**Description**

`pctile` creates a new variable containing the percentiles of `exp`, where the expression `exp` is typically just another variable.

`xtile` creates a new variable that categorizes `exp` by its quantiles. If the `cutpoints(varname)` option is specified, it categorizes `exp` using the values of `varname` as category cutpoints. For example, `varname` might contain percentiles of another variable, generated by `pctile`.

`pctile` is a programmer’s command that computes up to 4,096 percentiles and places the results in `r()`; see [U] 18.8 Accessing results calculated by other programs. `summarize`, `detail` computes some percentiles (1, 5, 10, 25, 50, 75, 90, 95, and 99th); see [R] `summarize`.

**Quick start**

Create `qrt1` containing the quartiles of `v`

```
pctile qrt1 = v, nq(4)
```

As above, and create `percent` containing the percentages

```
pctile qrt1 = v, nq(4) genp(percent)
```

As above, but apply sampling weights `wvar1`

```
pctile qrt1 = v [pweight=wvar1], nq(4) genp(percent)
```

Create `dec1` containing the deciles of `v`

```
pctile dec1 = v, nq(10)
```

As above, but create `dec2` indicating to which decile each observation belongs

```
xtile dec2 = v, nq(10)
```

As above, but apply frequency weights `wvar2`

```
xtile dec2 = v [fweight=wvar2], nq(10)
```

Compute the 10th and 90th percentiles, and store them in `r(r1)` and `r(r2)`

```
_pctile v, percentiles(10 90)
```

**Menu**

`pctile`

Statistics > Summaries, tables, and tests > Summary and descriptive statistics > Create variable of percentiles

`xtile`

Statistics > Summaries, tables, and tests > Summary and descriptive statistics > Create variable of quantiles
Syntax

Create variable containing percentiles

\[ \text{pctile [type] newvar = exp [if] [in] [weight] [ , pctile_options]} \]

Create variable containing quantile categories

\[ \text{xtile newvar = exp [if] [in] [weight] [ , xtile_options]} \]

Compute percentiles and store them in \( r() \)

\[ \_\text{pctile varname [if] [in] [weight] [ , _\text{pctile_options}]} \]

\[
\begin{array}{ll}
\text{pctile_options} & \text{Description} \\
\hline
\text{nquantiles(\#)} & \text{number of quantiles; default is nquantiles(2)} \\
\text{genp(newvar)} & \text{generate newvar, variable containing percentages} \\
\text{altdef} & \text{use alternative formula for calculating percentiles} \\
\hline
\end{array}
\]

\[
\begin{array}{ll}
\text{xtile_options} & \text{Description} \\
\hline
\text{nquantiles(\#)} & \text{number of quantiles; default is nquantiles(2)} \\
\text{cutpoints(varname)} & \text{use values of varname as cutpoints} \\
\text{altdef} & \text{use alternative formula for calculating percentiles} \\
\hline
\end{array}
\]

\[
\begin{array}{ll}
\text{pctile_options} & \text{Description} \\
\hline
\text{nquantiles(\#)} & \text{number of quantiles; default is nquantiles(2)} \\
\text{percentiles(numlist)} & \text{calculate percentiles corresponding to the specified percentages} \\
\text{altdef} & \text{use alternative formula for calculating percentiles} \\
\hline
\end{array}
\]

\text{aweights, fweights, and pweights are allowed (see [U] 11.1.6 weight), except when the altdef option is specified, in which case no weights are allowed.}

Options

\[
\begin{array}{ll}
\text{Main} \\
\text{nquantiles(\#)} & \text{specifies the number of quantiles. It computes percentiles corresponding to percentages 100 \( k/m \) for \( k = 1, 2, \ldots, m - 1 \), where \( m = \# \). For example, nquantiles(10) requests that the 10th, 20th, \ldots, 90th percentiles be computed. The default is nquantiles(2); that is, the median is computed.} \\
\text{genp(newvar)} & \text{(pctile only) specifies a new variable to be generated containing the percentages corresponding to the percentiles.} \\
\end{array}
\]
altdef uses an alternative formula for calculating percentiles. The default method is to invert the empirical distribution function by using averages, \((x_i + x_{i+1})/2\), where the function is flat (the default is the same method used by `summarize'; see [R] `summarize'). The alternative formula uses an interpolation method. See `Methods and formulas' at the end of this entry. Weights cannot be used when `altdef' is specified.

