[34]=
```
3.66519
```

Alternatively, we can simply use `Manipulate` and plot the data together with a model function subject to different parameters change.

In[35]:=
```
Manipulate[
 DynamicModule[{gr1, gr2},
  gr1 = ListPlot[data, PlotRange → All];
  gr2 = Plot[α Sin[ϕ + β x], {x, 0, 140}, PlotStyle → Darker[Red]];
  Show[{gr1, gr2}]
 ], {{α, 1}, 0.5, 10, 1}, {β, 0, 2, 0.01}, {ϕ, 0, 30, 0.25}]
```

Out[35]=



From the calculations we did so far we assume that the model for the data is

$y = \beta_0 + \beta_1 x + \beta_2 \text{Sin}[3.7 + x \pi / 15]$

Let us put the model functions for the regression fit in the variable `funcs`:

In[36]:=
```
funcs = {1, x, Sin[3.7 + x π / 15]};
```

We find the regression quantiles:

In[37]:=
```
qrFuncs = QuantileRegression[data, funcs, x, qs];
Grid[List /@ qrFuncs]
```

Out[38]=
$$1.2214 + 0.00888777\,x + 1.16952\,\text{Sin}\left[3.7 + \frac{\pi x}{15}\right]$$
$$2.21071 + 0.0134334\,x + 1.01833\,\text{Sin}\left[3.7 + \frac{\pi x}{15}\right]$$
$$2.77622 + 0.0178108\,x + 0.938605\,\text{Sin}\left[3.7 + \frac{\pi x}{15}\right]$$
$$3.45654 + 0.0213553\,x + 0.898343\,\text{Sin}\left[3.7 + \frac{\pi x}{15}\right]$$
$$4.51892 + 0.0268334\,x + 0.802801\,\text{Sin}\left[3.7 + \frac{\pi x}{15}\right]$$

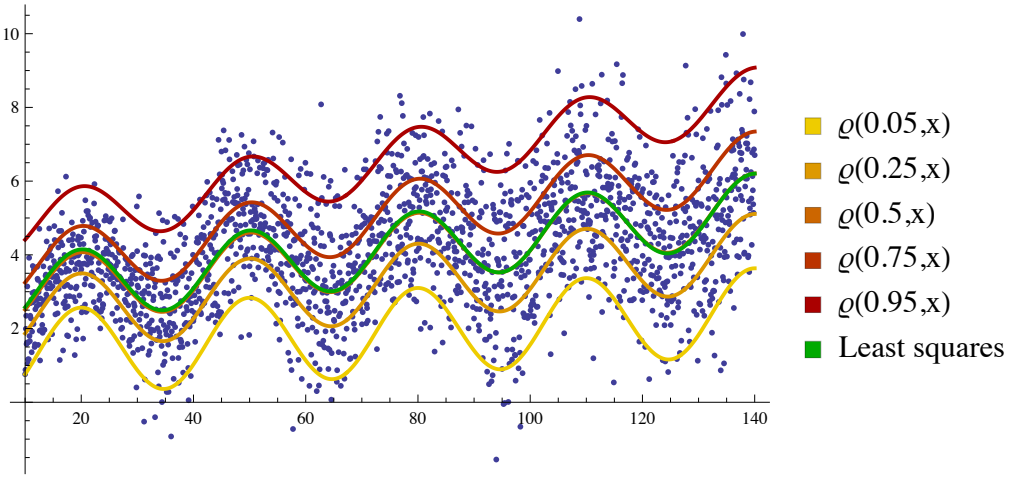
As in the previous example we also apply `Fit` to the data and the model functions in order to compare the regression quantiles with the least-squares regression fit:

In[39]:=
```
fFunc = Fit[data, funcs, x]
```

Out[39]=
$$2.85407 + 0.017137\,x + 0.951541\,\text{Sin}\left[3.7 + \frac{\pi x}{15}\right]$$

Here is a plot that combines the functions found:

Out[41]=



Let us check how good the regression quantiles are:

