Title

> **stbase —** Form baseline dataset

Description

stbase without the at() option converts multiple-record st data to st data with every variable set to its value at baseline, defined as the earliest time at which each subject was observed. stbase without at() does nothing to single-record st data.

stbase, at() converts single- or multiple-record st data to a cross-sectional dataset (not st data), recording the number of failures at the specified time. All variables are given their values at baseline—the earliest time at which each subject was observed. In this form, single-failure data could be analyzed by logistic regression and multiple-failure data by Poisson regression, for instance.

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

Quick start

Set all variables to their values at the earliest time the subject was observed using stset data

    stbase

Create a dataset with one observation per subject, recording number of failures at time 10, with all variables set to the value at the earliest time the subject was observed

    stbase, at(10)

Menu

Statistics > Survival analysis > Setup and utilities > Form baseline dataset

Syntax

> stbase $\begin{bmatrix} if \end{bmatrix}$ $\begin{bmatrix} in \end{bmatrix}$ $\begin{bmatrix} , options \end{bmatrix}$

options	Description
Main	
at(#)	convert single/multiple-record st data to cross-sectional dataset at time #
gap(*newvar*)	name of variable containing gap time; default is gap or gaptime
replace	overwrite current data in memory
noshow	do not show st setting information
nopreserve	programmer's option; see *Options* below

You must stset your data before using stbase; see [ST] stset.

fweights, iweights, and pweights may be specified using stset; see [ST] stset.

nopreserve does not appear in the dialog box.

Options

> **Main**

at(#) changes what stbase does. Without the at() option, stbase produces another related st
dataset. With the at() option, stbase produces a related cross-sectional dataset.

gap(*newvar*) is allowed only with at(); it specifies the name of a new variable to be added to
the data containing the amount of time the subject was not at risk after entering and before #
as specified in at(). If gap() is not specified, the new variable will be named gap or gaptime,
depending on which name does not already exist in the data.

replace specifies that it is okay to change the data in memory, even though the dataset has not been
saved to disk in its current form.

noshow prevents stbase from showing the key st variables. This option is rarely used because most
people type stset, show or stset, noshow to set once and for all whether they want to see
these variables mentioned at the top of the output of every st command; see [ST] stset.

The following option is available with stbase but is not shown in the dialog box:

nopreserve is for use by programmers using stbase as a subroutine. It specifies that stbase not
preserve the original dataset so that it can be restored should an error be detected or should the
user press *Break*. Programmers would specify this option if, in their program, they had already
preserved the original data.

Remarks and examples

stata.com

Remarks are presented under the following headings:

> *stbase without the at() option*
> *stbase with the at() option*
> *Single-failure st data where all subjects enter at time 0*
> *Single-failure st data where some subjects enter after time 0*
> *Single-failure st data with gaps and perhaps delayed entry*
> *Multiple-failure st data*

stbase without the at() option

Once you type stbase, you may not streset your data, even though the data are st. streset will refuse to run because the data have changed, and if the original rules were reapplied, they might produce different, incorrect results. The st commands use four key variables:

_t0	the time at which the record came under observation
_t	the time at which the record left observation
_d	1 if the record left under failure, 0 otherwise
_st	whether the observation is to be used (contains 1 or 0)

These variables are adjusted by stbase. The _t0 and _t variables, in particular, are derived from your variables according to options you specified at the time you stset the data, which might include an origin() rule, an entry() rule, and the like. Once intervening observations are eliminated, those rules will not necessarily produce the same results that they did previously.

