ctset — Declare data to be count-time data

Description

t refers to count-time data and is described here and in [ST] ct. Do not confuse count-time data with counting-process data, which can be analyzed using the st commands; see [ST] st.

When specified with a timevar and nfailvar, ctset declares the data in memory to be ct data. When you ctset your data, ctset also checks that what you have declared makes sense.

ctset, noshow will suppress display of the identities of the key ct variables before the output of other ct commands. By default, this information is shown. If you type ctset, noshow and then wish to restore the default behavior, type ctset, show.

ctset, clear is used mostly by programmers and causes Stata to no longer consider the data to be ct data. The dataset itself remains unchanged. It is not necessary to type ctset, clear before doing another ctset.

tset typed without arguments—which can be abbreviated ct—displays the identities of the key ct variables and reruns the checks on your data. Thus ct can remind you of what you have ctset (especially if you have ctset, noshow) and reverify your data if you make changes to the data.

Quick start

Declare count-time data with number of failures, fail, at each time in tvar
   ctset tvar fail

As above, and specify the number censored, cens, at each time
   ctset tvar fail cens

As above, and specify the number entering, enter, at each time
   ctset tvar fail cens enter

Specify that the number of failures and the number censored are recorded for groups identified by v1
   ctset tvar fail cens, by(v1)

Display previous ct settings, and verify that any changes to data correspond to settings
   ctset

Do not display information on variables specified in ctset when ct commands are run
   ctset, noshow

Menu

Statistics > Survival analysis > Setup and utilities > Declare data to be count-time data
Syntax

Declare data in memory to be count-time data and run checks on data

```
ctset timevar nfailvar [ncensvar [nentvar]] [, by(varlist) noshow]
```

Specify whether to display identities of key ct variables

```
ctset, {show|noshow}
```

Clear ct setting

```
ctset, clear
```

Display identity of key ct variables and rerun checks on data

```
{ctset|ct}
```

where `timevar` refers to the time of failure, censoring, or entry. It should contain times $\geq 0$.
`nfailvar` records the number failing at time `timevar`.
`ncensvar` records the number censored at time `timevar`.
`nentvar` records the number entering at time `timevar`.

Stata sequences events at the same time as

at `timevar` $\quad$ `nfailvar` failures occurred,
then at `timevar` + 0 $\quad$ `ncensvar` censorings occurred,
finally at `timevar` + 0 + 0 $\quad$ `nentvar` subjects entered the data.

Options

by(`varlist`) indicates that counts are provided by group. For instance, consider data containing records such as

```
t fail cens sex agecat
5 10 2 0 1
5 6 1 1 1
5 12 0 0 2
```

These data indicate that, in the category `sex` = 0 and `agecat` = 1, 10 failed and 2 were censored at time 5; for `sex` = 1, 1 was censored and 6 failed; and so on.

The above data would be declared

```
. ctset t fail cens, by(sex agecat)
```

The order of the records is not important, nor is it important that there be a record at every time for every group or that there be only one record for a time and group. However, the data must contain the full table of events.

`show` and `noshow` specify whether the identities of the key ct variables are to be displayed at the start of every ct command. Some users find the report reassuring; others find it repetitive. In any case, you can set and unset `show`, and you can always type `ct` to see the summary.

`clear` makes Stata no longer consider the data to be ct data.
Remarks and examples

Remarks are presented under the following headings:

Examples
Data errors flagged by ctset

Examples

About all you can do with ct data in Stata is convert it to survival-time (st) data so that you can use the survival analysis commands. To analyze count-time data with Stata,

. ctset ...
. cttost
.(now use any of the st commands)

Example 1: Simple ct data

We have data on generators that are run until they fail:

. use https://www.stata-press.com/data/r16/ctset1
. list, sep(0)

<table>
<thead>
<tr>
<th>failtime</th>
<th>fail</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>22</td>
</tr>
<tr>
<td>2.</td>
<td>30</td>
</tr>
<tr>
<td>3.</td>
<td>40</td>
</tr>
<tr>
<td>4.</td>
<td>52</td>
</tr>
<tr>
<td>5.</td>
<td>54</td>
</tr>
<tr>
<td>6.</td>
<td>55</td>
</tr>
<tr>
<td>7.</td>
<td>85</td>
</tr>
<tr>
<td>8.</td>
<td>97</td>
</tr>
<tr>
<td>9.</td>
<td>100</td>
</tr>
<tr>
<td>10.</td>
<td>122</td>
</tr>
<tr>
<td>11.</td>
<td>140</td>
</tr>
</tbody>
</table>

For instance, at time 54, four generators failed. To ctset these data, we could type

. ctset failtime fail

dataset name: https://www.stata-press.com/data/r16/ctset1.dta
time: failtime
no. fail: fail
no. lost: -- (meaning 0 lost)
no. enter: -- (meaning all enter at time 0)

