9 The Break key

Contents

9.1 Making Stata stop what it is doing

When you want to make Stata stop what it is doing and return to the Stata dot prompt, you click on *Break*:

Stata for Windows:	click on the **Break** button (it is the button with the big red X), or press *Ctrl+Pause/Break*
Stata for Mac:	click on the **Break** button or press *Command+.* (period)
Stata for Unix(GUI):	click on the **Break** button or press *Ctrl+k*
Stata for Unix(console):	press *Ctrl+c* or press *q*

Elsewhere in this manual, we describe this action as simply clicking on *Break*. Break tells Stata to cancel what it is doing and return control to you as soon as possible.

If you click on *Break* in response to the input prompt or while you are typing a line, Stata ignores it, because you are already in control.

If you click on *Break* while Stata is doing something—creating a new variable, sorting a dataset, making a graph, etc.—Stata stops what it is doing, undoes it, and issues an input prompt. The state of the system is the same as if you had never issued the command.

▷ Example 1

You are fitting a logit model, type the command, and, as Stata is working on the problem, realize that you omitted an important variable:

```
. logit foreign mpg weight
Iteration 0:  Log likelihood =  -45.03321
Iteration 1:  Log likelihood = -29.898968
--Break--
r(1);

. _
```

When you clicked on *Break*, Stata responded by typing `--Break--` and then typing `r(1);`. Clicking on *Break* always results in a return code of 1—that is why return codes are called return codes and not error codes. The 1 does not indicate an error, but it does indicate that the command did not complete its task.

◁

9.2 Side effects of clicking on Break

In general, there are no side effects of clicking on Break. We said above that Stata undoes what it is doing so that the state of the system is the same as if you had never issued the command. There are two exceptions to that statement.

If you are reading data from disk by using `import delimited`, `infile`, or `infix`, whatever data have already been read will be left behind in memory, the theory being that perhaps you stopped the process so you could verify that you were reading the right data correctly before sitting through the whole process. If not, you can always `clear`.

```
. infile v1-v9 using workdata
(eof not at end of obs)
(4 observations read)
--Break--
r(1);
```

The other exception is `sort`. You have a large dataset in memory, decide to sort it, and then change your mind.

```
. sort price
--Break--
r(1);
```

If the dataset was previously sorted by, say, the variable `prodid`, it is no longer. When you click on *Break* in the middle of a `sort`, Stata marks the data as unsorted.

9.3 Programming considerations

There are basically no programming considerations for handling Break because Stata handles it all automatically. If you write a program or do-file, execute it, and then click on *Break*, Stata stops execution just as it would with an internal command.

Advanced programmers may be concerned about cleaning up after themselves; perhaps they have generated a temporary variable they intended to drop later or a temporary file they intended to erase later. If a Stata user clicks on *Break*, how can you ensure that these temporary variables and files will be erased?

If you obtain names for such temporary items from Stata's `tempname`, `tempvar`, and `tempfile` commands, Stata will automatically erase the temporary items; see **[U] 18.7 Temporary objects**.

There are instances, however, when a program must commit to executing a group of commands without interruption, or the user's data would be left in an intermediate or undefined state. In these instances, Stata provides a

```
nobreak {
        ...
}
```

construct; see [P] **break**. Also see [M-5] **setbreakintr( )** to read about Break-key processing in Mata.