To illustrate how stbase works, consider multiple-record, time-varying st data, on which you have performed some analysis. You now wish to compare your results with a simpler, non-time-varying analysis. For instance, suppose that variables x1 and x2 measure blood pressure and weight, respectively, and that readings were taken at various times. Perhaps you fit the model

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

. stset
-> stset t, id(id) failure(d) exit(time .) noshow

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

      1734  total observations
         0  exclusions
_____

      1734  observations remaining, representing
       926  subjects
       808  failures in multiple-failure-per-subject data
    435855  total analysis time at risk and under observation
                                         at risk from t =         0
                              earliest observed entry t =         0
                                 last observed exit t =         960
```

```
. stcox x1 x2

Iteration 0:   log likelihood = -5034.9569
Iteration 1:   log likelihood = -4978.4198
Iteration 2:   log likelihood = -4978.1915
Iteration 3:   log likelihood = -4978.1914
Refining estimates:
Iteration 0:   log likelihood = -4978.1914

Cox regression -- Breslow method for ties

No. of subjects =            926                   Number of obs    =        1,734
No. of failures =            808
Time at risk    =         435855
                                                   LR chi2(2)       =       113.53
Log likelihood  =    -4978.1914                    Prob > chi2      =       0.0000
```

_t	Haz. Ratio	Std. Err.	z	P>|z|	[95% Conf. Interval]	
x1	2.273456	.216537	8.62	0.000	1.886311	2.740059
x2	.329011	.0685638	-5.33	0.000	.2186883	.4949888

with these data. You now wish to fit that same model but this time use the values of x1 and x2 at baseline. You do this by typing

```
. stbase, replace
notes:
    1.  no gaps
    2.  there were multiple failures or reentries after failures
    3.  baseline data has multiple records per id(id)
    4.  all records have covariate values at baseline
. stcox x1 x2

Iteration 0:   log likelihood = -7886.9779
Iteration 1:   log likelihood = -7863.9974
Iteration 2:   log likelihood = -7863.9295
Iteration 3:   log likelihood = -7863.9295
Refining estimates:
Iteration 0:   log likelihood = -7863.9295

Cox regression -- Breslow method for ties

No. of subjects =            926                   Number of obs    =        1,734
No. of failures =          1,337
Time at risk    =         435855
                                                   LR chi2(2)       =        46.10
Log likelihood  =    -7863.9295                    Prob > chi2      =       0.0000
```

_t	Haz. Ratio	Std. Err.	z	P>|z|	[95% Conf. Interval]	
x1	1.413195	.1107945	4.41	0.000	1.211903	1.647921
x2	.4566673	.0765272	-4.68	0.000	.3288196	.6342233

Another way you could perform the analysis is to type

```
. generate x1_0 = x1

. generate x2_0 = x2

. stfill x1_0 x2_0, baseline

. stcox x1 x2
```

See [ST] **stfill**. The method you use makes no difference, but if there were many explanatory variables, **stbase** would be easier.

stbase changes the data to record the same events but changes the values of all other variables to their values at the earliest time the subject was observed.

stbase also simplifies the st data where possible. Say that one of your subjects has three records in the original data and ends in a failure:
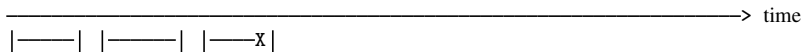
After running stbase, this subject would have one record in the data:
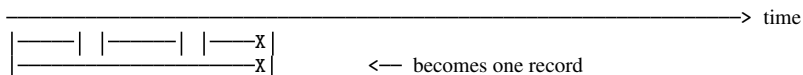
Here are some other examples of how stbase would process records with gaps and multiple failure events:
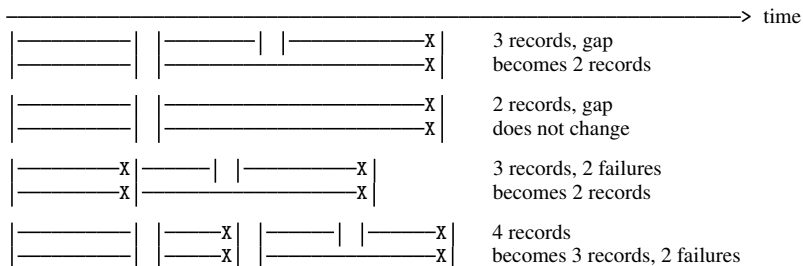
The following example shows numerically what is shown in the diagram above.