It is not important that there be only 1 observation per failure time. For instance, according to our data, at time 85 there were seven failures. We could remove that observation and substitute two in its place—one stating that at time 85 there were five failures and another that at time 85 there were two more failures. ctset would interpret that data just as it did the previous data.
In more realistic examples, the generators might differ from one another. For instance, the following data show the number failing with old-style and new-style bearings:

```
. use https://www.stata-press.com/data/r16/ctset2
. list, sepby(bearings)
```

```
bearings failtime fail
1. old-style 22 1
2. old-style 40 2
3. old-style 54 1
4. old-style 84 2
5. old-style 97 2
6. old-style 100 1
7. new-style 30 1
8. new-style 52 1
9. new-style 55 1
10. new-style 100 3
11. new-style 122 2
12. new-style 140 1
```

That the data are sorted on bearings is not important. The `ctset` command for these data is

```
. ctset failtime fail, by(bearings)
```

Example 2: ct data with censoring

In real data, not all units fail in the time allotted. Say that the generator experiment was stopped after 150 days. The data might be

```
. use https://www.stata-press.com/data/r16/ctset3
. list
```

```
bearings failtime fail censored
1. old-style 22 1 0
2. old-style 40 2 0
3. old-style 54 1 0
4. old-style 84 2 0
5. new-style 97 2 0
6. old-style 100 1 0
7. old-style 150 0 2
8. new-style 30 1 0
9. new-style 52 1 0
10. new-style 55 1 0
11. new-style 122 2 0
12. new-style 140 1 0
13. new-style 150 0 3
```
The `ctset` command for these data is

```
. ctset failtime fail censored, by(bearings)
```

dataset name: https://www.stata-press.com/data/r16/ctset3.dta

time: failtime
no. fail: fail
no. lost: censored
no. enter: -- (meaning all enter at time 0)
by: bearings

In some other data, observations might also be censored along the way; that is, the value of censored would not be 0 before time 150. For instance, a record might read

<table>
<thead>
<tr>
<th>bearings</th>
<th>failtime</th>
<th>fail</th>
<th>censored</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>84</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>

This would mean that at time 84, two failed and one was lost because of censoring. The failure and censoring occurred at the same time, and when we analyze these data, Stata will assume that the censored observation could have failed, that is, that the censoring occurred after the two failures.

Example 3: ct data with delayed entry

Data on survival time of patients with a particular kind of cancer are collected. Time is measured as time since diagnosis. After data collection started, the sample was enriched with some patients from hospital records who had been previously diagnosed. Some of the data are

<table>
<thead>
<tr>
<th>time</th>
<th>die</th>
<th>cens</th>
<th>ent</th>
<th>other variables</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>50</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>5</td>
<td>...</td>
</tr>
<tr>
<td>30</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>...</td>
</tr>
<tr>
<td>31</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>...</td>
</tr>
<tr>
<td>32</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>...</td>
</tr>
<tr>
<td>100</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td>...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>...</td>
</tr>
</tbody>
</table>

Fifty patients entered at time 0 (time of diagnosis); five patients entered 1 day after diagnosis; and three, two, and one patients entered 30, 31, and 32 days after diagnosis, respectively. On the 32nd day, one of the previously entered patients died.

If the other variables are named `sex` and `agecat`, the `ctset` command for these data is

```
. ctset time die cens ent, by(sex agecat)
```

time: time
no. fail: die
no. lost: cens
no. enter: ent
by: sex agecat

The count-time format is an inferior way to record data like these—data in which every subject does not enter at time 0—because some information is already lost. When did the patient who died on the 32nd day enter? There is no way of telling.
For traditional survival analysis calculations, it does not matter. More modern methods of estimating standard errors, however, seek to identify each patient, and these data do not support using such methods.

This issue concerns the robust estimates of variance and the `vce(robust)` options on some of the `st` analysis commands. After converting the data, you must not use the `vce(robust)` option, even if an `st` command allows it, because the identities of the subjects—tying together when a subject starts and ceases to be at risk—are assigned randomly by `cttost` when you convert your `ct` to `st` data. When did the patient who died on the 32nd day enter? For conventional calculations, it does not matter, and `cttost` chooses a time randomly from the available entry times.

Data errors flagged by `ctset`

`ctset` requires only two things of your data: that the counts all be positive or zero and, if you specify an entry variable, that the entering and exiting subjects (failure + censored) balance.

If all subjects enter at time 0, we recommend that you do not specify a number-that-enter variable. `ctset` can determine for itself the number who enter at time 0 by summing the failures and censorings.

Also see

[ST] `ct` — Count-time data

[ST] `cttost` — Convert count-time data to survival-time data