I have some data
<data><y>9</y><y>15</y><y>11</y><y>6</y><y>5</y><y>10</y><y>8</y><y>8</y><y>3</y><y>12</y><y>14</y><y>9</y><y>16</y><y>14</y></data>

And I want to look at the data as a histogram.
1 &nbsp; &nbsp;Like this

![Histogram example](https://homepages.cwi.nl/~steven/Talks/2018/06-10-markup/histogram.xhtml)
1 &nbsp; &nbsp; Histogram

Uses SVG

100×100 space

There are n values, so each rectangle is 100/n wide.

The range of the data is max - min

The vertical space is therefore divided over 100/range, which we'll call vscale

The height of the rectangle for each value v is v × vscale.
The SVG will look something like this:

```xml
<svg ...>
  <xf:repeat bind="values">
    <rect width="{...}"
      height="{...}"
      x="{...}"
      y="{...}"
    />
  </xf:repeat>
</svg>
```

Note: SVG coordinate system is 'upside down'. (0,0) is top left, and goes downwards and to the right.
Horizontal space

We load the data:

```xml
<instance id="data" src="data.xml"/>
```

Identify the values:

```xml
<bind ref="instance('data')/y" id="values"/>
```

The first thing we need is how many data values there are:

```xml
<bind ref="n" calculate="count(bind('values'))"/>
```

The width of each rectangle is:

```xml
<bind ref="width" calculate="100 div ..n" type="double"/>
```
1 &nbsp; &nbsp;SVG

Now we can fill in the horizontal values:

```xml
<svg ...>
  <xf:repeat bind="values">
    <rect width="{width}"
         height="{...}"
         x="{(position()-1) * width}"
         y="{...}"/>
  </xf:repeat>
</svg>
```
1 &nbsp; &nbsp;**Vertical space**

Get the minimum and maximum values:

```xml
<bind ref="min" calculate="min(bind('values'))"/>
<bind ref="max" calculate="max(bind('values'))"/>
```

Calculate the range minimum and maximum, and the range:

```xml
<bind ref="rmin" calculate="min(0, ../min)"/>
<bind ref="rmax" calculate="max(0, ../max)"/>
<bind ref="range" calculate="../rmax - ../rmin"/>
```

And so the vertical scale is

```xml
<bind ref="vscale" calculate="100 div ../range" type="double"/>
```

Except if the range is zero (i.e. all values are zero), so we actually need

```xml
calculate="if(../range = 0, 1, 100 div ../range)"
```
1nbsp; &nbsp;Height

Ideally we would just say

\[
\text{height} = \{. \times \text{vscale}\}
\]

Incredibly, although SVG allows lines to have negative length, rectangles are not allowed to have negative height.

So we have to do some extra work.

\[
\text{height} = \{|\text{abs}(.)\} \times \text{vscale}\}
\]
1nbsp; &nbsp;Nearly there

So now we have a row of rectangles, of the correct height, of the correct width, at the correct horizontal position:
1 &nbsp; &nbsp;**Vertical position**

Currently all rectangles start at SVG’s vertical zero, and the other end of the longest (positive) bar is where zero really ought to be.

That value is $r_{\text{max}}$, so all we have to do is push all the bars down by the difference between its value and $r_{\text{max}}$. All negative values just have to start at $r_{\text{max}}$:

$$y = \{ \text{vscale} \times \text{if}(. \lt 0, r_{\text{max}}, r_{\text{max}} - .) \}$$

Giving the final result:
1&nbsp;&nbsp;SVG

```xml
<svg ...>
  <xf:repeat bind="values">
    <rect width="{width}"
      height="{vscale * abs(.)}"
      x="{(position() - 1) * width}"
      y="{vscale * if(. &lt; 0, rmax, rmax - .)}"
    />
  </xf:repeat>
</svg>
```
Result (about 25 lines of XForms)

Source