Title

stata.com

sts — Generate, graph, list, and test the survivor and cumulative hazard functions

Description

`sts` reports and creates variables containing the estimated survivor and related functions, such as the Nelson–Aalen cumulative hazard function. For the survivor function, `sts` tests and produces Kaplan–Meier estimates or, via Cox regression, adjusted estimates.

`sts graph` is equivalent to typing `sts` by itself—it graphs the survivor function.

`sts list` lists the estimated survivor and related functions.

`sts test` tests the equality of the survivor function across groups.

`sts generate` creates new variables containing the estimated survivor function, the Nelson–Aalen cumulative hazard function, or related functions.

`sts` can be used with single- or multiple-record or single- or multiple-failure st data.

Syntax

`sts` [`graph`] [*if*] [*in*] [`,` ...]

`sts list` [*if*] [*in*] [`,` ...]

`sts test` *varlist* [*if*] [*in*] [`,` ...]

`sts generate` *newvar* `=` ... [*if*] [*in*] [`,` ...]

You must `stset` your data before using `sts`; see [ST] **stset**.
`fweights`, `iweights`, and `pweights` may be specified using `stset`; see [ST] **stset**.

See [ST] **sts graph**, [ST] **sts list**, [ST] **sts test**, and [ST] **sts generate** for details of syntax.

Remarks and examples

stata.com

Remarks are presented under the following headings:

`sts` concerns the survivor function, $S(t)$; the probability of surviving to t or beyond; the cumulative hazard function, $H(t)$; and the hazard function, $h(t)$. Its subcommands can list and generate variables containing $\widehat{S}(t)$ and $\widehat{H}(t)$ and test the equality of $S(t)$ over groups. Also:

- All subcommands share a common syntax.
- All subcommands deal with either the Kaplan–Meier product-limit or the Nelson–Aalen estimates unless you request adjusted survival estimates.
- If you request an adjustment, all subcommands perform the adjustment in the same way, which is described below.

The full details of each subcommand are found in the entries following this one, but each subcommand provides so many options to control exactly how the listing looks, how the graph appears, the form of the test to be performed, or what exactly is to be generated that the simplicity of `sts` can be easily overlooked.

So, without getting burdened by the details of syntax, let us demonstrate the `sts` commands by using the Stanford heart transplant data introduced in [ST] **stset**.

▷ Example 1

```
. use http://www.stata-press.com/data/r14/drugtr
```

Graph the Kaplan–Meier survivor function	`. sts graph` `. sts graph, by(drug)`
Graph the Nelson–Aalen cumulative hazard function	`. sts graph, cumhaz` `. sts graph, cumhaz by(drug)`
Graph the estimated hazard function	`. sts graph, hazard` `. sts graph, hazard by(drug)`
List the Kaplan–Meier survivor function	`. sts list` `. sts list, by(drug) compare`
List the Nelson–Aalen cumulative hazard function	`. sts list, cumhaz` `. sts list, cumhaz by(drug) compare`
Generate variable containing the Kaplan–Meier survivor function	`. sts gen surv = s` `. sts gen surv_by_drug = s, by(drug)`
Generate variable containing the Nelson–Aalen cumulative hazard function	`. sts gen haz = na` `. sts gen haz_by_drug = na, by(drug)`

Test equality of survivor functions	`. sts test drug`
	`. gen agecat = autocode(age,4,47,67)`
	`. sts test drug, strata(agecat)`

◁

Listing, graphing, and generating variables

You can list the overall survivor function by typing `sts list`, and you can graph it by typing `sts graph` or `sts`. `sts` assumes that you mean `graph` when you do not type a subcommand.

Or, you can list the Nelson–Aalen cumulative hazard function by typing `sts list, cumhaz`, and you can graph it by typing `sts graph, cumhaz`.

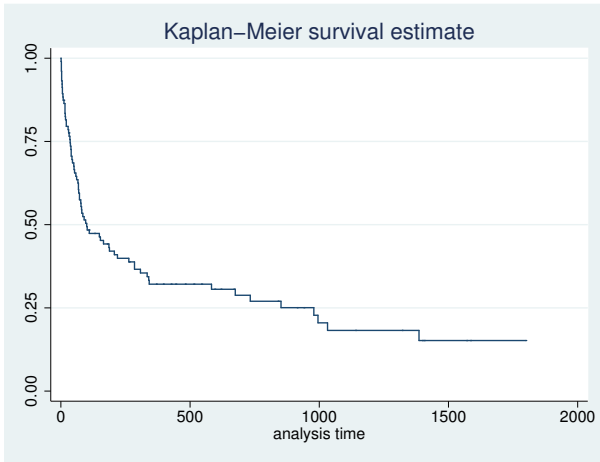
When you type `sts list`, you are shown all the details:

```
. use http://www.stata-press.com/data/r14/stan3
(Heart transplant data)
. stset, noshow
. sts list
```

