



**AgEcon** SEARCH  
RESEARCH IN AGRICULTURAL & APPLIED ECONOMICS

*The World's Largest Open Access Agricultural & Applied Economics Digital Library*

**This document is discoverable and free to researchers across the globe due to the work of AgEcon Search.**

**Help ensure our sustainability.**

Give to AgEcon Search

AgEcon Search  
<http://ageconsearch.umn.edu>  
[aesearch@umn.edu](mailto:aesearch@umn.edu)

*Papers downloaded from **AgEcon Search** may be used for non-commercial purposes and personal study only. No other use, including posting to another Internet site, is permitted without permission from the copyright owner (not AgEcon Search), or as allowed under the provisions of Fair Use, U.S. Copyright Act, Title 17 U.S.C.*

# THE STATA JOURNAL

**Guest Editor**

David M. Drukker  
StataCorp

**Editor**

H. Joseph Newton  
Department of Statistics  
Texas A & M University  
College Station, Texas 77843  
979-845-3142; FAX 979-845-3144  
jnewton@stata-journal.com

**Editor**

Nicholas J. Cox  
Geography Department  
Durham University  
South Road  
Durham City DH1 3LE UK  
n.j.cox@stata-journal.com

**Associate Editors**

Christopher Baum  
Boston College  
Rino Bellocco  
Karolinska Institutet, Sweden and  
Univ. degli Studi di Milano-Bicocca, Italy  
David Clayton  
Cambridge Inst. for Medical Research  
Mario A. Cleves  
Univ. of Arkansas for Medical Sciences  
William D. Dupont  
Vanderbilt University  
Charles Franklin  
University of Wisconsin, Madison  
Joanne M. Garrett  
University of North Carolina  
Allan Gregory  
Queen's University  
James Hardin  
University of South Carolina  
Ben Jann  
ETH Zurich, Switzerland  
Stephen Jenkins  
University of Essex  
Ulrich Kohler  
WZB, Berlin  
Jens Lauritsen  
Odense University Hospital

Stanley Lemeshow  
Ohio State University  
J. Scott Long  
Indiana University  
Thomas Lumley  
University of Washington, Seattle  
Roger Newson  
Imperial College, London  
Marcello Pagano  
Harvard School of Public Health  
Sophia Rabe-Hesketh  
University of California, Berkeley  
J. Patrick Royston  
MRC Clinical Trials Unit, London  
Philip Ryan  
University of Adelaide  
Mark E. Schaffer  
Heriot-Watt University, Edinburgh  
Jeroen Weesie  
Utrecht University  
Nicholas J. G. Winter  
Cornell University  
Jeffrey Wooldridge  
Michigan State University

**Stata Press Production Manager****Stata Press Copy Editor**

Lisa Gilmore  
Gabe Waggoner

**Copyright Statement:** The Stata Journal and the contents of the supporting files (programs, datasets, and help files) are copyright © by StataCorp LP. The contents of the supporting files (programs, datasets, and help files) may be copied or reproduced by any means whatsoever, in whole or in part, as long as any copy or reproduction includes attribution to both (1) the author and (2) the Stata Journal.

The articles appearing in the Stata Journal may be copied or reproduced as printed copies, in whole or in part, as long as any copy or reproduction includes attribution to both (1) the author and (2) the Stata Journal.

Written permission must be obtained from StataCorp if you wish to make electronic copies of the insertions. This precludes placing electronic copies of the Stata Journal, in whole or in part, on publicly accessible web sites, file servers, or other locations where the copy may be accessed by anyone other than the subscriber.

Users of any of the software, ideas, data, or other materials published in the Stata Journal or the supporting files understand that such use is made without warranty of any kind, by either the Stata Journal, the author, or StataCorp. In particular, there is no warranty of fitness of purpose or merchantability, nor for special, incidental, or consequential damages such as loss of profits. The purpose of the Stata Journal is to promote free communication among Stata users.

The *Stata Journal*, electronic version (ISSN 1536-8734) is a publication of Stata Press, and Stata is a registered trademark of StataCorp LP.

