

## hash1() — Jenkins's one-at-a-time hash function

Description	Syntax	Remarks and examples	Conformability
Diagnostics	References	Also see	

## Description

`hash1(x)` returns Jenkins's one-at-a-time hash calculated over the bytes of  $x$ ;  $0 \leq \text{hash1}(x) \leq 4,294,967,295$ .

`hash1(x, n)` returns Jenkins's one-at-a-time hash scaled to  $1 \leq \text{hash1}(x, n) \leq n$ , assuming  $n < .$  (missing). `hash1(x, .)` is equivalent to `hash1(x)`.

`hash1(x, n, byteorder)` returns `hash1(x, n)` performed on the bytes of  $x$  ordered as they would be on a HILO computer (`byteorder = 1`), or as they would be on a LOHI computer (`byteorder = 2`), or as they are on this computer (`byteorder ≥ .`). See [M-5] [byteorder\(\)](#) for a definition of byte order.

In all cases, the values returned by `hash1()` are integers.

## Syntax

*real scalar* `hash1`( $x$  [ $,$  *real scalar*  $n$  [ $,$  *real scalar* `byteorder` ] ])

where

$x$ : any type except `struct` and any dimension

$n$ :  $1 \leq n \leq 2,147,483,647$  or `.`; default is `.` (missing)

`byteorder`: 1 (HILO), 2 (LOHI), `.` (natural byte order); default `.` (missing)

## Remarks and examples

stata.com

Calculation is significantly faster using the natural byte order of the computer. Argument `byteorder` is included for those rare cases when it is important to calculate the same hash value across different computers, which in the case of `hash1()` is mainly for testing. `hash1()`, being a one-at-a-time method, is not sufficient for constructing digital signatures. It is sufficient for constructing hash tables; see [M-5] [asarray\(\)](#), in which case, byte order is irrelevant. Also note that because strings occur in the same order on all computers, the value of `byteorder` is irrelevant when  $x$  is a string.

For instance,

```
: hash1("this"), hash1("this",.,1), hash1("this",.,2)
      1                2                3
```

```
1 | 2385389520 2385389520 2385389520
```

```
: hash1(15), hash1(15,.,1), hash1(15,.,2)
      1                2                3
```

```
1 | 463405819 3338064604 463405819
```

The computer on which this example was run is evidently *byteorder* = 2, meaning LOHI, or least-significant byte first.

In a Mata context, it is the two-argument form of `hash1()` that is most useful. In that form, the full result is mapped onto  $[1, n]$ :

$$\text{hash1}(x, n) = \text{floor}((\text{hash1}(x)/4294967295)*n) + 1$$

For instance,

```
: hash1("this", 10)
6
: hash1(15, 10)
2
```

The result of `hash1(x, 10)` could be used directly to index a  $10 \times 1$  array.

## Conformability

```
hash1(x, n, byteorder):
  x:      r × c
  n:      1 × 1      (optional)
  byteorder: 1 × 1      (optional)
  result:  1 × 1
```

## Diagnostics

None.

Note that `hash1(x[ , ... ])` never returns a missing result, even if *x* is or contains a missing value. In the missing case, the hash value is calculated of the missing value. Also note that *x* can be a vector or a matrix, in which case the result is calculated over the elements aligned rowwise as if they were a single element. Thus `hash1(("a", "b")) == hash1("ab")`.

## References

- Jenkins, B. 1997. *Dr. Dobbs's Journal*. Algorithm alley: Hash functions. <http://www.ddj.com/184410284>.
- . unknown. A hash function for hash table lookup. <http://www.burtleburtle.net/bob/hash/doobs.html>.

## Also see

- [M-5] [asarray\(\)](#) — Associative arrays
- [M-4] [Programming](#) — Programming functions