```
. use http://www.stata-press.com/data/r14/stbasexmpl, clear
. list, sepby(id)
```

	id	time0	time	wgt	death
1.	1	0	2	114	0
2.	1	3	5	110	0
3.	1	5	11	118	1
4.	2	0	2	120	0
5.	2	3	11	111	1
6.	3	0	2	108	1
7.	3	2	4	105	0
8.	3	4	7	113	1
9.	4	0	2	98	0
10.	4	3	4	101	1
11.	4	5	6	106	0
12.	4	6	11	104	1

```
. stset time, id(id) fail(death) time0(time0) exit(time .)

                id:  id
     failure event:  death != 0 & death < .
obs. time interval:  (time0, time]
 exit on or before:  time .
_____
            12  total observations
             0  exclusions
_____
            12  observations remaining, representing
             4  subjects
             6  failures in multiple-failure-per-subject data
            36  total analysis time at risk and under observation
                                            at risk from t =          0
                                 earliest observed entry t =          0
                                    last observed exit t =           11

. list, sepby(id)
```

	id	time0	time	wgt	death	_st	_d	_t	_t0
1.	1	0	2	114	0	1	0	2	0
2.	1	3	5	110	0	1	0	5	3
3.	1	5	11	118	1	1	1	11	5
4.	2	0	2	120	0	1	0	2	0
5.	2	3	11	111	1	1	1	11	3
6.	3	0	2	108	1	1	1	2	0
7.	3	2	4	105	0	1	0	4	2
8.	3	4	7	113	1	1	1	7	4
9.	4	0	2	98	0	1	0	2	0
10.	4	3	4	101	1	1	1	4	3
11.	4	5	6	106	0	1	0	6	5
12.	4	6	11	104	1	1	1	11	6

```
. stbase, replace
         failure _d:  death
   analysis time _t:  time
  exit on or before:  time .
                 id:  id
notes:
    1.  there were gaps
    2.  there were multiple failures or reentries after failures
    3.  baseline data has multiple records per id(id)
    4.  all records have covariate values at baseline
```

```
. list, sepby(id)
```

	id	time0	time	wgt	death	_st	_d	_t	_t0
1.	1	0	2	114	0	1	0	2	0
2.	1	3	11	114	1	1	1	11	3
3.	2	0	2	120	0	1	0	2	0
4.	2	3	11	120	1	1	1	11	3
5.	3	0	2	108	1	1	1	2	0
6.	3	2	7	108	1	1	1	7	2
7.	4	0	2	98	0	1	0	2	0
8.	4	3	4	98	1	1	1	4	3
9.	4	5	11	98	1	1	1	11	5

stbase with the at() option

stbase, at() produces a cross-sectional dataset recording the status of each subject at the specified time. This new dataset is not st. Four "new" variables are created:

- the first entry time for the subject,
- the time on gap,
- the time at risk, and
- the number of failures during the time at risk.

The names given to those variables depend on how your data are stset. Pretend that your stset command was

```
. stset var1, failure(var2) time0(var3) ...
```

Then

the first entry time	will be named	*var3* or time0 or _t0
the time on gap	will be named	gap() or gap or gaptime
the time at risk	will be named	*var1*
the number of (or whether) failures	will be named	*var2* or failure or _d

The names may vary because, for instance, if you did not specify a *var2* variable when you stset your data, stbase, at() looks around for a name.

You need not memorize this; the names are obvious from the output produced by stbase, at().

Consider the actions of stbase, at() with some particular st datasets. Pretend that the command given is

```
. use http://www.stata-press.com/data/r14/stbasexmpl2, clear
. list, sepby(id)
```

	id	time0	time	wgt	death
1.	1	0	2	114	0
2.	1	2	8	110	0
3.	1	8	11	118	1
4.	2	0	1	120	0
5.	2	1	3	111	0
6.	2	3	8	108	0
7.	2	8	10	98	1