In[43]:=
```
tbl = Table[
    {qs[[i]], Length[Select[data, #[[2]] ≥ (qrFuncs[[i]] /. x → #[[1]]) &]] /
       Length[data] // N}, {i, Length[qs]}];
TableForm[tbl, TableHeadings → {None,
    {"quantile", "fraction\nabove"}}]
```

Out[44]//TableForm=

quantile	fraction above
0.05	0.95
0.25	0.75
0.5	0.5
0.75	0.25
0.95	0.0495

Profiling

It is interesting the see timing profile of the computations with `QuantileRegression` across two axes: (i) data size and (ii) number of functions to be fit.

First we need to choose a family or several families of test data. Also, since *Mathematica*'s function `LinearProgramming` has several methods it is a good idea to test with all of them. Here I am going to show results only with one family of data and two `LinearProgramming` methods. The data family is the skewed noise over a logarithmic curve used as an example above. The first `LinearProgramming` method is *Mathematica*'s (default) "InteriorPoint", the second method is "CLP" that uses the built-in COIN-OR CLP optimizer. I run the profiling tests using one quantile {0.5} and five quantiles {0.05, 0.25, 0.5, 0.75, 0.95}, which are shown in blue and red respectively. I also run tests with different number of model functions $\{1, x, \sqrt{x}, \text{Log}[x]\}$ and $\{1, x, \text{Log}[x]\}$ but there was no significant difference in the timings (less than 2%).

Test family functions definitions

In this sub-section are shown definitions of functions generating families of data sets.

```
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
  ];

Clear[SinWithUpwardTrend]
SinWithUpwardTrend[
   nPoints_Integer, start_?NumberQ, end_?NumberQ] :=
  Block[{data},
   data = Table[{t,
      Sin[t π / 15] + RandomReal[NormalDistribution[0, Log[t] / 3]]},
     {t, Rescale[Range[1, nPoints], {1, nPoints}, {start, end}]}];
   data[[All, 2]] = data[[All, 2]] + data[[All, 1]] ^ 1 / 6;
   data
  ];
```

```
Clear[SinWithParabolaTrend]
SinWithParabolaTrend[
   nPoints_Integer, start_?NumberQ, end_?NumberQ] :=
  Block[{data},
   data = Table[
     {t, Sin[t π / 10] + RandomReal[NormalDistribution[0, √t / 10]]},
     {t, Rescale[Range[1, nPoints], {1, nPoints}, {start, end}]}];
   data[[All, 2]] = data[[All, 2]] +
     (Mean[{start, end}] - data[[All, 1]])^2 / 300;
   data
  ];
```

LogarithmicCurveWithNoise 4 model functions with Method→ LinearProgramming

```
modelFuncs = {1, x, √x , Log[x]};
```

```
qs = {0.05`, 0.25`, 0.5`, 0.75`, 0.95`}
```

```
{0.05, 0.25, 0.5, 0.75, 0.95}
```

```
dataSets = LogarithmicCurveWithNoise[#, 10., 100.] & /@
   Range[500, 10 000, 500];
dataSets // Length
```