Time	Beg. Total	Fail	Net Lost	Survivor Function	Std. Error	[95% Conf.	Int.]
1	103	1	0	0.9903	0.0097	0.9331	0.9986
2	102	3	0	0.9612	0.0190	0.8998	0.9852
3	99	3	0	0.9320	0.0248	0.8627	0.9670
5	96	1	0	0.9223	0.0264	0.8507	0.9604
(*output omitted*)							
1586	2	0	1	0.1519	0.0493	0.0713	0.2606
1799	1	0	1	0.1519	0.0493	0.0713	0.2606

When you type `sts graph` or just `sts`, you are shown a graph of the same result detailed by `list`:

```
. sts graph
```



`sts generate` is a rarely used command. Typing `sts generate survf = s` creates a new variable, `survf`, containing the same survivor function that `list` just listed and `graph` just graphed:

```
. sts gen survf = s
. sort t1
. list t1 survf in 1/10
```

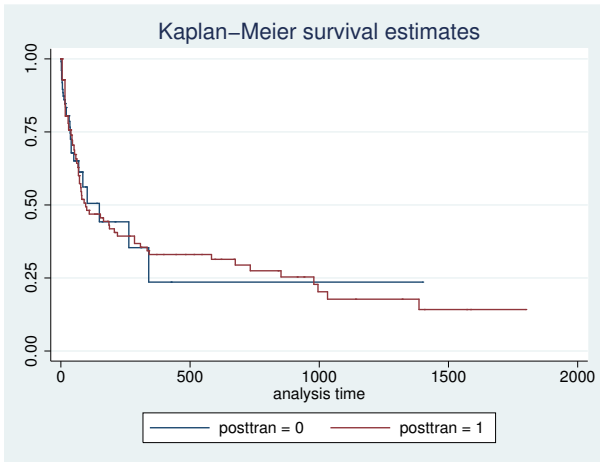
	t1	survf
1.	1	.99029126
2.	1	.99029126
3.	1	.99029126
4.	1	.99029126
5.	2	.96116505
6.	2	.96116505
7.	2	.96116505
8.	2	.96116505
9.	2	.96116505
10.	2	.96116505

`sts generate` is provided if you want to make a calculation, listing, or graph that `sts` cannot already do for you.

Comparing survivor or cumulative hazard functions

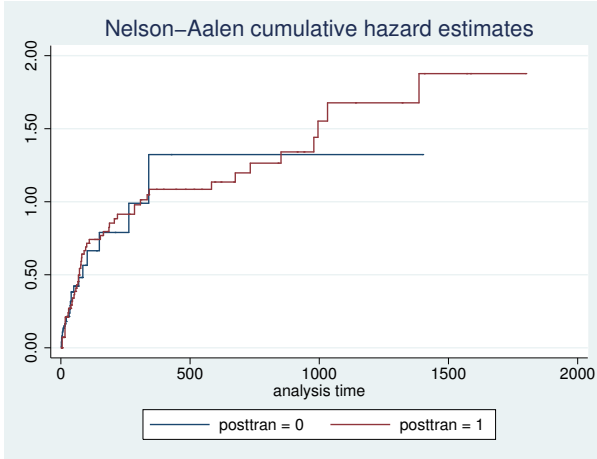
`sts` allows you to compare survivor or cumulative hazard functions. `sts graph` and `sts graph, cumhaz` are probably most successful at this. For example, survivor functions can be plotted using

```
. sts graph, by(posttran)
```



and Nelson–Aalen cumulative hazard functions can be plotted using

```
. sts graph, cumhaz by(posttran)
```



To compare survivor functions, we typed `sts graph`, just as before, and then we added `by(posttran)` to see the survivor functions for the groups designated by `posttran`. Here there are two groups, but as far as the `sts` command is concerned, there could have been more. `cumhaz` was also added to compare cumulative hazard functions.

You can also plot and compare estimated hazard functions by using `sts graph, hazard`. The hazard is estimated as a kernel smooth of the increments that sum to form the estimated cumulative hazard. The increments themselves do not estimate the hazard, but the smooth is weighted so that it estimates the hazard; see [ST] **sts graph**.

Just as you can compare survivor functions graphically by typing `sts graph, by(posttran)` and cumulative hazard functions by typing `sts graph, cumhaz by(posttran)`, you can obtain detailed listings by typing `sts list, by(posttran)` and `sts list, cumhaz by(posttran)`, respectively. Below we list the survivor function and specify `enter`, which adds a number-who-enter column:

```
. sts list, by(posttran) enter
```