```
. stset time, id(id) fail(death) time0(time0)
                id:  id
     failure event:  death != 0 & death < .
obs. time interval:  (time0, time]
 exit on or before:  failure
```
───
```
            7  total observations
            0  exclusions
```
───
```
            7  observations remaining, representing
            2  subjects
            2  failures in single-failure-per-subject data
           21  total analysis time at risk and under observation
                                            at risk from t =          0
                                 earliest observed entry t =          0
                                      last observed exit t =         11
. list, sepby(id)
```

	id	time0	time	wgt	death	_st	_d	_t	_t0
1.	1	0	2	114	0	1	0	2	0
2.	1	2	8	110	0	1	0	8	2
3.	1	8	11	118	1	1	1	11	8
4.	2	0	1	120	0	1	0	1	0
5.	2	1	3	111	0	1	0	3	1
6.	2	3	8	108	0	1	0	8	3
7.	2	8	10	98	1	1	1	10	8

```
. stbase, at(5) replace
          failure _d:  death
   analysis time _t:  time
                id:  id

         data now cross-section at time 5
```

Variable	description
id	subject identifier
time0	first entry time
gap	time on gap
time	time at risk
death	number of failures during interval time

Variable	Obs	Mean	Std. Dev.	Min	Max
time0	2	0	0	0	0
gap	2	0	0	0	0
time	2	5	0	5	5
death	2	0	0	0	0

```
. list
```

	id	wgt	death	time	time0	gap
1.	1	114	0	5	0	0
2.	2	120	0	5	0	0

thus producing a cross-section at analysis time 5.

Note that the value of time specified with the at() option must correspond to time in the analysis scale, that is, *t*. See [ST] **stset** for a definition of analysis time.

Single-failure st data where all subjects enter at time 0

The result of stbase, at(5) would be one record per subject. Any subject who was censored before time 5 would not appear in the data; the rest would. Those that failed after time 5 will be recorded as having been censored at time 5 (*failvar* = 0); those that failed at time 5 or earlier will have *failvar* = 1.

timevar will contain

for the failures:	
time of failure	if failed on or before time 5 or
5	if the subject has not failed yet
for the censored:	
5	if the subject has not failed yet

With such data, you could perform

- logistic regression of *failvar* on any of the characteristics or

- incidence-rate analysis, summing the failures (perhaps within strata) and the time at risk, *timevar*.

With these data, you could examine 5-year survival probabilities.

Single-failure st data where some subjects enter after time 0

The data produced by stbase, at(5) would be similar to the above, except

- persons who enter on or after time 5 would not be included in the data (because they have not entered yet) and
- the time at risk, *timevar*, would properly account for the time at which each patient entered.

timevar (the time at risk) will contain

<div style="margin-left: 6em;">

for the failures:
time of failure or less	if failed on or before time 5 (or less because the subject may not have entered at time 0); or
5 or less	if the subject has not failed yet (or less because the subject may not have entered at time 0)

for the censored:
5 or less	if the subject has not failed yet (or less because the subject may not have entered at time 0)

</div>

Depending on the analysis you are performing, you may have to discard those that enter late. This is easy to do because t0 contains the first time of entry.

With these data, you could perform the following:

- Logistic regression of *failvar* on any of the characteristics, but only if you restricted the sample to if t0 == 0 because those who entered after time 0 have a lesser risk of failing over the fixed interval.
- Incidence-rate analysis, summing the failures (perhaps within stratum) and the time at risk, *timevar*. Here you would have to do nothing differently from what you did in the previous example. The time-at-risk variable already includes the time of entry for each patient.

Single-failure st data with gaps and perhaps delayed entry

These data will be similar to the delayed-entry, no-gap data, but gap will contain 0 only for those observations that have no gap.

If analyzing these data, you could perform

- logistic regression, but the sample must be restricted to if t0 == 0 & gap == 0, or
- incidence-rate analysis, and nothing would need to be done differently; the time at risk, *timevar*, accounts for late entry and gaps.

Multiple-failure st data

The multiple-failure case parallels the single-failure case, except that fail will not solely contain 0 and 1; it will contain 0, 1, 2, . . . , depending on the number of failures observed. Regardless of late entry, gaps, etc., you could perform

- Poisson regression of fail, the number of events, but remember to specify exposure(*timevar*), and
- incidence-rate analysis.

Also see

[ST] **stfill** — Fill in by carrying forward values of covariates

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