STATA
TECHNICAL
BULLETIN

March 2001
STB-60

A publication to promote communication among Stata users

Editor

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

Associate Editors

Christopher F. Baum, Boston College
Nicholas J. Cox, University of Durham
Joanne M. Garrett, University of North Carolina
Marcello Pagano, Harvard School of Public Health
J. Patrick Royston, UK Medical Research Council
Jeroen Weesie, Utrecht University

Subscriptions are available from Stata Corporation, email stata@stata.com, telephone 979-696-4600 or 800-STATAPC, fax 979-696-4601. Current subscription prices are posted at www.stata.com/bookstore/stb.html.

Previous Issues are available individually from StataCorp. See www.stata.com/bookstore/stbj.html for details.

Submissions to the STB, including submissions to the supporting files (programs, datasets, and help files), are on a nonexclusive, free-use basis. In particular, the author grants to StataCorp the nonexclusive right to copyright and distribute the material in accordance with the Copyright Statement below. The author also grants to StataCorp the right to freely use the ideas, including communication of the ideas to other parties, even if the material is never published in the STB. Submissions should be addressed to the Editor. Submission guidelines can be obtained from either the editor or StataCorp.

The *Stata Technical Bulletin* (ISSN 1097-8879) is published six times per year by Stata Corporation. Stata is a registered trademark of Stata Corporation.

Contents of this issue

dm67.1	Enhancements to numbers of missing and present values

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

Abstract: `nmissing` and `npresent`, originally published in Cox (1999), are extended, so that they may report on numbers of missing or present values in each observation, as well as in each variable. The `min()` option specifying the minimum number to be reported is also extended.

Keywords: describing variables, missing values, data management.

Syntax

`nmissing` [*varlist*] [`if` *exp*] [`in` *range*] [`, min(#) obs`]

`npresent` [*varlist*] [`if` *exp*] [`in` *range*] [`, min(#) obs`]

Description

`nmissing` lists the number of missing values in each variable in *varlist* (or optionally in each observation). Missing means `.` for numeric variables and the empty string `""` for string variables.

`npresent` lists the number of present (nonmissing) values in each variable in *varlist* (or optionally in each observation).

Options

`min(#)` specifies that only numbers at least # should be listed. The default is 1. The argument may be any positive integer or expression evaluating to a positive integer. Expressions may include `N`. As a special case, `all` means the number of variables in *varlist* or, with `obs`, the number of observations.

`obs` specifies that counting is for observations, not variables.

Remarks

`nmissing` reports on numbers of missing values. When called with no arguments, it reports on the whole dataset, including both numeric and string variables. If a *varlist* is specified, or the minimum number of values to be reported is specified by the `min()` option, then the focus is restricted accordingly.

`npresent` is the complementary command that reports on present (nonmissing) values.

`nmissing` and `npresent` were originally published in Cox (1999), which includes some comparative comments on the reporting of missing values by `summarize`, `inspect` and `codebook`.

In this update, the new `obs` option extends reporting from columnwise (number of missing values in each variable) to rowwise (number of missing values in each observation). This can be valuable for a variety of reasons. Observations with many, or even all, values missing may be worth special note.

In addition, the `min()` option is now smarter; it understands expressions which evaluate to integers, including those in terms of the total number of observations in the dataset `N`. As a special case `min(all)` is understood as the number of variables in *varlist*, or, with `obs`, as the total number of observations under scrutiny, that is, excluding any observations not satisfying any `if` or `in` restrictions. In the latter case, note a subtle distinction, as illustrated with the auto dataset: for that set `N` is 74, the total number of observations, but `all` is 22 with `if foreign`, and may well vary depending on whatever other restrictions are specified.

Examples

With the familiar auto dataset,

```
. nmissing
```

yields

```
 rep78       5
```

while

```
. nmissing if foreign
```

yields

```
 rep78       1
```

while

```
. nmissing, obs min(all)
```

yields a report on those observations for which all values are missing (none in this dataset).

Reference

Cox, N. J. 1999. dm67: Numbers of missing and present values. *Stata Technical Bulletin* 49: 7–8. Reprinted in *Stata Technical Bulletin Reprints*, vol. 9, pp. 26–27.

dm78.1 | Describing variables in memory: update to Stata 7

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

Abstract: `ds2`, a variant on `ds` originally published in Cox (2000), is updated to Stata 7. Longer variable names are abbreviated to at most 8 characters, as in the results of `ds`.

Keywords: describing variables, variable names, data types, value labels, `ds`, data management.