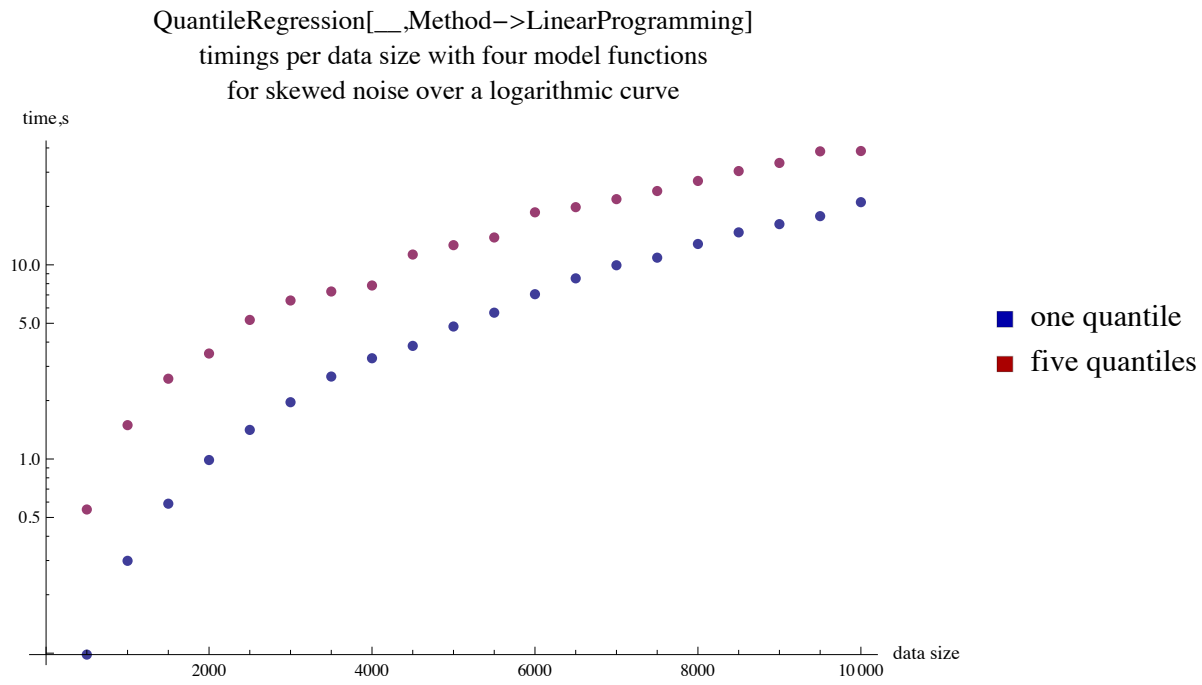
```
20
```

```
timingsLogarithmicCurveWithNoiseF4Q1 =
 Map[{Length[#], AbsoluteTiming[QuantileRegression[#, modelFuncs, x,
        {0.5}, Method → LinearProgramming];][[1]]} &, dataSets]
```

```
{{500, 0.098310}, {1000, 0.298962},
 {1500, 0.588580}, {2000, 0.987977}, {2500, 1.412103},
 {3000, 1.961045}, {3500, 2.658844}, {4000, 3.302903},
 {4500, 3.824936}, {5000, 4.812682}, {5500, 5.670178},
 {6000, 7.059653}, {6500, 8.519677}, {7000, 9.950919},
 {7500, 10.886583}, {8000, 12.807758}, {8500, 14.694095},
 {9000, 16.205932}, {9500, 17.815996}, {10 000, 21.027593}}
```

```
timingsLogarithmicCurveWithNoiseF4Q5 =
 Map[{Length[#], AbsoluteTiming[QuantileRegression[#, modelFuncs,
        x, qs, Method → LinearProgramming];][[1]]} &, dataSets]
```

```
{{500, 0.549529}, {1000, 1.494088},
 {1500, 2.591068}, {2000, 3.493207}, {2500, 5.204162},
 {3000, 6.551883}, {3500, 7.290538}, {4000, 7.828214},
 {4500, 11.299679}, {5000, 12.629508}, {5500, 13.832353},
 {6000, 18.640458}, {6500, 19.818611}, {7000, 21.795818},
 {7500, 23.999628}, {8000, 27.048659}, {8500, 30.410464},
 {9000, 33.484113}, {9500, 38.420541}, {10 000, 38.569162}}
```