Time	Beg. Total	Fail	Lost	Enter	Survivor Function	Std. Error	[95% Conf.	Int.]
posttran=0								
0	0	0	0	103	1.0000	.	.	.
1	103	1	3	0	0.9903	0.0097	0.9331	0.9986
2	99	3	3	0	0.9603	0.0195	0.8976	0.9849
(*output omitted*)								
427	2	0	1	0	0.2359	0.1217	0.0545	0.4882
1400	1	0	1	0	0.2359	0.1217	0.0545	0.4882
posttran=1								
1	0	0	0	3	1.0000	.	.	.
2	3	0	0	3	1.0000	.	.	.
3	6	0	0	3	1.0000	.	.	.
4	9	0	0	2	1.0000	.	.	.
5	11	0	0	3	1.0000	.	.	.
5.1	14	1	0	0	0.9286	0.0688	0.5908	0.9896
6	13	0	0	1	0.9286	0.0688	0.5908	0.9896
8	14	0	0	2	0.9286	0.0688	0.5908	0.9896
10	16	0	0	2	0.9286	0.0688	0.5908	0.9896
(*output omitted*)								
1586	2	0	1	0	0.1420	0.0546	0.0566	0.2653
1799	1	0	1	0	0.1420	0.0546	0.0566	0.2653

`sts list`'s `compare` option allows you to compare survivor or cumulative hazard functions by listing the groups side by side.

```
. sts list, by(posttran) compare
```

	Survivor Function	
posttran	0	1
time 1	0.9903	1.0000
225	0.4422	0.3934
449	0.2359	0.3304
673	0.2359	0.3139
897	0.2359	0.2535
1121	0.2359	0.1774
1345	0.2359	0.1774
1569	.	0.1420
1793	.	0.1420
2017	.	.

If we include the cumhaz option, the cumulative hazard functions are listed:

```
. sts list, cumhaz by(posttran) compare
           Nelson-Aalen Cum. Haz.
posttran             0          1
------------------------------------------
time       1     0.0097     0.0000
         225     0.7896     0.9145
         449     1.3229     1.0850
         673     1.3229     1.1350
         897     1.3229     1.3411
        1121     1.3229     1.6772
        1345     1.3229     1.6772
        1569          .     1.8772
        1793          .     1.8772
        2017          .          .
------------------------------------------
```

When you specify compare, the same detailed survivor or cumulative hazard function is calculated, but it is then evaluated at 10 or so given times, and those evaluations are listed. Above we left it to sts list to choose the comparison times, but we can specify them ourselves with the at() option:

```
. sts list, by(posttran) compare at(0 100 to 1700)
                Survivor Function
posttran             0          1
------------------------------------------
time       0     1.0000     1.0000
         100     0.5616     0.4814
         200     0.4422     0.4184
         300     0.3538     0.3680
         400     0.2359     0.3304
         500     0.2359     0.3304
         600     0.2359     0.3139
         700     0.2359     0.2942
         800     0.2359     0.2746
         900     0.2359     0.2535
        1000     0.2359     0.2028
        1100     0.2359     0.1774
        1200     0.2359     0.1774
        1300     0.2359     0.1774
        1400     0.2359     0.1420
        1500          .     0.1420
        1600          .     0.1420
        1700          .     0.1420
------------------------------------------
```

Testing equality of survivor functions

sts test tests equality of survivor functions:

```
. sts test posttran

Log-rank test for equality of survivor functions

           |   Events         Events
posttran   |  observed       expected
-----------+------------------------
0          |        30          31.20
1          |        45          43.80
-----------+------------------------
Total      |        75          75.00
                chi2(1) =        0.13
                Pr>chi2 =      0.7225
```

When you do not specify otherwise, `sts test` performs the log-rank test, but it can also perform the Wilcoxon test:

```
. sts test posttran, wilcoxon
```

Wilcoxon (Breslow) test for equality of survivor functions

posttran	Events observed	Events expected	Sum of ranks
0	30	31.20	-85
1	45	43.80	85
Total	75	75.00	0

chi2(1) = 0.14
Pr>chi2 = 0.7083

`sts test` also performs stratified tests; see [ST] **sts test**.

Adjusted estimates

All the estimates of the survivor function we have seen so far are the Kaplan–Meier product-limit estimates. `sts` can make adjusted estimates to the survivor function. We want to illustrate this and explain how it is done.

The heart transplant dataset is not the best for demonstrating this feature because we are starting with survivor functions that are similar already, so let's switch to data on a fictional drug trial:

```
. use http://www.stata-press.com/data/r14/drug2, clear
(Patient Survival in Drug Trial)
. stset, noshow
. stdescribe
```

Category	total	per subject: mean	min	median	max
no. of subjects	48				
no. of records	48	1	1	1	1
(first) entry time		0	0	0	0
(final) exit time		15.5	1	12.5	39
subjects with gap	0				
time on gap if gap	0				
time at risk	744	15.5	1	12.5	39
failures	31	.6458333	0	1	1

This dataset contains 48 subjects, all observed from time 0. The `st` command shows us how the dataset is currently declared:

```
. st
-> stset studytime, failure(died) noshow
     failure event:  died != 0 & died < .
obs. time interval:  (0, studytime]
 exit on or before:  failure
```