Description

`ds2` lists variable names in a compact format (or, optionally, in the same format as `describe`). It is a variant on `ds`.

`ds2` was originally published in Cox (2000). For details, please see that insert.

This update attends to just one important point: the longer variable names which are allowed in Stata 7 are abbreviated on display to at most 8 characters, just as in the results of `ds` in official Stata.

Reference

Cox, N. J. 2000. dm78: Describing variables in memory. *Stata Technical Bulletin* 56: 2–4.

dm87 | Calculating the row product of observations

Philip Ryan, University of Adelaide, Australia, philip.ryan@adelaide.edu.au

Abstract: This insert describes a new option in `egen` which creates a new variable whose values are the row product of observations.

Keywords: `egen`, row product.

Syntax

`egen` *newvar* `= rprod(`*varlist*`)` [`if` *exp*] [`in` *range*] [`, pmiss({ignore | missing | 1})`]

Description

`rprod` provides a multiplicative function for `egen` analogous to the additive `rsum` function. The row product of observations of *varlist* meeting optional `if` and `in` conditions is returned in *newvar*. `rprod` insists that *newvar* be calculated in double precision.

Option

`pmiss({ignore | missing | 1})` specifies what is to be the product of observations at least one of which is missing.

`pmiss(ignore)`, the default, ignores missing values and returns the product of all nonmissing observations in the row. If all values are missing, then missing is returned.

`pmiss(missing)` returns missing for the product of observations if any observation in the row is missing. This holds even if one or more of the nonmissing observations is zero.

`pmiss(1)` returns 1 for the product of observations if all observations in the row are missing. Otherwise it returns the product of all nonmissing values. This choice would rarely be made (see the Appendix in Ryan 1999).

Note that a missing value is returned for rows excluded by `if` or `in` qualifiers.

Examples

We consider the 11 observations on variables `a`, `b`, and `c` listed below:

```
. list
            a         b         c
 1.         2         3         4
 2.         5         0         1
 3.         0         0         0
 4.         .         .         .
 5.         1         .         1
 6.         8         3         .
 7.         0         .         0
 8.        -3         4         5
 9.        -1        -7         2
10.         .         .        -4
11.       .75        -2       .25
```

We create three new variables using `rprod` in various ways.

```
. egen d = rprod(a-c)
(1 missing value generated)
. egen k = rprod(a-c) , pmiss(missing)
(5 missing values generated)
. egen q = rprod(a-c) in 8/11
(7 missing values generated)
. list
            a         b         c          d          k          q
 1.         2         3         4         24         24          .
 2.         5         0         1          0          0          .
 3.         0         0         0          0          0          .
 4.         .         .         .          .          .          .
 5.         1         .         1          1          .          .
 6.         8         3         .         24          .          .
 7.         0         .         0          0          .          .
 8.        -3         4         5        -60        -60        -60
 9.        -1        -7         2         14         14         14
10.         .         .        -4         -4          .         -4
11.       .75        -2       .25       -.375      -.375      -.375
```

Acknowledgments

This program was written in response to a thread on the Statalist server started by Brian Jacob (ba-jacob@mail.consortium-chicago.org). Nick Cox (n.j.cox@durham.ac.uk) provided helpful comments.

Reference

Ryan, P. 1999. dm71: Calculating the product of observations. *Stata Technical Bulletin* 51: 3–4. Reprinted in *Stata Technical Bulletin Reprints*, vol. 9, pp. 45–48.

dm88	Renaming variables, multiply and systematically

Nicholas J. Cox, University of Durham, UK, n.j.cox@durham.ac.uk
Jeroen Weesie, Utrecht University, Netherlands, j.weesie@fss.uu.nl

Abstract: `renvars` is a command to rename variables, which will be useful for multiple changes of names, especially those which can be represented by a simple systematic rule, such as adding or changing of prefixes or postfixes. Both the ability to change several names at once and a variety of specific and general transformation options are supported. There are also inbuilt safety features; in particular, no variables will be renamed unless all new variable names specified or implied are acceptable as such.

Keywords: `renvars`, `rename`, variable names, data management.

