preserve — Preserve and restore data

Description   Syntax   Options   Remarks and examples   Also see

Description

`preserve` preserves the data, guaranteeing that data will be restored after program termination.

`restore` forces a restore of the data now.

`set max_preservemem`, available only in Stata/MP, controls the maximum amount of memory `preserve` will use to store preserved datasets in memory. Once this limit is exceeded, `preserve` will store datasets on disk.

Syntax

Preserve data

```
preserve [ , changed ]
```

Restore data

```
restore [ , not preserve ]
```

Set maximum memory for fast storage by `preserve`

```
set max_preservemem amt [ , permanently ]
```

where *amt* is #[b | k | m | g], and the default unit is `b`.

Options

`changed` instructs `preserve` to preserve only the flag indicating that the data have changed since the last save. Use of this option is strongly discouraged, as explained in the technical note below.

`not` instructs `restore` to cancel the previous `preserve`.

`preserve` instructs `restore` to restore the data now, but not to cancel the restoration of the data again at program conclusion. If `preserve` is not specified, the scheduled restoration at program conclusion is canceled.

`permanently` instructs `set max_preservemem` that, in addition to making the change right now, the new limit be remembered and become the default setting when you invoke Stata.

`once` is not shown in the syntax diagram but is allowed with `set max_preservemem`. It is for use by system administrators and allows them to set `max_preservemem` such that users cannot modify it; see *Notes for system administrators* in [D] **memory**.

Remarks and examples

preserve and restore deal with the programming problem where the user's data must be changed to achieve the desired result but, when the program concludes, the programmer wishes to undo the damage done to the data. When preserve is issued, the user's data are preserved. The data in memory remain unchanged. When the program or do-file concludes, the user's data are automatically restored.

After a preserve, the programmer can also instruct Stata to restore the data now with the restore command. This is useful when the programmer needs the original data back and knows that no more damage will be done to the data. restore, preserve can be used when the programmer needs the data back but plans further damage. restore, not can be used when the programmer wishes to cancel the previous preserve and to have the data currently in memory returned to the user.

For speed, Stata/MP uses frames to preserve datasets to memory rather than writing them to disk. It does so unless the max_preservemem limit has been reached in terms of memory consumed by preserved datasets. Once the limit has been reached, Stata/MP falls back to writing preserved datasets to disk. Stata/SE and Stata/BE are typically used on computers with less memory and as such always preserve datasets on disk.

The default setting for set max_preservemem is 1g, meaning 1 gigabyte. If *amt* is set to 0b (0 bytes), preserve will always use disk storage. If *amt* is set to ., preserve will use as much memory as the operating system is willing to supply. The memory used by preserve is in addition to the memory used by other datasets you may have in memory and is not included in your max_memory setting (see [D] **memory**). Keep this in mind when changing this setting.

▷ Example 1

preserve is usually used by itself and is used early in the program. Say that a programmer is writing a program to report some statistic, but the statistic cannot be calculated without changing the user's data. Here changing does not mean merely adding a variable or two; that could be done with temporary variables as described in [P] **macro**. Changing means that the data really must be changed: observations might be discarded, the contents of existing variables changed, and the like. Although the programmer could just ignore the destruction of the user's data, the programmer might actually want to use the program herself and knows that she will become exceedingly irritated when she uses it without remembering to first save her data. The programmer wishes to write a programmatically correct, or PC, command. Doing so is not difficult:

```
program myprog
        (code for interpreting—parsing—the user's request)
        preserve
        (code that destroys the data)
        (code that makes the calculation)
        (code that reports the result)
end
```

To preserve the data, preserve must make a copy of it on disk. Therefore, our programmer smartly performs all the parsing and setup, where errors are likely, before the preserve. Once she gets to the point in the code where the damage must be done, however, she preserves the data. After that, she forgets the problem. Stata handles restoring the user's data, even if the user presses *Break* in the middle of the program.

◁

▷ Example 2

Now let's consider a program that must destroy the user's data but needs the data back again, and, once the data are recovered, will do no more damage. The outline is

```
program myprog
        (code for interpreting—parsing—the user's request)
        preserve
        (code that destroys the data)
        (code that makes the first part of the calculation)
        restore
        (code that makes the second part of the calculation)
        (code that reports the result)
end
```

Although there are other ways the programmer could have arranged to save the data and get the data back [`snapshot` (see [D] **snapshot**) or `save` and `use` with temporary files as described in [P] **macro** come to mind], this method is better because should the user press *Break* after the data are damaged but before the data are restored, Stata will handle restoring the data.

◁

▷ Example 3

This time the program must destroy the user's data, bring the data back and destroy the data again, and finally report its calculation. The outline is

```
program myprog
        (code for interpreting—parsing—the user's request)
        preserve
        (code that destroys the data)
        (code that makes the first part of the calculation)
        restore, preserve
        (code that makes the second part of the calculation)
        (code that reports the result)
end
```

The programmer could also have coded a `restore` on one line and a `preserve` on the next. It would have the same result but would be inefficient, because Stata would then rewrite the data to disk. `restore, preserve` tells Stata to reload the data but to leave the copy on disk for ultimate restoration.

◁

▷ Example 4

A programmer is writing a program that intends to change the user's data in memory—the damage the programmer is about to do is not damage at all. Nevertheless, if the user pressed *Break* while the programmer was in the midst of the machinations, what would be left in memory would be useless. The programmatically correct outline is

```
program myprog
        (code for interpreting—parsing—the user's request)
        preserve
        (code that reforms the data)
        restore, not
end
```

Before undertaking the reformation, the programmer smartly preserves the data. When everything is complete, the programmer cancels the restoration by typing `restore, not`.

◁

❑ Technical note

We said above that with `set max_preservemem`, if you set *amt* to `0b` (0 bytes), `preserve` will use disk storage. In fact, if you set *amt* to anything less than the size of one of Stata's data segments (see `set segmentsize` in [D] **memory**), `preserve` will always use disk storage. You can type `query memory` to see the current `segmentsize` and `max_preservemem` settings.

❑

❑ Technical note

`preserve, changed` is best avoided, although it is very fast. `preserve, changed` does not preserve the data; it merely records whether the data have changed since the data were last saved (as mentioned by `describe` and as checked by `exit` and `use` when the user does not also say `clear`) and restores the flag at the conclusion of the program. The programmer must ensure that the data really have not changed.

As long as the programs use temporary variables, as created by `tempvar` (see [P] **macro**), the changed-since-last-saved flag would not be changed anyway—Stata can track such temporary changes to the data that it will, itself, be able to undo. In fact, we cannot think of one use for `preserve, changed`, and included it only to preserve the happiness of our more imaginative users.

❑

Also see

[P] **nopreserve option** — nopreserve option

[D] **snapshot** — Save and restore data snapshots

[P] **macro** — Macro definition and manipulation