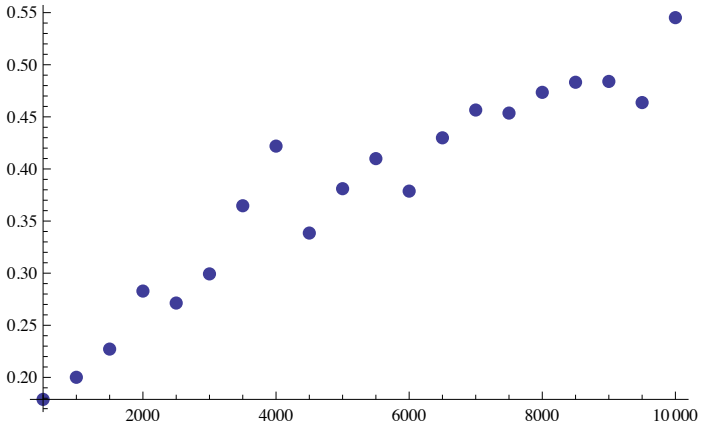
Average ratio between the execution times.

```
Mean[timingsLogarithmicCurveWithNoiseF4Q1[[All, 2]] /
  timingsLogarithmicCurveWithNoiseF4Q5[[All, 2]]]
```

```
0.380121
```

Ratios between the execution times:

```
ListPlot[Transpose[{timingsLogarithmicCurveWithNoiseF4Q1[[All, 1]],
    timingsLogarithmicCurveWithNoiseF4Q1[[All, 2]] /
     timingsLogarithmicCurveWithNoiseF4Q5[[All, 2]]}],
 PlotStyle → {PointSize[0.02]}, PlotRange → All]
```



LogarithmicCurveWithNoise 4 model functions with Method→ {LinearProgramming,Method→"CLP"}

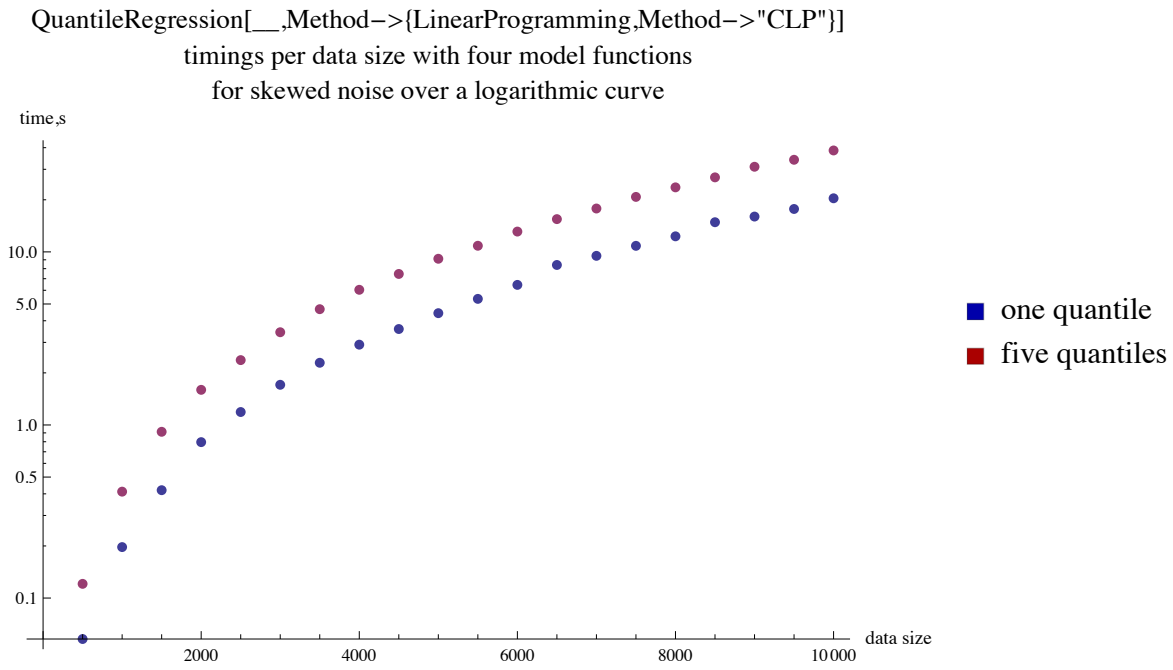
The same model functions and quantiles were used as in the previous sub-section.

```
timingsLogarithmicCurveWithNoiseCLPF4Q1 =
 Map[{Length[#], AbsoluteTiming[
      QuantileRegression[#, modelFuncs, x, {0.5}, Method →
         {LinearProgramming, Method → "CLP"}];][[1]]} &, dataSets]
```

```
{{500, 0.057988}, {1000, 0.196983},
 {1500, 0.419909}, {2000, 0.795889}, {2500, 1.187998},
 {3000, 1.706534}, {3500, 2.287483}, {4000, 2.910575},
 {4500, 3.583797}, {5000, 4.425360}, {5500, 5.352640},
 {6000, 6.446963}, {6500, 8.402463}, {7000, 9.487401},
 {7500, 10.838176}, {8000, 12.300428}, {8500, 14.839014},
 {9000, 15.998385}, {9500, 17.698214}, {10 000, 20.404420}}
```

```
timingsLogarithmicCurveWithNoiseCLPF4Q5 = Map[{Length[#],
    AbsoluteTiming[QuantileRegression[#, modelFuncs, x, qs, Method →
         {LinearProgramming, Method → "CLP"}];][[1]]} &, dataSets]
```

```
{{500, 0.120797}, {1000, 0.411864},
 {1500, 0.913235}, {2000, 1.595715}, {2500, 2.369918},
 {3000, 3.433828}, {3500, 4.666453}, {4000, 6.038684},
 {4500, 7.456060}, {5000, 9.125450}, {5500, 10.848903},
 {6000, 13.103446}, {6500, 15.463412}, {7000, 17.807215},
 {7500, 20.780654}, {8000, 23.596392}, {8500, 26.951688},
 {9000, 31.034202}, {9500, 34.052749}, {10 000, 38.549676}}
```



