Title

pctile — Create variable containing percentiles

Syntax

Create variable containing percentiles

`pctile` [*type*] *newvar* `=` *exp* [*if*] [*in*] [*weight*] [, *pctile_options*]

Create variable containing quantile categories

`xtile` *newvar* `=` *exp* [*if*] [*in*] [*weight*] [, *xtile_options*]

Compute percentiles and store them in r()

`_pctile` *varname* [*if*] [*in*] [*weight*] [, *_pctile_options*]

pctile_options	Description
Main	
`nquantiles(#)`	number of quantiles; default is `nquantiles(2)`
`genp(`*newvar*$_p$`)`	generate *newvar*$_p$ variable containing percentages
`altdef`	use alternative formula for calculating percentiles

xtile_options	Description
Main	
`nquantiles(#)`	number of quantiles; default is `nquantiles(2)`
`cutpoints(`*varname*`)`	use values of *varname* as cutpoints
`altdef`	use alternative formula for calculating percentiles

_pctile_options	Description
`nquantiles(#)`	number of quantiles; default is `nquantiles(2)`
`percentiles(`*numlist*`)`	calculate percentiles corresponding to the specified percentages
`altdef`	use alternative formula for calculating percentiles

`aweight`s, `fweight`s, and `pweight`s are allowed (see **[U] 11.1.6 weight**), except when the `altdef` option is specified, in which case no weights are allowed.

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

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 1,000 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**.

Options

Main

`nquantiles(`*#*`)` 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, …, 90th percentiles be computed. The default is `nquantiles(2)`; that is, the median is computed.

`genp(`*newvar*$_p$`)` (`pctile` only) specifies a new variable to be generated containing the percentages corresponding to the percentiles.

`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 1,000 (inclusive) percentiles can be requested. See [P] **numlist** for the syntax of a numlist.

Remarks and examples

stata.com

Remarks are presented under the following headings:

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 http://www.stata-press.com/data/r13/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

. pctile pct = mpg, nq(10) genp(percent)

. list percent pct in 1/10

     +---------------+
     | percent   pct |
     |---------------|
  1. |      10    14 |
  2. |      20    17 |
  3. |      30    18 |
  4. |      40    19 |
  5. |      50    20 |
     |---------------|
  6. |      60    22 |
  7. |      70    24 |
  8. |      80    25 |
  9. |      90    29 |
 10. |       .     . |
     +---------------+
```

`summarize, detail` calculates standard percentiles.

```
. summarize mpg, detail
                        Mileage (mpg)
-------------------------------------------------------------
      Percentiles      Smallest
 1%           12             12
 5%           14             12
10%           14             14       Obs                  74
25%           18             14       Sum of Wgt.          74

50%           20                      Mean            21.2973
                        Largest       Std. Dev.      5.785503
75%           25             34
90%           29             35       Variance       33.47205
95%           34             35       Skewness       .9487176
99%           41             41       Kurtosis       3.975005
```

`summarize, detail` can calculate only these particular percentiles. The `pctile` and `_pctile` 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

     +---------------+
     | percent   pct |
     |---------------|
  1. |      10    14 |
  2. |      20    16 |
  3. |      30    17 |
  4. |      40    18 |
  5. |      50    19 |
     |---------------|
  6. |      60    20 |
  7. |      70    22 |
  8. |      80    24 |
  9. |      90    28 |
 10. |       .     . |
     +---------------+
```

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 http://www.stata-press.com/data/r13/bp1, clear
. list bp, sep(4)

     +-----+
     |  bp |
     |-----|
  1. |  98 |
  2. | 100 |
  3. | 104 |
  4. | 110 |
     |-----|
  5. | 120 |
  6. | 120 |
  7. | 120 |
  8. | 120 |
     |-----|
  9. | 125 |
 10. | 130 |
 11. | 132 |
     +-----+
```

`xtile` can be used to create a variable, `quart`, that indicates the quartiles of `bp`.

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

     +-------------+
     |  bp   quart |
     |-------------|
  1. |  98       1 |
  2. | 100       1 |
  3. | 104       1 |
     |-------------|
  4. | 110       2 |
  5. | 120       2 |
  6. | 120       2 |
  7. | 120       2 |
  8. | 120       2 |
     |-------------|
  9. | 125       3 |
     |-------------|
 10. | 130       4 |
 11. | 132       4 |
     +-------------+
```