## Stata tip 33: Sweet sixteen: Hexadecimal formats and precision problems

Nicholas J. Cox  
Department of Geography  
Durham University  
Durham City, UK  
n.j.cox@durham.ac.uk

Computer users generally supply numeric inputs as decimals and expect numerical outputs as decimals. But underneath the mapping from inputs to outputs lies software (such as Stata) and hardware that are really working with binary representations of those decimals. Much ingenuity goes into ensuring that conversions between decimal and binary are invisible to you, but occasionally you may see apparently strange side effects of this fact. This problem is documented in [U] **13.10 Precision and problems therein**, but it still often bites and puzzles Stata users. This tip emphasizes that the special hexadecimal format `%21x` can be useful in understanding what is happening. The format is also documented, but in just one place, [U] **12.5.1 Numeric formats**. Decimal formats such as `%23.18f` can also be helpful for investigating precision problems.

Binary representations of numbers, using just the two digits 0 and 1, can be difficult for people to interpret without extra calculations. The great advantage of a hexadecimal format, using base 16 (i.e.,  $2^4$ ), is that it is closer to base 10 representations while remaining truthful about what can be held in memory as a representation of a number. It is conventional to use the decimal digits 0–9 and the extra digits a–f when base 16 is used. Thus a represents 10 and f represents 15. Hence, at its simplest, hexadecimal 10 represents decimal 16, hexadecimal 11 represents decimal 17, and so forth. (Think of 11 as  $1 \times 16^1 + 1 \times 16^0$ , for example.) In practice, we want to hold fractions and, as far as possible, some extremely large and extremely small numbers. The general format of a hexadecimally represented number in Stata is thus `mXp`, to be read as  $m \times 2^p$ . Thus if you use the format `%21x` with `display`, you can see examples:

```
. di %21x 1
+1.0000000000000X+000
. di %21x -16
-1.0000000000000X+004
. di %21x 1/16
+1.0000000000000X-004
```

You see that 1,  $-16$ , and  $1/16$  are, respectively,  $1 \times 2^0$ ,  $-1 \times 2^4$ , and  $1 \times 2^{-4}$ .

The special format is useful to others besides the numerical analysts mentioned in [U] **12.5.1 Numeric formats**. If you encounter puzzling results, looking at the numbers in question should help clarify what Stata is doing and why it does not match your expectation.

Users get bitten in two main ways. First, they forget that most of the decimal digits .1, .2, .3, .4, .5, .6, .7, .8, and .9 cannot be held exactly. Of these, only .5 ( $1/2$ ) can possibly be represented exactly by a binary approximation; all the others must be held approximately only—regardless of how many bytes are used. To convince yourself of this, see that, e.g., 42.5 can be held exactly,

```
. di %21x 42.5
+1.5400000000000X+005
. di (1 + 5/16 + 4/256) * 2^5
42.5
```

whereas 42.1 cannot be held exactly,

```
. di %21x 42.1
+1.50cccccccccdX+005
. di %23.18f 42.1
42.10000000000001421
```

Close, but not exact. Second, users forget that although very large or very small numbers can be held approximately, not all possible numbers can be distinguished, even when those numbers are integers within the limits of the variable type being used.

A common source of misery is trying to hold nine-digit integers in numeric variables. If these are identifiers, holding them as **str9** variables is a good idea, but let us focus on what often happens when users read such integers into numeric variables. This experiment shows the problems that can ensue.

```
. gen pinid = 123456789
. di %9.0f pinid[1]
123456792
. di %21x pinid[1]
+1.d6f346000000X+01a
```

Stata did not complain, but it did not oblige. The value is off by 3. You will see that the value held is a multiple of 4, as the last two digits 92 are divisible by 4. Did we or Stata do something stupid? Can we fix it?

```
. replace pinid = pinid - 3
(0 real changes made)
```

Trying to subtract 3 gives us the same number, so far as Stata is concerned. What is going on? By default, Stata is using a **float** variable. See [D] **data types** if you want more information. At this size of number, such a variable can hold only multiples of 4 exactly, so we lose many final digits. The remedy, if a numeric variable is needed, is to use a **long** or **double** storage type instead.