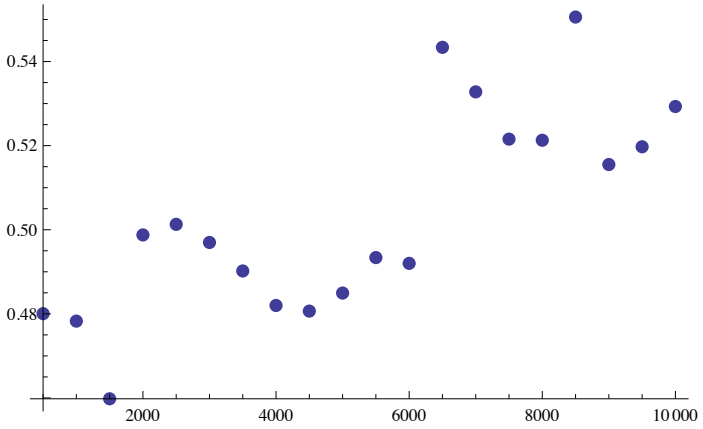
Average ratio between the execution times.

```
Mean[timingsLogarithmicCurveWithNoiseCLPF4Q1[[All, 2]] /
  timingsLogarithmicCurveWithNoiseCLPF4Q5[[All, 2]]]
```

```
0.50362
```

Ratios between the execution times:

```
ListPlot[
 Transpose[{timingsLogarithmicCurveWithNoiseCLPF4Q1[[All, 1]],
   timingsLogarithmicCurveWithNoiseCLPF4Q1[[All, 2]] /
    timingsLogarithmicCurveWithNoiseCLPF4Q5[[All, 2]]}],
 PlotStyle → {PointSize[0.02]}, PlotRange → All]
```



Notes

It is interesting to note that the average ratio of the timings with 1 vs. 5 quantiles is 0.38 for "InteriorPoint" and 0.5 for "CLP".

