return — Preserve stored results

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

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.

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
\[
\text{\texttt{program example}} \\
\text{\texttt{version 13}} \\
\text{\texttt{syntax ...}} \\
\text{\texttt{marksample touse}} \\
\text{\texttt{if "'exp'"; !="" {}} \\
\text{\texttt{touse e}} \\
\text{\texttt{qui gen double 'e' = 'exp' if 'touse'}} \\
\text{\texttt{}} \\
\text{\texttt{... (code to calculate final results) ...}} \\
\text{\texttt{end}}
\]

In the program above, no commands are given that change the contents of \texttt{r()} until all parsing is complete and the \texttt{if} \texttt{exp} and \texttt{=exp} are evaluated. Thus the user can type
\[
\text{\texttt{. summarize myvar}} \\
\text{\texttt{. example ... if myvar>r(mean) ...}}
\]
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 \texttt{r-class} commands might be run before the user-specified expressions are evaluated. Consider a command that reads
\[
\text{\texttt{program example2}} \\
\text{\texttt{version 13}} \\
\text{\texttt{... (commands that parse)...}} \\
\text{\texttt{... (r() might be reset at this stage)...}} \\
\text{\texttt{... commands that evaluate user-specified expressions...}} \\
\text{\texttt{tempvar touse}} \\
\text{\texttt{mark 'touse' 'if'}} \\
\text{\texttt{tempvar v1 v2}} \\
\text{\texttt{gen double 'v1' = 'exp1' if 'touse'}} \\
\text{\texttt{... (code to calculate final results)...}}
\]

\[
\text{\texttt{end}}
\]

Here it would be a disaster if the user typed
\[
\text{\texttt{. summarize myvar}} \\
\text{\texttt{. example2 ... if myvar>r(mean) ...}}
\]
because \texttt{r(mean)} would not mean what the user expected it to mean, which is the mean of \texttt{myvar}. The solution to this problem is to code the following:
The example code demonstrates the use of the \_return hold and \_return restore commands. The code snippet includes:

- Declaration of a temporary name `myr` to hold the contents of `r()`.
- Use of `\_return hold 'myr'` to hold the contents of `r()` in `myr`.
- The contents of `r()` might be reset at this stage.
- Evaluation of user-specified expressions.
- Use of `\_return restore 'myr'` to restore the contents of `r()` from `myr`.

The code also includes:

- Declaration of temporary variables `touse`, `v1`, and `v2`.
- Marking `touse` as a condition using `mark 'touse' 'if'`.
- Generation of variables `v1` and `v2` from expressions `exp1` and `exp2`.

In summary, `\_return hold` is used to hold the contents of `r()` in a temporary name, and `\_return restore` is used to restore the contents of `r()` from the temporary name. This is useful for preserving results that may be needed later in the program.

### 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