



**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.*

*No endorsement of AgEcon Search or its fundraising activities by the author(s) of the following work or their employer(s) is intended or implied.*

## Sort a list of items

Patrick Royston  
UK Medical Research Council  
patrick.royston@ctu.mrc.ac.uk

**Abstract.** The command `listsort` for sorting the contents of a set of local macros is introduced and illustrated.

**Keywords:** dm0001, sorting, local macros

### Introduction

Sometimes when you are writing an ado-file, you wish to sort some numbers (or less often, strings) that you have stored in local macros. You can, of course, transfer each of these items to a variable, sort that, transfer the results back to local macros, and then delete the variable. This is obviously tedious and inefficient; you also have to handle the fact that you have reordered the data, which may have side effects elsewhere.

`listsort` is intended to perform such tasks without invoking Stata's variable `sort`.

### Syntax

```
listsort "item [ item ... ]" [, lexicographic reverse ]
```

### Options

`lexicographic` performs a lexicographic (alphanumeric) sort. Case is respected.

`reverse` sorts in descending order (default is ascending).

### Example

`listsort` stores results in s-class macros. The entire list is placed in `s(list)` and the individual items in `s(i1)`, `s(i2)`, etc. Typical code using `listsort` might be as follows. Suppose you have 'n' local macros 'p1', ..., 'pn' that contain numbers you want to sort and replace in the same local macros. You could code as follows:

```
local i 1
local p
/* Concatenate contents of macros */
while 'i'<='n' {
    local p 'p' 'p'i'
    local i='i'+1
}
listsort "'p'"
/* Recover results */
local i 1
while 'i'<='n' {
    local p'i' 's(i'i)'
    local i='i'+1
}
```

The recovery could equally have been done as follows—a matter of taste:

```
local i 1
while 'i'<='n' {
    local p'i' : word 'i' of 's(list)'
    local i='i'+1
}
```

As a result, macro 'p1' will contain the smallest of the original items, 'p2' the second smallest, and so on.

#### **About the Author**

Patrick Royston is a medical statistician of 25 years of experience, with a strong interest in biostatistical methodology and in statistical computing and algorithms. At present he works in clinical trials and related research issues in cancer. Currently he is focusing on problems of model building and validation with survival data, including prognostic factors studies, on parametric modeling of survival data, and on novel trial designs.