Careful with "CLP"

During the profiling experiments with some of data families was observed that "CLP" gives curves that do not follow the data closely. Below is shown such a computation with both "InteriorPoint" and "CLP", the former looks good, the latter does not.

```
modelFuncs =
  {1, x, Sin[x π / 10], -Sin[x π / 10], -Sin[x π / 200], Sin[x π / 200]};

qs = {0.05`, 0.25`, 0.5`, 0.75`, 0.95`}
```

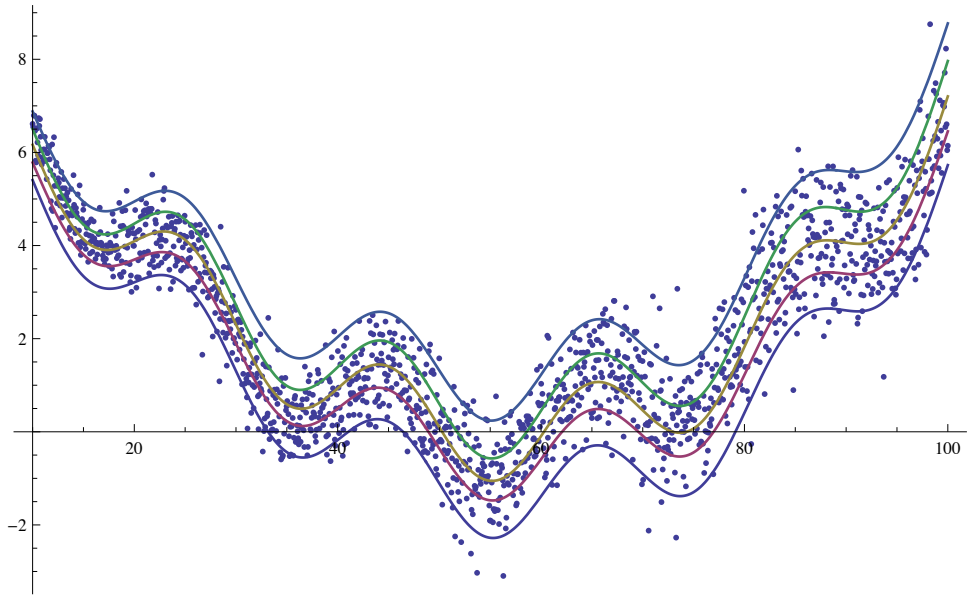
```
{0.05, 0.25, 0.5, 0.75, 0.95}
```

```
dataSet = SinWithParabolaTrend[1500, 10., 100.];

AbsoluteTiming[
 qrFuncs = QuantileRegression[dataSet,
   modelFuncs, x, qs, Method → {LinearProgramming}]
]
```

$$\left\{4.363524,\ \left\{7.74411 + 0.3587\,x - 37.887\,\text{Sin}\left[\frac{\pi x}{200}\right] + 0.939716\,\text{Sin}\left[\frac{\pi x}{10}\right],\right.\right.$$
$$8.00623 + 0.351127\,x - 36.6666\,\text{Sin}\left[\frac{\pi x}{200}\right] + 0.908217\,\text{Sin}\left[\frac{\pi x}{10}\right],$$
$$8.37742 + 0.359772\,x - 37.1517\,\text{Sin}\left[\frac{\pi x}{200}\right] + 0.966431\,\text{Sin}\left[\frac{\pi x}{10}\right],$$
$$8.67913 + 0.365803\,x - 37.2965\,\text{Sin}\left[\frac{\pi x}{200}\right] + 1.00878\,\text{Sin}\left[\frac{\pi x}{10}\right],$$
$$\left.\left.8.95491 + 0.362633\,x - 36.4475\,\text{Sin}\left[\frac{\pi x}{200}\right] + 0.94464\,\text{Sin}\left[\frac{\pi x}{10}\right]\right\}\right\}$$

```
grData = ListPlot[dataSet, PlotRange → All];
grFit = Plot[Evaluate[MapThread[Tooltip[#1, #2] &, {qrFuncs, qs}]],
   {x, Min[dataSet[[All, 1]]], Max[dataSet[[All, 1]]]},
   PlotStyle → AbsoluteThickness[1.4]];
Show[{grData, grFit}, ImageSize → 500]
```




```
AbsoluteTiming[
 qrFuncs = QuantileRegression[dataSet, modelFuncs, x, qs, Method →
    {LinearProgramming, Method → "CLP", Tolerance → 10^-14.0}]
]
```

$$\left\{1.158118,\ \left\{-3.09553 + 3.15342\ \text{Sin}\left[\frac{\pi x}{200}\right] + 0.968002\ \text{Sin}\left[\frac{\pi x}{10}\right],\right.$$