The categories created are

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

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:

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

. list bp quart percent pct, sepby(quart)

     +-------------------------------+
     |  bp   quart   percent     pct |
     |-------------------------------|
  1. |  98       1        25     104 |
  2. | 100       1        50     120 |
  3. | 104       1        75     125 |
     |-------------------------------|
  4. | 110       2         .       . |
  5. | 120       2         .       . |
  6. | 120       2         .       . |
  7. | 120       2         .       . |
  8. | 120       2         .       . |
     |-------------------------------|
  9. | 125       3         .       . |
     |-------------------------------|
 10. | 130       4         .       . |
 11. | 132       4         .       . |
     +-------------------------------+
```

`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:

$$(-\infty, 100], \quad (100, 110], \quad (110, 120], \quad (120, 130], \quad (130, +\infty)$$

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)

     +--------------------------+
     |  bp   class   category   |
     |--------------------------|
  1. |  98     100          1   |
  2. | 100     110          1   |
     |--------------------------|
  3. | 104     120          2   |
  4. | 110     130          2   |
     |--------------------------|
  5. | 120       .          3   |
  6. | 120       .          3   |
  7. | 120       .          3   |
  8. | 120       .          3   |
     |--------------------------|
  9. | 125       .          4   |
 10. | 130       .          4   |
     |--------------------------|
 11. | 132       .          5   |
     +--------------------------+
```

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 http://www.stata-press.com/data/r13/bp2, clear

. list in 1/11, sep(4)

     +------------+
     |  bp   case |
     |------------|
  1. |  98      1 |
  2. | 100      1 |
  3. | 104      1 |
  4. | 110      1 |
     |------------|
  5. | 120      1 |
  6. | 120      1 |
  7. | 120      1 |
  8. | 120      1 |
     |------------|
  9. | 125      1 |
 10. | 130      1 |
 11. | 132      1 |
     +------------+
```

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

     +-----+
     | pct |
     |-----|
  1. | 104 |
  2. | 117 |
  3. | 124 |
  4. |   . |
     +-----+
```

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)

     +-----------------------+
     |  bp   case   category |
     |-----------------------|
  1. |  98      1          1 |
  2. | 100      1          1 |
  3. | 104      1          1 |
     |-----------------------|
  4. | 110      1          2 |
     |-----------------------|
  5. | 120      1          3 |
  6. | 120      1          3 |
  7. | 120      1          3 |
  8. | 120      1          3 |
     |-----------------------|
  9. | 125      1          4 |
 10. | 130      1          4 |
 11. | 132      1          4 |
     +-----------------------+
```

❑ 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

```
. use http://www.stata-press.com/data/r13/bp1, clear
. xtile quint = bp, nq(5)
. pctile pct = bp, nq(5) genp(percent)
. list bp quint pct percent, sepby(quint)

     +-----------------------------+
     |  bp   quint   pct   percent |
     |-----------------------------|
  1. |  98       1   104        20 |
  2. | 100       1   120        40 |
  3. | 104       1   120        60 |
     |-----------------------------|
  4. | 110       2   125        80 |
  5. | 120       2     .         . |
  6. | 120       2     .         . |
  7. | 120       2     .         . |
  8. | 120       2     .         . |
     |-----------------------------|
  9. | 125       4     .         . |
     |-----------------------------|
 10. | 130       5     .         . |
 11. | 132       5     .         . |
     +-----------------------------+
```

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 http://www.stata-press.com/data/r13/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.

❑ 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.

❑

Stored results

`pctile` and `_pctile` store the following in `r()`:

Scalars

`r(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 pth 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 pth percentile is then

$$x_{[p]} = \begin{cases} \dfrac{x_{(i-1)} + x_{(i)}}{2} & \text{if } W_{(i-1)} = P \\ x_{(i)} & \text{otherwise} \end{cases}$$

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 pth 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_kth percentiles, where $p_k = 100\,k/m$ for $k = 1, 2, \ldots, m-1$.

If $x_{[p_{k-1}]} = x_{[p_k]}$, the kth 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] **3.4 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**