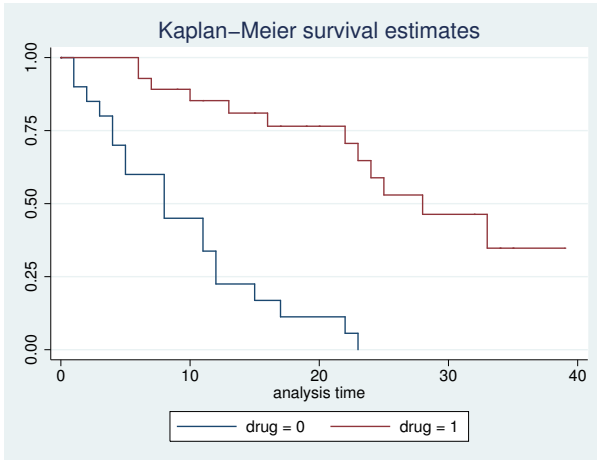
The dataset contains variables age and drug:

```
. summarize age drug
```

Variable	Obs	Mean	Std. Dev.	Min	Max
age	48	47.125	9.492718	32	67
drug	48	.5833333	.4982238	0	1

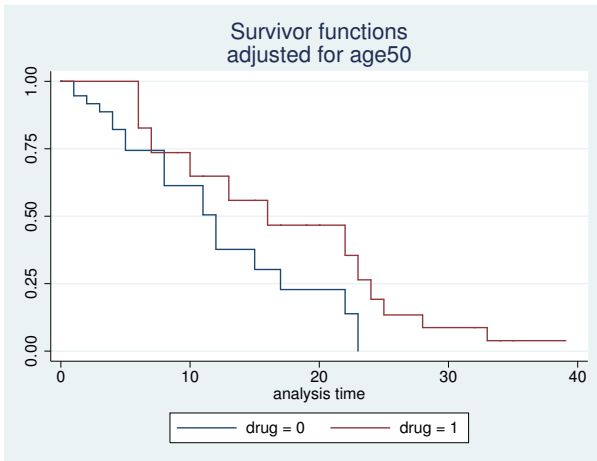
We are comparing the outcomes of drug = 1 with that of the placebo, drug = 0. Here are the survivor curves for the two groups:

```
. sts graph, by(drug)
```



Here are the survivor curves adjusted for age (and scaled to age 50):

```
. generate age50 = age-50
. sts graph, by(drug) adjustfor(age50)
```



The age difference between the two samples accounts for much of the difference between the survivor functions.

When you type `by(`*group*`) adjustfor(`*vars*`)`, `sts` fits a separate Cox proportional hazards model on *vars* (estimation via `stcox`) and retrieves the separately estimated baseline survivor functions. `sts graph` graphs the baseline survivor functions, `sts list` lists them, and `sts generate` saves them.

Thus `sts list` can list what `sts graph` plots:

```
. sts list, by(drug) adjustfor(age50) compare
        Adjusted Survivor Function
drug                 0          1
-------------------------------------
time       1     0.9463     1.0000
           5     0.7439     1.0000
           9     0.6135     0.7358
          13     0.3770     0.5588
          17     0.2282     0.4668
          21     0.2282     0.4668
          25          .     0.1342
          29          .     0.0872
          33          .     0.0388
          37          .     0.0388
          41          .          .
-------------------------------------
Survivor function adjusted for age50
```

In both the graph and the listing, we must adjust for variable `age50` = `age` − 50 and not just `age`. Adjusted survivor functions are adjusted to the `adjustfor()` variables and scaled to correspond to the `adjustfor()` variables set to 0. Here is the result of adjusting for `age`, which is 0 at birth:

```
. sts list, by(drug) adjustfor(age) compare
        Adjusted Survivor Function
drug                 0          1
-------------------------------------
time       1     0.9994     1.0000
           5     0.9970     1.0000
           9     0.9951     0.9995
          13     0.9903     0.9990
          17     0.9853     0.9987
          21     0.9853     0.9987
          25          .     0.9965
          29          .     0.9958
          33          .     0.9944
          37          .     0.9944
          41          .          .
-------------------------------------
Survivor function adjusted for age
```

These are equivalent to what we obtained previously but not nearly so informative because of the scaling of the survivor function. The `adjustfor(age)` option scales the survivor function to correspond to `age` = 0. `age` is calendar age, so the survivor function is scaled to correspond to a newborn.

There is another way that `sts` will adjust the survivor function. Rather than specifying `by(`*group*`) adjustfor(`*vars*`)`, we can specify `strata(`*group*`) adjustfor(`*vars*`)`:

```
. sts list, strata(drug) adjustfor(age50) compare
        Adjusted Survivor Function
drug                 0          1
-------------------------------------
time       1     0.9526     1.0000
           5     0.7668     1.0000
           9     0.6417     0.7626
          13     0.4080     0.5995
          17     0.2541     0.5139
          21     0.2541     0.5139
          25          .     0.1800
          29          .     0.1247
          33          .     0.0614
          37          .     0.0614
          41          .          .
-------------------------------------
Survivor function adjusted for age50
```

When we specify `strata()` instead of `by()`, instead of fitting separate Cox models for each stratum, `sts list` fits one stratified Cox model and retrieves the stratified baseline survivor function. That is, `strata()` rather than `by()` constrains the effect of the `adjustfor()` variables to be the same across strata.

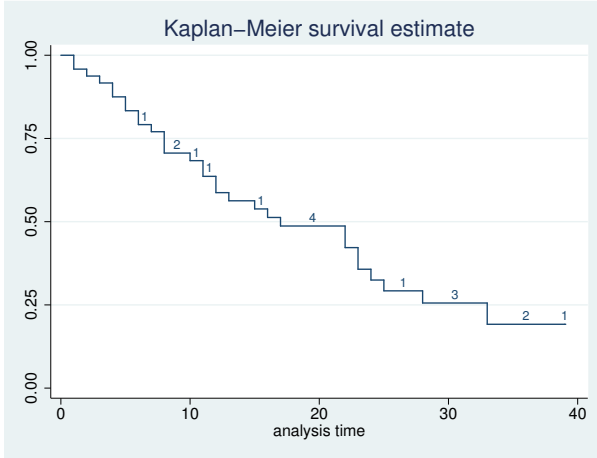
Counting the number lost due to censoring

`sts list`, in the detailed output, shows the number lost in the fourth column:

```
. sts list
          Beg.          Net          Survivor      Std.
  Time    Total   Fail   Lost         Function     Error     [95% Conf. Int.]
-------------------------------------------------------------------------------
     1       48      2      0           0.9583    0.0288     0.8435    0.9894
     2       46      1      0           0.9375    0.0349     0.8186    0.9794
     3       45      1      0           0.9167    0.0399     0.7930    0.9679
  (output omitted)
     8       36      3      1           0.7061    0.0661     0.5546    0.8143
     9       32      0      1           0.7061    0.0661     0.5546    0.8143
    10       31      1      1           0.6833    0.0678     0.5302    0.7957
  (output omitted)
    39        1      0      1           0.1918    0.0791     0.0676    0.3634
-------------------------------------------------------------------------------
```

`sts graph`, if you specify the `lost` option, will show that number, too:

```
. sts graph, lost
```



The number on the listing and on the graph is the number of net lost, defined as the number of censored minus the number who enter. With simple survival data—with 1 observation per subject—net lost corresponds to lost.

With more complicated survival data—meaning delayed entry or multiple records per subject—the number of net lost may surprise you. With complicated data, the vague term *lost* can mean many things. Sometimes subjects are lost, but mostly there are many censorings followed by reentries—a subject is censored at time 5 immediately to reenter the data with different covariates. This is called thrashing.

There are other possibilities: a subject can be lost, but only for a while, and so reenter the data with a gap; a subject can be censored out of one stratum to enter another. There are too many possibilities to dedicate a column in a table or a plotting symbol in a graph to each one. `sts`'s solution is to define *lost* as *net lost*, meaning censored minus entered, and show that number. How we define *lost* does not affect the calculation of the survivor function; it merely affects a number that researchers often report.

Defining lost as censored minus entered results in exactly what is desired for simple survival data. Because everybody enters at time 0, calculating censored minus entered amounts to calculating censored $-$ 0. The number of net lost is the number of censored.

In more complicated data, calculating censored minus entered results in the number really lost if there are no gaps and no delayed entry. Then the subtraction smooths the thrashing. In an interval, five might be censored and three reenter, so $5 - 3 = 2$ were lost.

In even more complicated data, calculating censored minus entered results in something reasonable once you understand how to interpret negative numbers and are cautious in interpreting positive ones. Five might be censored and three might enter (from the five? who can say?), resulting in two net lost; or three might be censored and five enter, resulting in -2 being lost.

`sts`, by default, reports the net lost but will, if you specify the `enter` option, report the pure number censored and the pure number who enter. Sometimes you will want to do that. Earlier in this entry, we used `sts list` to display the survivor functions in the Stanford heart transplant data for subjects pre- and posttransplantation, and we slipped in an `enter` option:

```
. use http://www.stata-press.com/data/r14/stan3, clear
(Heart transplant data)
```

```
. stset, noshow
. sts list, by(posttran) enter
```

Time	Beg. Total	Fail	Lost	Enter	Survivor Function	Std. Error	[95% Conf.	Int.]
posttran=0								
0	0	0	0	103	1.0000	.	.	.
1	103	1	3	0	0.9903	0.0097	0.9331	0.9986
2	99	3	3	0	0.9603	0.0195	0.8976	0.9849
3	93	3	3	0	0.9293	0.0258	0.8574	0.9657
(*output omitted*)								
427	2	0	1	0	0.2359	0.1217	0.0545	0.4882
1400	1	0	1	0	0.2359	0.1217	0.0545	0.4882
posttran=1								
1	0	0	0	3	1.0000	.	.	.
2	3	0	0	3	1.0000	.	.	.
3	6	0	0	3	1.0000	.	.	.
4	9	0	0	2	1.0000	.	.	.
5	11	0	0	3	1.0000	.	.	.
5.1	14	1	0	0	0.9286	0.0688	0.5908	0.9896
6	13	0	0	1	0.9286	0.0688	0.5908	0.9896
8	14	0	0	2	0.9286	0.0688	0.5908	0.9896
(*output omitted*)								
1586	2	0	1	0	0.1420	0.0546	0.0566	0.2653
1799	1	0	1	0	0.1420	0.0546	0.0566	0.2653

We did that to keep you from being shocked at negative numbers for the net lost. In this complicated dataset, the value of `posttran` changes over time. All patients start with `posttran` = 0, and later some change to `posttran` = 1.

Thus, at time 1 in the `posttran` = 0 group, three are lost—to the group but not to the experiment. Simultaneously, in the `posttran` = 1 group, we see that three enter. Had we not specified the `enter` option, you would not have seen that three enter, and you would have seen that −3 were, in net, lost:

```
. sts list, by(posttran)
```

Time	Beg. Total	Fail	Net Lost	Survivor Function	Std. Error	[95% Conf.	Int.]
posttran=0							
1	103	1	3	0.9903	0.0097	0.9331	0.9986
2	99	3	3	0.9603	0.0195	0.8976	0.9849
3	93	3	3	0.9293	0.0258	0.8574	0.9657
(*output omitted*)							
427	2	0	1	0.2359	0.1217	0.0545	0.4882
1400	1	0	1	0.2359	0.1217	0.0545	0.4882
posttran=1							
1	0	0	-3	1.0000	.	.	.
2	3	0	-3	1.0000	.	.	.
3	6	0	-3	1.0000	.	.	.
4	9	0	-2	1.0000	.	.	.
5	11	0	-3	1.0000	.	.	.
5.1	14	1	0	0.9286	0.0688	0.5908	0.9896
6	13	0	-1	0.9286	0.0688	0.5908	0.9896
8	14	0	-2	0.9286	0.0688	0.5908	0.9896
(*output omitted*)							
1586	2	0	1	0.1420	0.0546	0.0566	0.2653
1799	1	0	1	0.1420	0.0546	0.0566	0.2653

Here specifying `enter` makes the table easier to explain, but do not jump to the conclusion that specifying `enter` is always a good idea. In this same dataset, let's look at the overall survivor function, first with the `enter` option:

```
. sts list, enter
```

Time	Beg. Total	Fail	Lost	Enter	Survivor Function	Std. Error	[95% Conf.	Int.]
0	0	0	0	103	1.0000	.	.	.
1	103	1	3	3	0.9903	0.0097	0.9331	0.9986
2	102	3	3	3	0.9612	0.0190	0.8998	0.9852
3	99	3	3	3	0.9320	0.0248	0.8627	0.9670
(*output omitted*)								
1571	3	0	1	0	0.1519	0.0493	0.0713	0.2606
1586	2	0	1	0	0.1519	0.0493	0.0713	0.2606
1799	1	0	1	0	0.1519	0.0493	0.0713	0.2606

At time 1, three are lost and three enter. There is no delayed entry in this dataset, and there are no gaps; so, it is the same three that were lost and reentered, and no one was really lost. At time 1571, on the other hand, a patient really was lost. This is all more clearly revealed when we do not specify the `enter` option:

```
. sts list
```

Time	Beg. Total	Fail	Net Lost	Survivor Function	Std. Error	[95% Conf.	Int.]
1	103	1	0	0.9903	0.0097	0.9331	0.9986
2	102	3	0	0.9612	0.0190	0.8998	0.9852
3	99	3	0	0.9320	0.0248	0.8627	0.9670
(*output omitted*)							
1571	3	0	1	0.1519	0.0493	0.0713	0.2606
1586	2	0	1	0.1519	0.0493	0.0713	0.2606
1799	1	0	1	0.1519	0.0493	0.0713	0.2606

Thus, to summarize:

- The `sts list` and `graph` commands will show the number lost or censored. `sts list`, by default, shows this number on the detailed output. `sts graph` shows the number when you specify the `lost` option.
- By default, the number lost is the net number lost, defined as censored minus entered.
- Both commands allow you to specify the `enter` option and then show the number who actually entered, and the number lost becomes the actual number censored, not censored minus entered.

❑ Technical note

There is one other issue about the Kaplan–Meier estimator regarding delayed entry. When the earliest entry into the study occurs after $t = 0$, one may still calculate the Kaplan–Meier estimation, but the interpretation changes. Rather than estimating $S(t)$, you are now estimating $S(t|t_{\min})$, the probability of surviving past time t given survival to time $t_{\min}$, where $t_{\min}$ is the earliest entry time.

❑

Video examples

How to graph survival curves

How to calculate the Kaplan-Meier survivor and Nelson-Aalen cumulative hazard functions

How to test the equality of survivor functions using nonparametric tests

Stored results

See *Stored results* in [ST] **sts test**.

Methods and formulas

Unless adjusted estimates are requested, `sts` estimates the survivor function by using the Kaplan–Meier product-limit method.

When the `cumhaz` option is specified, `sts` estimates the cumulative hazard function by using the Nelson–Aalen estimator.

For an introduction to the Kaplan–Meier product-limit method and the log-rank test, see Pagano and Gauvreau (2000, 495–499) and Oliveira (2013); for a detailed discussion, see Cox and Oakes (1984), Kalbfleisch and Prentice (2002), or Klein and Moeschberger (2003). For an introduction to survival analysis with examples using the `sts` commands, see Dupont (2009).

Let t_j, $j = 1, \ldots,$ denote the times at which failure occurs. Let n_j be the number at risk of failure just before time t_j and d_j be the number of failures at time t_j. Then the nonparametric maximum likelihood estimate of the survivor function (Kaplan and Meier 1958) is

$$\widehat{S}(t) = \prod_{j|t_j \le t} \left(\frac{n_j - d_j}{n_j} \right)$$

(Kalbfleisch and Prentice 2002, 15).

The failure function, $\widehat{F}(t)$, is defined as $1 - \widehat{S}(t)$.

The standard error reported is given by Greenwood's formula (Greenwood 1926):

$$\widehat{\mathrm{Var}}\{\widehat{S}(t)\} = \widehat{S}^2(t) \sum_{j|t_j \le t} \frac{d_j}{n_j(n_j - d_j)}$$

(Kalbfleisch and Prentice 2002, 17–18). These standard errors, however, are not used for confidence intervals. Instead, the asymptotic variance of $\ln[-\ln \widehat{S}(t)]$,

$$\widehat{\sigma}^2(t) = \frac{\sum \frac{d_j}{n_j(n_j - d_j)}}{\left\{ \sum \ln\left(\frac{n_j - d_j}{n_j}\right) \right\}^2}$$

is used, where sums are calculated over $j|t_j \le t$ (Kalbfleisch and Prentice 2002, 18). The confidence bounds are then $\widehat{S}(t)^{\exp(\pm z_{\alpha/2}\widehat{\sigma}(t))}$, where $z_{\alpha/2}$ is the $(1 - \alpha/2)$ quantile of the normal distribution. `sts` suppresses reporting the standard error and confidence bounds if the data are `pweighted` because these formulas are no longer appropriate.

When the `adjustfor()` option is specified, the survivor function estimate, $\widehat{S}(t)$, is the baseline survivor function estimate $\widehat{S}_0(t)$ of `stcox`; see [ST] **stcox**. If, `by()`, is specified, $\widehat{S}(t)$ is obtained from fitting separate Cox models on `adjustfor()` for each of the `by()` groups. If instead `strata()` is specified, one Cox model on `adjustfor()`, stratified by `strata()`, is fit.

The Nelson–Aalen estimator of the cumulative hazard rate function is derived from Nelson (1972) and Aalen (1978) and is defined up to the largest observed time as

$$\widehat{H}(t) = \sum_{j|t_j \le t} \frac{d_j}{n_j}$$

Its variance (Aalen 1978) may be estimated by

$$\widehat{\mathrm{Var}}\{\widehat{H}(t)\} = \sum_{j|t_j \le t} \frac{d_j}{{n_j}^2}$$

Pointwise confidence intervals are calculated using the asymptotic variance of $\ln\widehat{H}(t)$,

$$\widehat{\phi}^2(t) = \frac{\widehat{\mathrm{Var}}\{\widehat{H}(t)\}}{\{\widehat{H}(t)\}^2}$$

The confidence bounds are then $\widehat{H}(t)\exp\{\pm z_{\alpha/2}\widehat{\phi}(t)\}$. If the data are `pweighted`, these formulas are not appropriate, and then confidence intervals are not reported.

References

Aalen, O. O. 1978. Nonparametric inference for a family of counting processes. *Annals of Statistics* 6: 701–726.

Breslow, N. E. 1992. Kaplan and Meier (1958) "Nonparametric estimation from incomplete observations". In *Breakthroughs in Statistics, Vol. II: Methodology and Distribution*, ed. S. Kotz and N. L. Johnson, 311–338. New York: Springer.

Clerc-Urmès, I., M. Grzebyk, and G. Hédelin. 2014. Net survival estimation with stns. *Stata Journal* 14: 87–102.

Cleves, M. A. 1999. stata53: censored option added to sts graph command. *Stata Technical Bulletin* 50: 34–36. Reprinted in *Stata Technical Bulletin Reprints*, vol. 9, pp. 4–7. College Station, TX: Stata Press.

Cox, D. R., and D. Oakes. 1984. *Analysis of Survival Data*. London: Chapman & Hall/CRC.

Dupont, W. D. 2009. *Statistical Modeling for Biomedical Researchers: A Simple Introduction to the Analysis of Complex Data*. 2nd ed. Cambridge: Cambridge University Press.

Greenwood, M. 1926. The natural duration of cancer. *Reports on Public Health and Medical Subjects* 33: 1–26.

Hogben, L. T. 1950. Major Greenwood, 1880–1949. *Obituary Notices of Fellows of the Royal Society* 7: 139–154.

Kalbfleisch, J. D., and R. L. Prentice. 2002. *The Statistical Analysis of Failure Time Data*. 2nd ed. New York: Wiley.

Kaplan, E. L., and P. Meier. 1958. Nonparametric estimation from incomplete observations. *Journal of the American Statistical Association* 53: 457–481.

Klein, J. P., and M. L. Moeschberger. 2003. *Survival Analysis: Techniques for Censored and Truncated Data*. 2nd ed. New York: Springer.

Linhart, J. M., J. S. Pitblado, and J. F. Hassell. 2004. From the help desk: Kaplan–Meier plots with stsatrisk. *Stata Journal* 4: 56–65.

Lo, S.-H., and D. Madigan. 2012. Obituary: Paul Meier 1924–2011. *IMS Bulletin* 41(1): 6.

Marks, H. M. 2004. A conversation with Paul Meier. *Clinical Trials* 1: 131–138.

Nelson, W. 1972. Theory and applications of hazard plotting for censored failure data. *Technometrics* 14: 945–966.

Newman, S. C. 2001. *Biostatistical Methods in Epidemiology*. New York: Wiley.

Oliveira, A. G. 2013. *Biostatistics Decoded*. Chichester, UK: Wiley.

Pagano, M., and K. Gauvreau. 2000. *Principles of Biostatistics*. 2nd ed. Belmont, CA: Duxbury.

Smythe, B. 2006. Obituary: Edward Kaplan 1920–2006. *IMS Bulletin* 35(10): 7.

Wilkinson, L. 1998. Greenwood, Major. In Vol. 2 of *Encyclopedia of Biostatistics*, ed. P. Armitage and T. Colton, 1778–1780. Chichester, UK: Wiley.

Major Greenwood (1880–1949) was born in London to a medical family. His given name, "Major", was also that of his father and grandfather. Greenwood trained as a doctor but followed a career in medical research, learning statistics from Karl Pearson. He worked as a medical statistician and epidemiologist at the Lister Institute, the Ministry of Health, and the London School of Hygiene and Tropical Medicine. With interests ranging from clinical trials to historical subjects, Greenwood played a major role in developing biostatistics in the first half of the twentieth century.

Edward Lynn Kaplan (1920–2006) was working at Bell Telephone Laboratories on the lifetimes of vacuum tubes when, through John W. Tukey, he became aware of the work of Paul Meier on essentially the same statistical problem. They had both previously been graduate students at Princeton. Their two separate papers were merged and the result was published after some years. Kaplan became a professor of mathematics at Oregon State University, where he wrote a book on mathematical programming and games.

Paul Meier (1924–2011) was born in Newark, New Jersey; took degrees at Oberlin and Princeton; and then served on the faculty at Lehigh, Johns Hopkins, Chicago, and Columbia. In addition to his key contribution with Kaplan, the most cited paper in statistical science, he worked as a biostatistician, making many theoretical and applied contributions in the area of clinical trials, especially through his early and sustained advocacy of randomization.

Wayne B. Nelson (1936–) was born in Chicago and received degrees in physics and statistics from Caltech and the University of Illinois. A longtime employee of General Electric, he now works as a consultant, specializing in reliability analysis and accelerated testing.

Odd Olai Aalen (1947–) was born in Oslo, Norway, and studied there and at Berkeley, where he was awarded a PhD in 1975 for a thesis on counting processes. He is a professor of statistics at the University of Oslo and works on survival and event history analysis. Aalen was one of the prime movers in introducing martingale ideas to this branch of statistics.

Nelson and Aalen met for the first time at a conference at the University of South Carolina in 2003.

Also see

[ST] **stci** — Confidence intervals for means and percentiles of survival time

[ST] **stcox** — Cox proportional hazards model

[ST] **sts generate** — Create variables containing survivor and related functions

[ST] **sts graph** — Graph the survivor, hazard, or cumulative hazard function

[ST] **sts list** — List the survivor or cumulative hazard function

[ST] **sts test** — Test equality of survivor functions

[ST] **stset** — Declare data to be survival-time data

[ST] **survival analysis** — Introduction to survival analysis