Description

_return sets aside and restores the contents of r().

_return hold stores under name the contents of r() and clears r(). If name is a name obtained from tempname, name will be dropped automatically at the program’s conclusion, if it is not automatically or explicitly dropped before that.

_return restore restores from name the contents of r() and, unless option hold is specified, drops name.

_return drop removes from memory (drops) name or, if _all is specified, all _return names currently saved.

_return dir lists the names currently set aside by _return.

Syntax

Set aside contents of r()

_return hold name

Restore contents of r() from name

_return restore name [, hold ]

Drop specified _return name

_return drop {name | _all}

List names currently stored by _return

_return dir

Option

hold, specified with _return restore, specifies that results continue to be held so that they can be _return restored later, as well. If the option is not specified, the specified results are restored and name is dropped.
Remarks and examples

_return is rarely necessary. Most programs open with

```stata
program example
    version 16.1
    syntax ...
    marksample touse
    if ""exp"" != "" {
        touse e
        qui generate double 'e' = 'exp' if 'touse'
    }
    ... (code to calculate final results)...
end
```

In the program above, no commands are given that change the contents of _r_() until all parsing is complete and the _if__ exp__ and _=exp__ are evaluated. Thus the user can type

```stata
. summarize myvar
. example ... if myvar>r(mean) ...
```

and the results will be as the user expects.

Some programs, however, have nonstandard and complicated syntax, and in the process of deciphering that syntax, other _r_-class commands might be run before the user-specified expressions are evaluated. Consider a command that reads

```stata
program example2
    version 16.1
    ... (commands that parse)...
    ... (_r_() might be reset at this stage)...
    ... commands that evaluate user-specified expressions...
    tempvar touse
    mark 'touse' 'if'
    tempvar v1 v2
    generate double 'v1' = 'exp1' if 'touse'
        // 'exp1' specified by user
    generate double 'v2' = 'exp2' if 'touse'
        // 'exp2' specified by user
    ... (code to calculate final results)...
end
```

Here it would be a disaster if the user typed

```stata
. summarize myvar
. example2 ... if myvar>r(mean) ...
```

because _r(mean)_ would not mean what the user expected it to mean, which is the mean of _myvar_. The solution to this problem is to code the following:
In the above example, we hold on to the contents of $r()$ in ‘myr’ and then later bring them back.

**Stored results**

`_return restore` restores in $r()$ those results that were stored in $r()$ when `return hold` was executed.

**Also see**

[P] `return` — Return stored results