Syntax

`renvars` [*varlist*]`\` *newvarlist* [`, display test`]

`renvars` [*varlist*]`,` *transformation_option* [`display test symbol(`*str*`)`]

where *transformation_option* is one of

upper

lower

prefix(*str*)

postfix(*str*) (synonym suffix(*str*))

presub(*str1 str2*)

postsub(*str1 str2*)

subst(*str1 str2*)

predrop(*#*)

postdrop(*#*)

trim(*#*)

map(*string_expression*)

Description

renvars renames the variables listed in *varlist*. If not specified, *varlist* defaults to _all.

renvars [*varlist*] \ *newvarlist* renames each variable to the corresponding new name in *newvarlist*.

renvars [*varlist*], *transformation_option* applies the transformation specified to each variable name in *varlist*.

renvars will not rename any variable unless all the new names specified are acceptable as new variable names when the command is issued. However, a "new" name that is just the same as the existing name presents no problem.

Variable labels, value labels, formats, and characteristics are maintained.

Options

One of the following *transformation_options* may be specified:

upper converts the variable names to uppercase.

lower converts the variable names to lowercase.

prefix(*str*) prefixes variable names with *str*.

postfix(*str*) postfixes variable names with *str*. suffix(*str*) is an exact synonym.

presub(*str1 str2*) replaces the leading string *str1* by *str2* in variable names. *str2* may be empty.

postsub(*str1 str2*) replaces the trailing string *str1* by *str2* in variable names. *str2* may be empty

subst(*str1 str2*) substitutes (all occurrences of) *str1* by *str2* in variable names. *str2* may be empty.

predrop(*#*) removes the first *#* characters from the variable names.

postdrop(*#*) removes the last *#* characters from variable names.

trim(*#*) keeps the first *#* characters in the variable names while dropping the remaining characters.

map(*string_expression*) specifies a rule for building new variable names from existing names. By default, @ is the placeholder for existing names. This placeholder can be changed by specifying symbol().

display specifies that each change is displayed.

test specifies that the name changes are shown but not performed.

symbol(*str*) specifies a symbol to be used as a placeholder for the existing name in the map expression. The default is @. The symbol used should not include characters used in existing variable names. It is difficult to imagine why you might want to use this option.

Discussion

Renaming variables is a common need in data management. The Stata command rename provides a means of renaming variables one at a time. A special supplementary command renpfix provides a systematic way of changing prefixes of several variable names; see [R] **rename**. Jenkins and Cox (2001) have recently written rensfix, which tackles the similar problem of

changing suffixes. Despite these two extras, there are many problems which require renaming several variables at once. You may wish either to specify lists of existing and new variable names or to apply some systematic rule to transform existing names into new names.

It may be useful to warn at the outset that if there is any doubt about the consequences of renaming, you should first run `renvars` with the option `test`. Then, `renvars` displays the name changes that would result, so you can verify that the consequences are as expected.

Note first that programming constructs such as `for` (see [R] **for**) or, in Stata 7, `foreach` (see [P] **foreach**) can be used to automate various multiple or systematic changes.

The easiest changes to make with `for` are one-to-one changes between a list of existing variable names and a corresponding list of new names. The pattern is

`for var` *varlist* `\ new` *newvarlist* `: rename X Y`

which maps each existing name in *varlist* to the corresponding name in *newvarlist*. A crucial detail here is that syntax checking within `for` ensures that first, the lists are of the same length (after expansion); and second, the variable names in the two lists are indeed all existing and all new (and legal) at the time of issuing the command. For many users, if not all, this detail is a valuable safety feature.

The easiest changes to make with `foreach` are systematic changes, whereby some rule maps existing variable names to new names. One minimal pattern is

```
foreach oldvar of varlist varlist {
    local newvar = function of 'oldvar'
    rename 'oldvar' 'newvar'
}
```

(Stata 6.0 users could write this in more long-winded terms using `while`.) The crucial differences here are twofold. Many different transformations can be put inside the `foreach` loop. If necessary, a transformation could be coded in two or more lines. In addition, in the version here, each new name is checked for validity only at the time of each `rename`. In the event of a problem, the `foreach` loop is exited with an error message. The consequence will typically be that some variables in *varlist* are renamed but not others. In some situations, users will be left with a mess, and the need to clear it up.

`renvars` provides an alternative, written to be, as far as possible, safe and smart. It permits two syntaxes: one in which new variable names are specified explicitly, and one in which a systematic rule is used to transform old names into new names. The underlying principle is that no variables will be renamed unless all new variable names specified or implied are acceptable as such. However, a "new" name that is just the same as the existing name presents no problem. For example, you may wish to make all variable names lowercase. Any variable name which is already lowercase is acceptable. Think of this as no new name (that is, one different and not in use) being implied.

