

Stata tip 66: **ds**—A hidden gem

Martin Weiss
University of Tuebingen
Tuebingen, Germany
martin.weiss@uni-tuebingen.de

ds is one of a few dozen “undocumented” commands in Stata whose names are available in **help undocumented**. Contrary to the help file assertion that “an undocumented command is a command of very limited interest, usually only to Stata programmers”, **ds** is extremely helpful, both interactively and in programs. The main hurdle to its widespread adoption by Stata users seems to be limited awareness of its existence.

Stata users are generally familiar with the **describe** (see [D] **describe**) command. **describe** allows you to gain a rapid overview of the dataset in memory, or with the **using** modifier, a dataset residing on a hard disk or available on the Internet. **ds** also allows that with its **detail** option, omitting only the general information provided in the header of the output of the **describe** command. For example,

```
. sysuse uslifeexp2.dta
(U.S. life expectancy, 1900–1940)
. describe
Contains data from C:\Program Files\Stata10\ado\base/u/uslifeexp2.dta
  obs:          41                U.S. life expectancy, 1900–1940
  vars:          2                2 Apr 2007 14:39
  size:          574 (99.9% of memory free)  (_dta has notes)
```

variable name	storage type	display format	value label	variable label
year	int	%9.0g		Year
le	float	%9.0g		life expectancy

```
Sorted by: year
. ds, detail
```

variable name	storage type	display format	value label	variable label
year	int	%9.0g		Year
le	float	%9.0g		life expectancy

More importantly, **ds** also provides the means of identifying subsets of variables with specified properties. It complements **lookfor** (see [D] **lookfor**), which allows you to search for certain strings in variable names or variable labels. **ds** enhances this functionality extensively, letting you specify

- certain types of variables;
- whether variable labels, value labels, and characteristics have been attached, and if so, whether they match certain patterns; and
- variables with specific formats.

A further systematic feature is that you can specify either the subset of variables satisfying particular properties or the complementary subset that does not satisfy those properties. As a simple example of the latter, when you are using the `auto.dta`, the command `ds make, not` specifies all variables other than `make`.

These capabilities prove particularly handy with large or poorly known datasets. As a simple example, pretend you were not familiar with the `auto` dataset and were looking for string variables.

```
. sysuse auto.dta
(1978 Automobile Data)
. * show all variables featuring type string
. ds, has(type string)
make
```

While `describe` would list the variable types, leaving the task of finding a certain type to you, `ds` can provide precisely what you were looking for. Despite its “un-documented” status, a dialog box can ease navigation through the intricacies of this command: to try the dialog box, type `db ds`. Beyond the results shown as usual, `ds` also leaves behind a list of variables found in `r(varlist)`, which is available for use by subsequent commands, such as `list` (see [D] `list`) or `summarize` (see [R] `summarize`). Many of the properties that you can search for with `ds` can also be extracted with extended macro functions; see [P] `macro`. To illustrate, consider the `voter` dataset shipped with Stata.

```
. sysuse voter.dta
. * show all variables with value labels attached
. ds, has(vall)
candidat inc
. * show all variables not of type float
. ds, not(type float)
candidat inc pfrac pop
. * show mean of all variables with format %10.0g
. ds, has(format %10.0g)
pfrac pop
. tabstat `r(varlist)´
```

stats	pfrac	pop
mean	6.733333	104299.3

The help file accessed by typing `help ds` gives several more examples. The help file accessed by typing `help varfind_pat_examp` explains the use of wildcards within the specification of patterns.