cutpoints(varname) (xtile only) requests that `xtile' use the values of `varname', rather than quantiles, as cutpoints for the categories. All values of `varname' are used, regardless of any if or in restriction; see the technical note in the xtile section below.

percentiles(numlist) (pctile only) requests percentiles corresponding to the specified percentages. Percentiles are placed in `r(r1)', `r(r2)', ..., etc. For example, `percentiles(10(20)90)' requests that the 10th, 30th, 50th, 70th, and 90th percentiles be computed and placed into `r(r1)', `r(r2)', `r(r3)', `r(r4)', and `r(r5)'. Up to 4,096 (inclusive) percentiles can be requested. See [P] `numlist' for the syntax of a numlist.

Remarks and examples

Remarks are presented under the following headings:

`pctile'
`xtile'
`-_pctile'

pctile

`pctile' creates a new variable containing percentiles. You specify the number of quantiles that you want, and `pctile' computes the corresponding percentiles. Here we use Stata's `auto' dataset and compute the deciles of `mpg':

```
. use https://www.stata-press.com/data/r16/auto
(1978 Automobile Data)
. pctile pct = mpg, nq(10)
. list pct in 1/10
```

```
Pct
1. 14
2. 17
3. 18
4. 19
5. 20
6. 22
7. 24
8. 25
9. 29
10.
```
If we use the `genp()` option to generate another variable with the corresponding percentages, it is easier to distinguish between the percentiles.

```
    . drop pct
    . pctlie pct = mpg, nq(10) genp(percent)
    . list percent pct in 1/10

<table>
<thead>
<tr>
<th>percent</th>
<th>pct</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>10</td>
</tr>
<tr>
<td>2.</td>
<td>20</td>
</tr>
<tr>
<td>3.</td>
<td>30</td>
</tr>
<tr>
<td>4.</td>
<td>40</td>
</tr>
<tr>
<td>5.</td>
<td>50</td>
</tr>
<tr>
<td>6.</td>
<td>60</td>
</tr>
<tr>
<td>7.</td>
<td>70</td>
</tr>
<tr>
<td>8.</td>
<td>80</td>
</tr>
<tr>
<td>9.</td>
<td>90</td>
</tr>
<tr>
<td>10.</td>
<td>.</td>
</tr>
</tbody>
</table>
```

```
    . summarize mpg, detail

Mileage (mpg)

    Percentiles  Smallest
    1%           12  12  Obs  74
    5%           14  14
    10%          14  14  Sum of Wgt.  74
    25%          18  14
    50%          20
    75%          25  34
    90%          29  35
    95%          34  35
    99%          41  41

Largest  Std. Dev.  5.785503
Variance  33.47205
Skewness  .9487176
Kurtosis  3.975005

```

summarize, detail can calculate only these particular percentiles. The `pctlie` and `_pctlie` commands allow you to compute any percentile.
Weights can be used with `pctile`, `xtile`, and `_pctile`:

```
drop pct percent
.pctile pct = mpg [w=weight], nq(10) genp(percent)
(analytic weights assumed)
.list percent pct in 1/10
```

<table>
<thead>
<tr>
<th>percent</th>
<th>pct</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10</td>
</tr>
<tr>
<td>2</td>
<td>20</td>
</tr>
<tr>
<td>3</td>
<td>30</td>
</tr>
<tr>
<td>4</td>
<td>40</td>
</tr>
<tr>
<td>5</td>
<td>50</td>
</tr>
<tr>
<td>6</td>
<td>60</td>
</tr>
<tr>
<td>7</td>
<td>70</td>
</tr>
<tr>
<td>8</td>
<td>80</td>
</tr>
<tr>
<td>9</td>
<td>90</td>
</tr>
<tr>
<td>10</td>
<td>.</td>
</tr>
</tbody>
</table>

The result is the same, no matter which weight type you specify—`aweight`, `fweight`, or `pweight`.