The particular *transformation_options* supplied cover various common needs: namely, changing case; adding, changing, or deleting prefixes or postfixes; substituting one substring for another; and dropping characters at the beginning or end of variable names. In addition, a general `map()` option has been supplied. Minimalists will note that several of the specific transformation options are strictly redundant given `map()`. Thus, for example, `lower` is equivalent to `map(lower("@"))`. Experience indicates, however, that many users prefer to have options directly supplied for specific tasks, rather than be obliged to work out the syntax needed with a more general option. Conversely, more experienced users may appreciate the flexibility of `map()`.

Examples

```
. renvars v1-v4 \ id  time income status
. renvars MYVAR1 MYVAR2 MYVAR3, lower
. renvars v1-v10, upper
. renvars, pref(X) test
. renvars, subs(X Y)
. renvars, predrop(1)
. renvars, map("D_" + substr("@", 1, 6))
```

Reference

Jenkins, S. P. and N. J. Cox. 2001. dm83: Renaming variables: changing suffixes. *Stata Technical Bulletin* 59: 5–6.

dm89	Dropping variables or observations with missing values

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

Abstract: `dropmiss` drops variables or observations with all values (optionally any values) missing.

Keywords: dropping variables, dropping observations, missing values, data management.

Syntax

`dropmiss` [*varlist*] [`, any trim piasm`]

`dropmiss` [*varlist*] [`if` *exp*] [`in` *range*] `, obs` [`any trim piasm`]

Description

`dropmiss` drops those variables in *varlist* for which all (optionally any) observations have missing values, that is `.` for a numeric variable and `""` for a string variable.

`dropmiss` with the `obs` option drops those observations for which all (optionally any) variables in *varlist* have missing values.

With both syntaxes, *varlist* defaults to all variables.

Options

`any` specifies that variables or observations are to be dropped if any values are missing in those variables or observations. Note that this can have drastic effects and may even result in the dropping of all data.

`trim` specifies that whether string values are missing is to be determined after trimming leading and trailing spaces. For example, with `trim " "` (a single blank) would count as missing.

`piasm` specifies that Period Is Also String Missing, that is, `"."` counts as missing. Some Stata users use `"."` to indicate missing.

`trim piasm` would specify that, for example, `" . "` counts as missing.

`obs` specifies that observations are to be dropped for which values of *varlist* are missing.

Remarks

A common problem in data management is that values may be missing. In some cases, all the values in one or more variables or observations may be missing. This could arise for a variety of reasons. For example, many spreadsheet users find it useful to have blank rows or columns as separators between different blocks of data. When read into Stata, such blanks will typically be translated into missing values, which serve little or no purpose, so it may be desired to drop them from memory.

Before doing this, it is important to be clear which variables and which observations contain missing values, and several Stata commands may be used to determine this, including `summarize`, `list`, `inspect`, and `codebook`. In addition, the command `nmissing` (Cox 1999, 2001) produces relatively concise reports on numbers of missing values.

It is easy enough to type, for example, `drop if` *varname* `== .` if that is desired for individual variables, but it is also helpful to have more systematic ways of handling the problem, especially for large datasets with many observations and/or many variables.

`dropmiss` treats both numeric and string variables. In Stata, missing values are, syntactically, numeric missing or `.` for numeric variables and empty strings `""` for string variables. However, many users might have a broader view of missing values. First, string values consisting of just one or more spaces might be regarded as missing just as much as empty strings. The option `trim` trims values of leading and trailing spaces before determining whether they are empty. Second, some Stata users treat `"."` as indicating missing string values. The option `piasm` (an acronym for `p`eriod `i`s `a`lso `s`tring `m`issing) treats these as missing, in addition to empty strings. `trim` and `piasm` may be combined.

Most commonly, users wish to drop variables or observations for which all values are missing, and this is the default for `dropmiss`. Note that this is determined for any *varlist* specified. Thus, if you type `dropmiss a b c`, then any one of those variables is dropped, if, and only if, all values are missing for that variable. More crucially, `dropmiss a b c, obs` drops any observation, if, and only if, `a`, `b` and `c` are all missing for that observation; no account is taken of any other variables in the dataset. If no *varlist* is specified, then all variables in the dataset are considered.

Less commonly, users wish to drop variables or observations for which any values (meaning one or more) are missing. This is possible with the `any` option. It is, naturally, a much more drastic possibility and may even result in dropping all of the dataset. Be sure that this option is exactly what you want before you use it; if in any doubt, `save` your dataset first.

`dropmiss` does not tackle all possible problems in its field. Note also that `egen, rmiss` may be useful for counting the number of missing values in each observation; see [R] **egen**.

Examples

To drop all variables with all values missing, type

```
. dropmiss
```

while to drop all observations with all values missing, type

```
. dropmiss, obs
```

With the auto dataset,

```
. dropmiss
```

drops nothing;

```
. dropmiss, any
```

drops `rep78`;

```
. dropmiss, obs
```

drops nothing; and

```
. dropmiss, obs any
```

drops 5 observations for which `rep78` is missing.

Acknowledgments

Jeroen Weesie suggested adding `if` and `in`. Estie Sid Hudes prompted the addition of an optional *varlist* when the `obs` option is specified and of the `any` option.

References

Cox, N. J. 1999. dm67: Numbers of missing and present values. *Stata Technical Bulletin* 49: 7–8. Reprinted in *Stata Technical Bulletin Reprints*, vol. 9, pp. 26–27.

——. 2001. dm67.1: Enhancements to numbers of missing and present values. *Stata Technical Bulletin* 60: 2–3.

dm90	Listing distinct values of a variable

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

Abstract: `vallist` puts a sorted list of the distinct nonmissing values of *varname* into local macro `r(list)` and displays that list. It may be useful interactively or in programming.

Keywords: `vallist`, programming, macros.

Syntax

`vallist` *varname* [`if` *exp*] [`in` *range*] [`,` `global(`*macname*`)` `labels` `quoted` [`no`]`trim` `max(`#`)` `words` `format(`*format*`)` `sep(`*string*`)` `missing`]

Description

`vallist` puts a sorted list of the distinct nonmissing values of *varname*, which may be numeric or string, into local macro `r(list)` and displays that list.

`vallist` may be used interactively, but it is most likely to be useful to programmers dealing with categorical or counted data.

Options

`global(`*macname*`)` puts the list into global macro *macname*.

`labels` specifies that labels should be used for numeric variables with value labels.

`quoted` specifies that values should be placed in quotes. This may be useful for string values or value labels containing embedded spaces.

`notrim` suppresses trimming of trailing spaces from string values.

`max(#)` specifies that at most, the first # characters of text (string values or value labels) should be used for each value. For example, `max(8)` may be needed to cut down text to elements acceptable in Stata 6 as matrix row or column names.

`words` specifies that text should be truncated to the first whole 'word', that is, just before the first space after a nonspace character. For example, a string value of `"foo bar"` would be represented by `"foo"`.

`format(`*format*`)` specifies a format for use. This is likely to be most useful with noninteger numeric values. A string format should be specified for string values or value labels and a numeric format otherwise.

`sep(`*string*`)` specifies a separator other than a space, which is the default.

The sequence of operations is `max()`, `words`, `format()`, `sep()`.

`missing` specifies that missing values should also be listed. Missing (that is, empty) values of string variables are specified as `"missing"`. Note that this description may be truncated with the `max()` option.

Discussion

Users tabulating their data, especially categorical or integer-valued variables, can see a listing of the distinct values observed, together with other information such as frequencies of occurrence. Sometimes, it is desired to see a list of values presented more concisely, or especially to put a list of those values into a local or global macro for use in a later command. In practice, this need seems to occur most when the number of distinct values is (say) between 5 and 50. With a variable such as gender which takes only a few values, it is not a burden to type out those values explicitly. While with a variable which takes on hundreds or more distinct values, it is unlikely that a long list of those values is of use or interest. `vallist` is aimed at intermediate cases.

Examples

We start with some simple examples which show how `vallist` concisely displays distinct values for categorical variables and how they are saved by default in `r(list)` and optionally in a named global macro.

```
. use auto
(1978 Automobile Data)
. vallist rep78
1 2 3 4 5
. di "`r(list)'"
1 2 3 4 5
. vallist rep78 , g(rep78v)
1 2 3 4 5
. di "$rep78v"
1 2 3 4 5
. vallist foreign, labels
Domestic Foreign
```

For more information on macros such as `r(list)` saved by r class commands such as `vallist`, see [U] **21.8 Accessing results calculated by other programs**. The main practical point for users is that `r(list)` will be overwritten by the next r class command that is issued, and so it may have only a brief existence. This is not a problem so long as `r(list)` is used immediately. To put results in a longer-lasting macro, either copy the results from `r(list)` as soon as produced, or (as here) use a global macro in addition.

Another use of `vallist` is to pick up for later use any list which it would be too tedious to enter by hand. Here a tick is placed at every distinct value as marginal decoration for a histogram:

```
. vallist mpg , g(mpgval)
12 14 15 16 17 18 19 20 21 22 23 24 25 26 28 29 30 31 34 35 41
. graph mpg, bin(14) xla(12 41) xti($mpgval) xsc(10,45) yla
```

The same trick may be used to ensure that a series of graphs is labeled informatively. An example of one standard graphics idiom is

```
. sort rep78
. by rep78 : graph mpg price
```

However, this idiom has two key disadvantages: individual graphs are not labeled informatively, and they cannot be saved to individual files. An alternative is

```
. vallist rep78
1 2 3 4 5
. for num `r(list)' : graph price mpg if rep78 == X, t1(rep78 is X) saving(rep78X)
->  graph price mpg if rep78 == 1, t1(rep78 is 1) saving(rep781)
->  graph price mpg if rep78 == 2, t1(rep78 is 2) saving(rep782)
->  graph price mpg if rep78 == 3, t1(rep78 is 3) saving(rep783)
->  graph price mpg if rep78 == 4, t1(rep78 is 4) saving(rep784)
->  graph price mpg if rep78 == 5, t1(rep78 is 5) saving(rep785)
```

In this toy example, it would have been no more work to do it the direct way

```
. for num 1 2 3 4 5 : graph price mpg if rep78 == X, t1(rep78 is X) saving(rep78X)
```

It is not difficult to imagine examples in which `vallist` offers a better route, especially when distinct values are numerous and irregular in pattern.

With string or measured variables, `vallist` still works, but it is perhaps less useful. With string variables, there are some choices over handling of string values which include spaces or which may be too long for some purpose. The `quoted`, `notrim`, `max()`, `words()`, and `sep()` options may be useful here. Note that Stata itself trims leading spaces in any operation of the form

`local` *macname* `=` *string_variable*`[`*i*`]`

Trailing spaces are also trimmed by default. With measured variables containing fractional parts, the `format` option often helps.

```
. vallist make
AMC Concord AMC Pacer AMC Spirit Audi 5000 Audi Fox BMW 320i Buick Century Buic
> k Electra Buick LeSabre Buick Opel Buick Regal Buick Riviera Buick Skylark Ca
> d. Deville Cad. Eldorado Cad. Seville Chev. Chevette Chev. Impala Chev. Malib
> u Chev. Monte Carlo Chev. Monza Chev. Nova Datsun 200 Datsun 210 Datsun 510 D
> atsun 810 Dodge Colt Dodge Diplomat Dodge Magnum Dodge St. Regis Fiat Strada
> Ford Fiesta Ford Mustang Honda Accord Honda Civic Linc. Continental Linc. Mar
> k V Linc. Versailles Mazda GLC Merc. Bobcat Merc. Cougar Merc. Marquis Merc.
> Monarch Merc. XR-7 Merc. Zephyr Olds 98 Olds Cutl Supr Olds Cutlass Olds Delt
> a 88 Olds Omega Olds Starfire Olds Toronado Peugeot 604 Plym. Arrow Plym. Cha
> mp Plym. Horizon Plym. Sapporo Plym. Volare Pont. Catalina Pont. Firebird Pon
> t. Grand Prix Pont. Le Mans Pont. Phoenix Pont. Sunbird Renault Le Car Subaru
>  Toyota Celica Toyota Corolla Toyota Corona VW Dasher VW Diesel VW Rabbit VW
> Scirocco Volvo 260
. vallist make, sep(", ")
AMC Concord, AMC Pacer, AMC Spirit, Audi 5000, Audi Fox, BMW 320i, Buick Centur
> y, Buick Electra, Buick LeSabre, Buick Opel, Buick Regal, Buick Riviera, Buic
> k Skylark, Cad. Deville, Cad. Eldorado, Cad. Seville, Chev. Chevette, Chev. I
> mpala, Chev. Malibu, Chev. Monte Carlo, Chev. Monza, Chev. Nova, Datsun 200,
> Datsun 210, Datsun 510, Datsun 810, Dodge Colt, Dodge Diplomat, Dodge Magnum,
>  Dodge St. Regis, Fiat Strada, Ford Fiesta, Ford Mustang, Honda Accord, Honda
>  Civic, Linc. Continental, Linc. Mark V, Linc. Versailles, Mazda GLC, Merc. B
> obcat, Merc. Cougar, Merc. Marquis, Merc. Monarch, Merc. XR-7, Merc. Zephyr,
> Olds 98, Olds Cutl Supr, Olds Cutlass, Olds Delta 88, Olds Omega, Olds Starfi
> re, Olds Toronado, Peugeot 604, