

Note15.1: STSET: Declare data to be Survival-time data

INTRODUCTION: Time to Event analysis, follow subjects over time and observe at which point in time they experience the event of interest.

For example:

1) How long after diagnosis of breast cancer before death? Survival analysis is predominately used in biomedical sciences, observing time to death either of patients or of laboratory animals.

2) Does using of new material A significantly increase the time for machine B to break down? Engineering sciences: reliability analysis, or failure time analysis: time it takes for machines or electronic components to break down

3) How long it takes to get divorce after get marry.

Social sciences: analyzing time to events such as job changes, marriage, birth of children and so forth.

Two aspects of survival analysis data: 1) Non-normality, this aspect of data violates the normality assumption of most traditional statistical models such as Regression or ANOVA. 2) Right Censoring: incomplete information about the subject, for instance the subject does not have an event before a study or experiment ends.

Before any st commands can be use, you should make sure you already have survival-time data. Many often, you may have count-time data instead.

For example: `/**count data 1**/`

```
. use http://www.stata-press.com/data/r9/ctset1,clear
. list
```

	failtime	fail
1.	22	1
2.	30	1
3.	40	2
4.	52	1
5.	54	4
6.	55	2
7.	85	7
8.	97	1
9.	100	3
10.	122	2
11.	140	1

Above data are count-time data: generators that are run until they fail. Time variable is failtime, and count variable is "fail". For this kind of data, you should 1) declare data to be count-time data by using command "ct", and 2) convert it to survival-time data.

Step 1: ctset data: `. ctset time_var fail_var,`

```
. ctset failtime fail /**for this specific data, our time variable is "failtime",
and failure variable is "fail"*/
```

```
dataset name: http://www.stata-press.com/data/r9/ctset1.dta
time: failtime
no. fail: fail
no. lost: -- (meaning 0 lost)
no. enter: -- (meaning all enter at time 0)
```

Step 2: convert it to survival-time data:

```
. cttost
```

```

failure event: fail != 0 & fail < .
obs. time interval: (0, failtime]
exit on or before: failure
weight: [fweight=w]

```

```

-----
11 total obs.
0 exclusions
-----

```

```

11 physical obs. remaining, equal to
25 weighted obs., representing
25 failures in single record/single failure data
1886 total analysis time at risk, at risk from t = 0
earliest observed entry t = 0
last observed exit t = 140

```

```
. list
```

	failtime	fail	w	_st	_d	_t	_t0
1.	22	1	1	1	1	22	0
2.	30	1	1	1	1	30	0
3.	40	1	2	1	1	40	0
4.	52	1	1	1	1	52	0
5.	54	1	4	1	1	54	0
6.	55	1	2	1	1	55	0
7.	85	1	7	1	1	85	0
8.	97	1	1	1	1	97	0
9.	100	1	3	1	1	100	0
10.	122	1	2	1	1	122	0
11.	140	1	1	1	1	140	0

After count-data is converted to survival-time data, five variables will be automatically created:

```

w is frequency weight;
_st=1 if the obs is to be included in the analysis
_d=failed;
_t: 1st study time
_t0: started time

```

Another example of count-time data is:

```

. /**count-data 2**/
. use http://www.stata-press.com/data/r9/ctset2,clear
. list

```

	bearings	failtime	fail
1.	0	22	1
2.	0	40	2
3.	0	54	1
4.	0	84	2
5.	0	97	2
6.	0	100	1
7.	1	30	1
8.	1	52	1
9.	1	55	1
10.	1	100	3
11.	1	122	2
12.	1	140	1

+-----+
 Above data show the number failing with old-style (bearings=0) and new-style (bearings=1) bearings. To **ctset** and **cttost** this kind of data we may use commands:

. ctset failtime fail,by(bearings)

```

dataset name:  http://www.stata-press.com/data/r9/ctset2.dta
              time:  failtime
              no. fail:  fail
              no. lost:  --                (meaning 0 lost)
              no. enter:  --              (meaning all enter at time 0)
              by:  bearings
  
```

. cttost

```

failure event:  fail != 0 & fail < .
obs. time interval:  (0, failtime]
exit on or before:  failure
weight:  [fweight=w]
  
```

```

-----
12 total obs.
0 exclusions
-----
12 physical obs. remaining, equal to
18 weighted obs., representing
18 failures in single record/single failure data
1439 total analysis time at risk, at risk from t =      0
      earliest observed entry t =      0
      last observed exit t =      140
  
```

. list

	bearings	failtime	fail	w	_st	_d	_t	_t0
1.	0	22	1	1	1	1	22	0
2.	0	40	1	2	1	1	40	0
3.	0	54	1	1	1	1	54	0
4.	0	84	1	2	1	1	84	0
5.	0	97	1	2	1	1	97	0

6.	0	100	1	1	1	1	100	0
7.	1	30	1	1	1	1	30	0
8.	1	52	1	1	1	1	52	0
9.	1	55	1	1	1	1	55	0
10.	1	100	1	3	1	1	100	0

11.	1	122	1	2	1	1	122	0
12.	1	140	1	1	1	1	140	0

Finally, we may also have count-time data like this:

```

. /**count-data 3**/
. use http://www.stata-press.com/data/r9/ctset3,clear
. list
  
```

	bearings	failtime	fail	censored
1.	0	22	1	0
2.	0	40	2	0
3.	0	54	1	0
4.	0	84	2	0
5.	1	97	2	0

6.	0	100	1	0
7.	0	150	0	2

```

8. |         1         30         1         0 |
9. |         1         52         1         0 |
10. |         1         55         1         0 |
-----+-----
11. |         1        122         2         0 |
12. |         1        140         1         0 |
13. |         1        150         0         3 |
-----+-----

```

Above data contain a variable "censored" which providing the number of generators with incomplete information after the experiment was stopped. For example, for obs=7 and obs=13, we find that after 150 hours after the experiment started, there were two generators with old-style (bearings=0) bearings, and 3 generators with new-style bearings, respectively, didn't fail when the experiment stopped because of power outage. For this type of data, we should use commands showed below to ctset:

```

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

```

Question: How to tell whether it is a count-time or survival-time data?

If event variable is count (e.g. 1,2,3) rather than dummy variable, then it is count-time data.

What is survival-time data and how to declare data to be survival-time data?

Standard commands: `stset time_var`

Example 1: single-record data

```

. use http://www.stata-press.com/data/r9/kva,clear
(Generator experiment)

```

```

. list in 1/3

```

```

-----+-----
| failtime  load  bearings |
-----+-----
1. |      100    15      0    |
2. |      140    15      1    |
3. |       97    20      0    |
-----+-----

```

Note: all generators are run until they fail.

```

. stset failtime

```

```

failure event: (assumed to fail at time=failtime)
obs. time interval: (0, failtime]
exit on or before: failure

```

```

-----
12 total obs.
0 exclusions
-----

```

```

12 obs. remaining, representing
12 failures in single record/single failure data
896 total analysis time at risk, at risk from t =      0
earliest observed entry t =      0
last observed exit t =      140

```

```

. list in 1/3

```

```

-----+-----
| failtime  load  bearings  _st  _d  _t  _t0 |
-----+-----
1. |      100    15      0      1  1  100  0 |
2. |      140    15      1      1  1  140  0 |
3. |       97    20      0      1  1   97  0 |
-----+-----

```

+-----+

Example 2: single-record data with censoring

```
. use http://www.stata-press.com/data/r9/kva2,clear
(Generator experiment)
. list in 1/4
```

	failtime	load	bearings	failed
1.	100	15	0	1
2.	140	15	1	0
3.	97	20	0	1
4.	122	20	1	1

Note: all generators are supposed to be run until they fail. However, there was a power outage in 140 hours of the experiment, thus the second generator, for example, did not fail at time 140. The whole experiment was discontinued at that point.

```
. stset failtime,failure(failed)
```

```
failure event: failed != 0 & failed < .
obs. time interval: (0, failtime]
exit on or before: failure
```

```
-----
12 total obs.
0 exclusions
-----
```

```
-----
12 obs. remaining, representing
11 failures in single record/single failure data
896 total analysis time at risk, at risk from t = 0
earliest observed entry t = 0
last observed exit t = 140
-----
```

```
. list in 1/4
```

	failtime	load	bearings	failed	_st	_d	_t	_t0
1.	100	15	0	1	1	1	100	0
2.	140	15	1	0	1	0	140	0
3.	97	20	0	1	1	1	97	0
4.	122	20	1	1	1	1	122	0

Example 3: Multiple-record data

```
. list
```

	patid	t	died
1.	90	100	0
2.	90	150	1
3.	91	50	1
4.	92	100	0
5.	92	150	0
6.	92	190	0
7.	93	100	0

Note: From above data, we find there are multiple records for a same patient id, because patient was observed every 50 days, and if he/she is able to survive to next 50 days (that is, died=0).

```
. stset t, id(patid) failure(died)

           id: patid
failure event: died != 0 & died < .
obs. time interval: (t[_n-1], t]
exit on or before: failure
```

```
-----
      7 total obs.
      0 exclusions
-----
      7 obs. remaining, representing
      4 subjects
      2 failures in single failure-per-subject data
490 total analysis time at risk, at risk from t =      0
           earliest observed entry t =      0
           last observed exit t =      190
```

```
. list
```

	patid	t	died	_st	_d	_t	_t0
1.	90	100	0	1	0	100	0
2.	90	150	1	1	1	150	100
3.	91	50	1	1	1	50	0
4.	92	100	0	1	0	100	0
5.	92	150	0	1	0	150	100
6.	92	190	0	1	0	190	150
7.	93	100	0	1	0	100	0

Example 4: Multiple-record data with multiple events (and only one value of the event variable is failure).

```
. list
```

	patid	t	event_~e
1.	90	100	401
2.	90	150	177
3.	91	50	402
4.	92	100	204
5.	92	150	401
6.	92	190	402
7.	93	100	122

Note: where code 401 means, for example, being discharged alive, 177 means transferring to other hospital, and code 402 means death, which is our event of interest.

```
. stset t, id(patid) failure(event_code=402)
```

```
           id: patid
failure event: event_code == 402
obs. time interval: (t[_n-1], t]
exit on or before: failure
```

```
-----
      7 total obs.
      0 exclusions
```

```

-----
      7  obs. remaining, representing
      4  subjects
      2  failures in single failure-per-subject data
     490 total analysis time at risk, at risk from t =      0
           earliest observed entry t =      0
           last observed exit t =      190

```

. list

```

+-----+
| patid   t   event_~e   _st   _d   _t   _t0 |
+-----+
1. |    90  100    401     1   0   100   0 |
2. |    90  150    177     1   0   150  100 |
3. |    91   50    402     1   1    50   0 |
4. |    92  100    204     1   0   100   0 |
5. |    92  150    401     1   0   150  100 |
+-----+
6. |    92  190    402     1   1   190  150 |
7. |    93  100    122     1   0   100   0 |
+-----+

```

Example 5: multiple-record data without "length_time" variable, but two point-time variables available.

. list

```

+-----+
| patid   event_~e   adday   curday |
+-----+
1. |    90    401    287    292 |
2. |    90    177     .    300 |
3. |    91    402    289    308 |
4. |    92    204    301    311 |
5. |    92    401     .    321 |
+-----+
6. |    92    402     .    333 |
7. |    93    122    260    310 |
+-----+

```

Note: above data do not have time-var, instead, we have patient's admitted date and current data. for this type of data, we should use commands:

```

stset 2nd_time_var, id(id_var) fail(event_var=value of failure) origin(time
1st_time_var)

```

For above specified example:

```

. stset curday,id(patid) fail(event_code=402) origin(time adday)

```

```

      id:  patid
  failure event:  event_code == 402
obs. time interval:  (curday[_n-1], curday]
  exit on or before:  failure
      t for analysis:  (time-origin)
      origin:  time adday

```

```

-----
      7  total obs.
      0  exclusions

```

```

-----
      7  obs. remaining, representing
      4  subjects
      2  failures in single failure-per-subject data
     114 total analysis time at risk, at risk from t =      0
           earliest observed entry t =      0
           last observed exit t =      50

```

. list

	patid	event_~e	adday	curday	_st	_d	_origin	_t	_t0
1.	90	401	287	292	1	0	287	5	0
2.	90	177	.	300	1	0	287	13	5
3.	91	402	289	308	1	1	289	19	0
4.	92	204	301	311	1	0	301	10	0
5.	92	401	.	321	1	0	301	20	10
6.	92	402	.	333	1	1	301	32	20
7.	93	122	260	310	1	0	260	50	0

Commands for data verification:

```
. use http://stata-press.com/data/cgg/hip,clear
(hip fracture study)

. sort id time0
. list in 1/20
```

	id	time0	time1	fracture	protect	age	calcium
1.	1	0	1	1	0	76	9.35
2.	2	0	1	1	0	80	7.8
3.	3	0	2	1	0	74	8.8
4.	4	0	3	1	0	67	10.1
5.	5	0	4	1	0	71	9.11
6.	6	0	4	1	0	82	8.43
7.	7	0	5	1	0	78	7.4
8.	8	0	5	1	0	73	8.99
9.	9	0	5	0	0	71	9.75
10.	9	5	8	1	.	.	9.18
11.	10	0	5	0	0	73	9.69
12.	10	5	8	0	.	.	9.47
13.	11	0	5	0	0	67	11.33
14.	11	5	8	1	.	.	10.72
15.	12	0	5	0	0	64	11.77
16.	12	5	8	1	.	.	11.34
17.	13	0	5	0	0	65	11.91
18.	13	5	11	1	.	.	11.31
19.	14	0	5	0	0	70	7.92
20.	14	5	11	1	.	.	10.8

```
. stset time1,id(id) fail(fracture=1) origin(time0)
```

```
      id: id
failure event: fracture == 1
obs. time interval: (time1[_n-1], time1]
exit on or before: failure
t for analysis: (time-origin)
origin: time time0
```

```
-----
106 total obs.
  0 exclusions
```

```
-----
106 obs. remaining, representing
 48 subjects
 31 failures in single failure-per-subject data
```

```

744 total analysis time at risk, at risk from t =      0
          earliest observed entry t =      0
          last observed exit t =      39

```

Command 1: **stdes**, describe survival-time data: what is event variable; what is time variable and id variable; how many subjects and records (note: subjects != records if one id has two or more records), how many failure (in other way: count if fracutre==1)**/

```
. stdes
```

```

failure _d: fracture == 1
analysis time _t: (time1-origin)
origin: time time0
id: id

```

Category	total	per subject			
		mean	min	median	max
no. of subjects	48				
no. of records	106	2.208333	1	2	3
(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

Command 2: **stvary**, reporting variables that vary over time, normally used with multiple-record datasets. It reports whether values of variables within subject vary over time and reports their pattern of missing values.

```
. stvary
```

```

failure _d: fracture == 1
analysis time _t: (time1-origin)
origin: time time0
id: id

```

variable	subjects for whom the variable is		never	always	sometimes
	constant	varying	missing	missing	missing
time0	8	40	48	0	0
protect	48	0	8	0	40
age	48	0	8	0	40
calcium	8	40	48	0	0

Command 3: **stfill**, you may have noticed that age and protect were only entered for the first record for each subject, the others are missing: you may use **stfill** command below to have Stata carry forward the first record values for these variables to later records:

```
. stfill protect age, forward
```

```

failure _d: fracture == 1
analysis time _t: (time1-origin)
origin: time time0
id: id

```

```

replace missing values with previously observed values:
protect: 58 real changes made

```

age: 58 real changes made

. list in 1/20

	id	time0	timel	fracture	protect	age	calcium	_st	_d	_t	_t0
1.	1	0	1	1	0	76	9.35	1	1	1	0
2.	2	0	1	1	0	80	7.8	1	1	1	0
3.	3	0	2	1	0	74	8.8	1	1	2	0
4.	4	0	3	1	0	67	10.1	1	1	3	0
5.	5	0	4	1	0	71	9.11	1	1	4	0
6.	6	0	4	1	0	82	8.43	1	1	4	0
7.	7	0	5	1	0	78	7.4	1	1	5	0
8.	8	0	5	1	0	73	8.99	1	1	5	0
9.	9	0	5	0	0	71	9.75	1	0	5	0
10.	9	5	8	1	0	71	9.18	1	1	8	5
11.	10	0	5	0	0	73	9.69	1	0	5	0
12.	10	5	8	0	0	73	9.47	1	0	8	5
13.	11	0	5	0	0	67	11.33	1	0	5	0
14.	11	5	8	1	0	67	10.72	1	1	8	5
15.	12	0	5	0	0	64	11.77	1	0	5	0
16.	12	5	8	1	0	64	11.34	1	1	8	5
17.	13	0	5	0	0	65	11.91	1	0	5	0
18.	13	5	11	1	0	65	11.31	1	1	11	5
19.	14	0	5	0	0	70	7.92	1	0	5	0
20.	14	5	11	1	0	70	10.8	1	1	11	5

Command 4: **st**, you may use command **st** without argument to ask Stata displays how the dataset is currently declared.

```
. st
-> stset timel, id(id) failure(fracture==1) origin(time time0)

      id: id
failure event: fracture == 1
obs. time interval: (timel[_n-1], timel]
exit on or before: failure
t for analysis: (time-origin)
origin: time time0
```