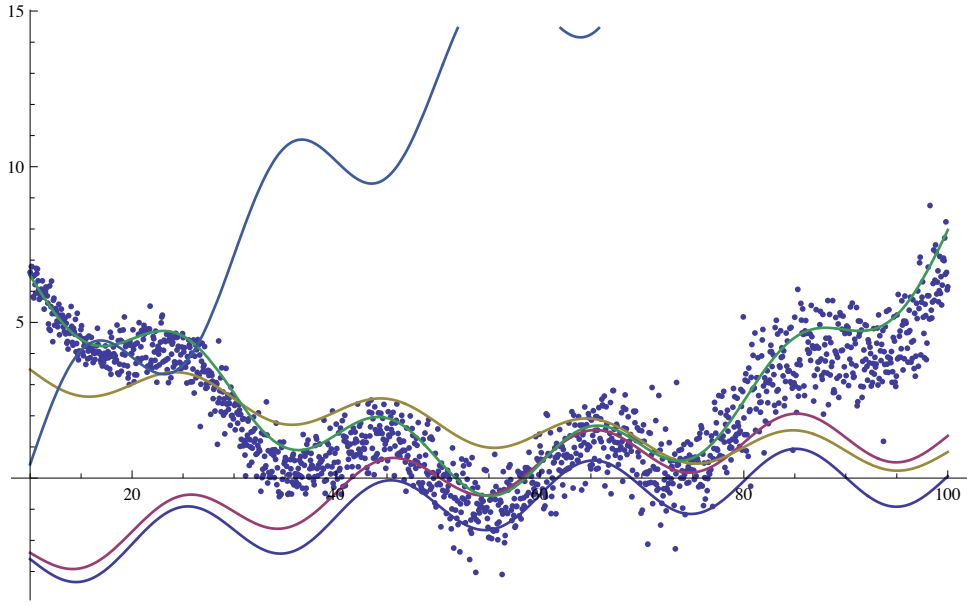
$$-3.09553 + 4.4509\ \text{Sin}\left[\frac{\pi x}{200}\right] + 0.835651\ \text{Sin}\left[\frac{\pi x}{10}\right],$$

$$3.97571 - 3.13847\ \text{Sin}\left[\frac{\pi x}{200}\right] + 0.608836\ \text{Sin}\left[\frac{\pi x}{10}\right],$$

$$8.67913 + 0.365803\ x - 37.2965\ \text{Sin}\left[\frac{\pi x}{200}\right] + 1.00878\ \text{Sin}\left[\frac{\pi x}{10}\right],$$

$$\left.\left.-3.09553 + 22.5929\ \text{Sin}\left[\frac{\pi x}{200}\right] - 1.91918\ \text{Sin}\left[\frac{\pi x}{10}\right]\right\}\right\}$$

```
grData = ListPlot[dataSet, PlotRange → All];
grFit = Plot[Evaluate[MapThread[Tooltip[#1, #2] &, {qrFuncs, qs}]],
   {x, Min[dataSet[[All, 1]]], Max[dataSet[[All, 1]]]},
   PlotStyle → AbsoluteThickness[1.4]];
Show[{grData, grFit}, ImageSize → 500]
```



References

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

[2] 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 .

[3] Wikipedia, Quantile regression, http://en.wikipedia.org/wiki/Quantile_regression .

[4] Brian Cade, Barry Noon, “A gentle introduction to quantile regression for ecologists”, Front. Ecol. Environ. 1(8), 2003, pp. 412–420.