Title

stata.com

_datasignature — Determine whether data have changed

Syntax | Description | Options | Remarks and examples
Stored results | Reference | Also see

Syntax

`_datasignature` [*varlist*] [*if*] [*in*] [, *options*]

options	Description
`fast`	perform calculation in machine-dependent way
`esample`	restrict to estimation sample
`nonames`	do not include checksum for variable names
`nodefault`	treat empty *varlist* as null

Description

`_datasignature` calculates, displays, and stores in `r(_datasignature)` checksums of the data, forming a signature. A signature might be

```
162:11(12321):2725060400:4007406597
```

The signature can be stored and later used to determine whether the data have changed.

Options

`fast` specifies that the checksum calculation be made in a faster, less computationally intensive, and machine-dependent way. With this option, `_datasignature` runs faster on all computers and can run in less than one-third of the time on some computers. The result can be compared with other `fast` computations made on the same computer, and computers of the same make, but not across computers of different makes. See *Remarks and examples* below.

`esample` specifies that the checksum be calculated on the data for which `e(sample)` = 1. Coding

```
_datasignature `varlist', esample
```

or

```
_datasignature `varlist' if e(sample)
```

produces the same result. The former is a little quicker. If the `esample` option is specified, `if` *exp* may not be specified.

`nonames` specifies that the variable-names checksum in the signature be omitted. Rather than the signature being 74:12(71728):2814604011:3381794779, it would be 74:12:2814604011:3381794779. This option is useful when you do not care about the names or you know that the names have changed, such as when using temporary variables.

`nodefault` specifies that when *varlist* is not specified, it be taken to mean no variables rather than all variables in the dataset. Thus you may code

```
_datasignature `modelvars', nodefault
```

and obtain desired results even if `` `modelvars' `` expands to nothing.

Remarks and examples

stata.com

For an introduction to data signatures, see [D] **datasignature**. To briefly summarize:

- A signature is a short string that is calculated from a dataset, such as 74:12(71728):3831085005:1395876116. If a dataset has the same signature at two different times, then it is highly likely that the data have not changed. If a dataset has a different signature, then it is certain that the data have changed.
- An example data signature is 74:12(71728):3831085005:1395876116. The components are
  a. `74`, the number of observations;
  b. `12`, the number of variables;
  c. `71728`, a checksum function of the variable names and the order in which they occur; and
  d. `3831085005` and `1395876116`, checksum functions of the values of the variables, calculated two different ways.
- Signatures are functions of
  a. the number of observations and number of variables in the data;
  b. the values of the variables;
  c. the names of the variables;
  d. the order in which the variables occur in the dataset if *varlist* is not specified, or in *varlist* if it is; and
  e. the storage types of the variables.

  If any of these change, the signature changes. The signature is not a function of the sort order of the data. The signature is not a function of variable labels, value labels, contents of characteristics, and the like.

Programs sometimes need to verify that they are running on the same data at two different times. This verification is especially common with estimation commands, where the estimation is performed by one command and postestimation analyses by another. To ensure that the data have not changed, one obtains the signature at the time of estimation and then compares that with the signature obtained when the postestimation command is run. See [P] **signestimationsample** for an example.

If you are producing signatures for use within a Stata session—signatures that will not be written to disk and thus cannot possibly be transferred to different computers—specify `_datasignature`'s `fast` option. On some computers, `_datasignature` can run in less than one-third of the time if this option is specified.

`_datasignature, fast` is faster for two reasons: (1) the option uses a less computationally intensive algorithm and (2) the computation is made in a machine-dependent way. The first affects the quality of the signature, and the second does not.

Remember that signatures have two checksums for the data. When `fast` is specified, a different, inferior algorithm is substituted for the second checksum. In the `fast` case, the second signature is not conditionally independent of the first and thus does not provide 48 bits of additional information; it probably provides around 24 bits. The default second checksum calculation was selected to catch problems that the first calculation does not catch. In the `fast` case, the second checksum does not have that property. These details make the `fast` signature sound markedly inferior. Nevertheless, the first checksum calculation, which is used both in the default and the `fast` cases, is good, and when `_datasignature` was written, we considered using only the first calculation in both cases. We believe that, for within-session testing, where one does not have to guard against changes produced by an intelligent enemy who may be trying to fool you, the first checksum alone is adequate. The inferior second checksum we include in the `fast` case provides more protection than we think necessary.

The second difference has nothing to do with quality. Modern computers come in two types: those that record least-significant bytes (LSBs) first and those that record most-significant bytes (MSBs) first. Intel-based computers, for instance, are usually LSB, whereas Sun computers are MSB.

By default, `_datasignature` makes the checksum calculation in an LSB way, even on MSB computers. MSB computers must therefore go to extra work to emulate the LSB calculation, and so `_datasignature` runs slower on them.

When you specify `fast`, `_datasignature` calculates the checksum the native way. The checksum is every bit as good, but the checksum produced will be different on MSB computers. If you merely store the signature in memory for use later in the session, however, that does not matter.

Stored results

`_datasignature` stores the following in `r()`:

Macros

`r(datasignature)`	the signature

Reference

Gould, W. W. 2006. Stata tip 35: Detecting whether data have changed. *Stata Journal* 6: 428–429.

Also see

[D] **datasignature** — Determine whether data have changed

[P] **signestimationsample** — Determine whether the estimation sample has changed

[D] **compare** — Compare two variables

[D] **cf** — Compare two datasets