**xtile**

`xtile` creates a categorical variable that contains categories corresponding to quantiles. We illustrate this with a simple example. Suppose that we have a variable, `bp`, containing blood pressure measurements:

```
use https://www.stata-press.com/data/r16/bp1, clear
.list
```

<table>
<thead>
<tr>
<th>bp</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td>6</td>
</tr>
<tr>
<td>7</td>
</tr>
<tr>
<td>8</td>
</tr>
<tr>
<td>9</td>
</tr>
<tr>
<td>10</td>
</tr>
<tr>
<td>11</td>
</tr>
</tbody>
</table>
xtile can be used to create a variable, quart, that indicates the quartiles of bp.

```stata
    . xtile quart = bp, nq(4)
    . list bp quart
```

<table>
<thead>
<tr>
<th>bp</th>
<th>quart</th>
</tr>
</thead>
<tbody>
<tr>
<td>98</td>
<td>1</td>
</tr>
<tr>
<td>100</td>
<td>1</td>
</tr>
<tr>
<td>104</td>
<td>1</td>
</tr>
<tr>
<td>110</td>
<td>2</td>
</tr>
<tr>
<td>120</td>
<td>2</td>
</tr>
<tr>
<td>120</td>
<td>2</td>
</tr>
<tr>
<td>120</td>
<td>2</td>
</tr>
<tr>
<td>120</td>
<td>2</td>
</tr>
<tr>
<td>125</td>
<td>3</td>
</tr>
<tr>
<td>130</td>
<td>4</td>
</tr>
<tr>
<td>132</td>
<td>4</td>
</tr>
</tbody>
</table>

The categories created are

\[ (-\infty, x_{25}], (x_{25}, x_{50}], (x_{50}, x_{75}], (x_{75}, +\infty) \]

where \( x_{25}, x_{50}, \) and \( x_{75} \) are, respectively, the 25th, 50th (median), and 75th percentiles of bp. We could use the `pctile` command to generate these percentiles:

```stata
    . pctile pct = bp, nq(4) genp(percent)
    . list bp quart percent pct
```

<table>
<thead>
<tr>
<th>bp</th>
<th>quart</th>
<th>percent</th>
<th>pct</th>
</tr>
</thead>
<tbody>
<tr>
<td>98</td>
<td>1</td>
<td>25</td>
<td>104</td>
</tr>
<tr>
<td>100</td>
<td>1</td>
<td>50</td>
<td>120</td>
</tr>
<tr>
<td>104</td>
<td>1</td>
<td>75</td>
<td>125</td>
</tr>
<tr>
<td>110</td>
<td>2</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>120</td>
<td>2</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>120</td>
<td>2</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>120</td>
<td>2</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>125</td>
<td>3</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>130</td>
<td>4</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>132</td>
<td>4</td>
<td>.</td>
<td>.</td>
</tr>
</tbody>
</table>

xtile can categorize a variable on the basis of any set of cutpoints, not just percentiles. Suppose that we wish to create the following categories for blood pressure:

\[ (-\infty, 100], (100, 110], (110, 120], (120, 130], (130, +\infty) \]
To do this, we simply create a variable containing the cutpoints,

```
. input class
    class
  1. 100
  2. 110
  3. 120
  4. 130
  5. end
```

and then use `xtile` with the `cutpoints()` option:

```
. xtile category = bp, cutpoints(class)
. list bp class category, sepby(category)

<table>
<thead>
<tr>
<th>bp</th>
<th>class</th>
<th>category</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>98</td>
<td>100</td>
</tr>
<tr>
<td>2</td>
<td>100</td>
<td>110</td>
</tr>
<tr>
<td>3</td>
<td>104</td>
<td>120</td>
</tr>
<tr>
<td>4</td>
<td>110</td>
<td>130</td>
</tr>
<tr>
<td>5</td>
<td>120</td>
<td>.</td>
</tr>
<tr>
<td>6</td>
<td>120</td>
<td>.</td>
</tr>
<tr>
<td>7</td>
<td>120</td>
<td>.</td>
</tr>
<tr>
<td>8</td>
<td>120</td>
<td>.</td>
</tr>
<tr>
<td>9</td>
<td>125</td>
<td>.</td>
</tr>
<tr>
<td>10</td>
<td>130</td>
<td>.</td>
</tr>
<tr>
<td>11</td>
<td>132</td>
<td>.</td>
</tr>
</tbody>
</table>
```

The cutpoints can, of course, come from anywhere. They can be the quantiles of another variable or the quantiles of a subgroup of the variable. Suppose that we had a variable, `case`, that indicated whether an observation represented a case (`case = 1`) or control (`case = 0`).

```
. use https://www.stata-press.com/data/r16/bp2, clear
. list in 1/11, sep(4)

<table>
<thead>
<tr>
<th>bp</th>
<th>case</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>98</td>
</tr>
<tr>
<td>2</td>
<td>100</td>
</tr>
<tr>
<td>3</td>
<td>104</td>
</tr>
<tr>
<td>4</td>
<td>110</td>
</tr>
<tr>
<td>5</td>
<td>120</td>
</tr>
<tr>
<td>6</td>
<td>120</td>
</tr>
<tr>
<td>7</td>
<td>120</td>
</tr>
<tr>
<td>8</td>
<td>120</td>
</tr>
<tr>
<td>9</td>
<td>125</td>
</tr>
<tr>
<td>10</td>
<td>130</td>
</tr>
<tr>
<td>11</td>
<td>132</td>
</tr>
</tbody>
</table>
```
We can categorize the cases on the basis of the quantiles of the controls. To do this, we first generate a variable, `pct`, containing the percentiles of the controls’ blood pressure data:

```
pctile pct = bp if case==0, nq(4)
list pct in 1/4
```

<table>
<thead>
<tr>
<th>pct</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. 104</td>
</tr>
<tr>
<td>2. 117</td>
</tr>
<tr>
<td>3. 124</td>
</tr>
<tr>
<td>4. .</td>
</tr>
</tbody>
</table>

Then we use these percentiles as cutpoints to classify `bp`: for all subjects.

```
xtile category = bp, cutpoints(pct)
gsort -case bp
list bp case category in 1/11, sepby(category)
```

<table>
<thead>
<tr>
<th>bp</th>
<th>case</th>
<th>category</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. 98</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2. 100</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3. 104</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>4. 110</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>5. 120</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>6. 120</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>7. 120</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>8. 120</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>9. 125</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>10. 130</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>11. 132</td>
<td>1</td>
<td>4</td>
</tr>
</tbody>
</table>

⚠️ Technical note

In the last example, if we wanted to categorize only cases, we could have issued the command

```
xtile category = bp if case==1, cutpoints(pct)
```

Most Stata commands follow the logic that using an `if exp` is equivalent to dropping observations that do not satisfy the expression and running the command. This is not true of `xtile` when the `cutpoints()` option is used. (When the `cutpoints()` option is not used, the standard logic is true.) `xtile` uses all nonmissing values of the `cutpoints()` variable whether or not these values belong to observations that satisfy the `if` expression.

If you do not want to use all the values in the `cutpoints()` variable as cutpoints, simply set the ones that you do not need to missing. `xtile` does not care about the order of the values or whether they are separated by missing values.
Technical note

Quantiles are not always unique. If we categorize our blood pressure data by quintiles rather than quartiles, we get

```stata
use https://www.stata-press.com/data/r16/bp1, clear
.xtile quint = bp, nq(5)
.pctile pct = bp, nq(5) genp(percent)
.list bp quint pct percent, sepby(quint)
```

<table>
<thead>
<tr>
<th></th>
<th>bp</th>
<th>quint</th>
<th>pct</th>
<th>percent</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>98</td>
<td>1</td>
<td>104</td>
<td>20</td>
</tr>
<tr>
<td>2</td>
<td>100</td>
<td>1</td>
<td>120</td>
<td>40</td>
</tr>
<tr>
<td>3</td>
<td>104</td>
<td>1</td>
<td>120</td>
<td>60</td>
</tr>
<tr>
<td>4</td>
<td>110</td>
<td>2</td>
<td>125</td>
<td>80</td>
</tr>
<tr>
<td>5</td>
<td>120</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>120</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>120</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>120</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>125</td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>130</td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>132</td>
<td>5</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The 40th and 60th percentile are the same; they are both 120. When two (or more) percentiles are the same, they are given the lower category number.

_pctile

_pctile is a programmer’s command. It computes percentiles and stores them in `r(); see [U] 18.8 Accessing results calculated by other programs.
You can use _pctile to compute quantiles, just as you can with pctile:

```
. use https://www.stata-press.com/data/r16/auto, clear
(1978 Automobile Data)
. _pctile weight, nq(10)
. return list
  scalars:
    r(r1) = 2020
    r(r2) = 2160
    r(r3) = 2520
    r(r4) = 2730
    r(r5) = 3190
    r(r6) = 3310
    r(r7) = 3420
    r(r8) = 3700
    r(r9) = 4060
```

The percentiles() option (abbreviation p()) can be used to compute any percentile you wish:

```
. _pctile weight, p(10, 33.333, 45, 50, 55, 66.667, 90)
. return list
  scalars:
    r(r1) = 2020
    r(r2) = 2640
    r(r3) = 2830
    r(r4) = 3190
    r(r5) = 3250
    r(r6) = 3400
    r(r7) = 4060
```

_pctile, pctile, and xtile each have an option that uses an alternative definition of percentiles, based on an interpolation scheme; see Methods and formulas below.

```
. _pctile weight, p(10, 33.333, 45, 50, 55, 66.667, 90) altdef
. return list
  scalars:
    r(r1) = 2005
    r(r2) = 2639.985
    r(r3) = 2830
    r(r4) = 3190
    r(r5) = 3252.5
    r(r6) = 3400.005
    r(r7) = 4060
```

The default formula inverts the empirical distribution function. The default formula is more commonly used, although some consider the “alternative” formula to be the standard definition. One drawback of the alternative formula is that it does not have an obvious generalization to noninteger weights.

FAQ Technical note

summarize, detail computes the 1st, 5th, 10th, 25th, 50th (median), 75th, 90th, 95th, and 99th percentiles. There is no real advantage in using _pctile to compute these percentiles. Both summarize, detail and _pctile use the same internal code. _pctile is slightly faster because summarize, detail computes a few extra things. The value of _pctile is its ability to compute percentiles other than these standard ones.
pctile and _pctile store the following in r():

Scalars
\( r(\#) \) value of \#-requested percentile

Methods and formulas

The default formula for percentiles is as follows: Let \( x(j) \) refer to the \( x \) in ascending order for \( j = 1, 2, \ldots, n \). Let \( w(j) \) refer to the corresponding weights of \( x(j) \); if there are no weights, \( w(j) = 1 \). Let \( N = \sum_{j=1}^{n} w(j) \).

To obtain the \( p \)th percentile, which we will denote as \( x[p] \), let \( P = Np/100 \), and let

\[
W(i) = \sum_{j=1}^{i} w(j)
\]

Find the first index, \( i \), such that \( W(i) > P \). The \( p \)th percentile is then

\[
x[p] = \begin{cases} 
\frac{x(i-1) + x(i)}{2} & \text{if } W(i-1) = P \\
x(i) & \text{otherwise}
\end{cases}
\]

When the `altdef` option is specified, the following alternative definition is used. Here weights are not allowed.

Let \( i \) be the integer floor of \( (n+1)p/100 \); that is, \( i \) is the largest integer \( i \leq (n+1)p/100 \). Let \( h \) be the remainder \( h = (n+1)p/100 - i \). The \( p \)th percentile is then

\[
x[p] = (1 - h)x(i) + hx(i+1)
\]

where \( x(0) \) is taken to be \( x(1) \) and \( x(n+1) \) is taken to be \( x(n) \).

`xtile` produces the categories

\([-\infty, x[p_1]], (x[p_1], x[p_2]], \ldots, (x[p_{m-2}], x[p_{m-1}]], (x[p_{m-1}], +\infty)\]

numbered, respectively, 1, 2, \ldots, \( m \), based on the \( m \) quantiles given by the \( p_k \)th percentiles, where \( p_k = 100k/m \) for \( k = 1, 2, \ldots, m - 1 \).

If \( x[p_{k-1}] = x[p_k] \), the \( k \)th category is empty. All elements \( x = x[p_{k-1}] = x[p_k] \) are put in the \((k-1)\)th category: \((x[p_{k-2}], x[p_{k-1}]]\).

If `xtile` is used with the `cutpoints(varname)` option, the categories are

\([-\infty, y(1)], (y(1), y(2)], \ldots, (y(m-1), y(m]], (y(m), +\infty)\]

and they are numbered, respectively, 1, 2, \ldots, \( m + 1 \), based on the \( m \) nonmissing values of `varname`: \( y(1), y(2), \ldots, y(m) \).
Acknowledgment

xtile is based on a command originally posted on Statalist (see [U] 3.2.4 The Stata Forum) by Philip Ryan of the Discipline of Public Health at the University of Adelaide, Australia.

Also see

[R] centile — Report centile and confidence interval
[R] summarize — Summary statistics
[U] 18.8 Accessing results calculated by other programs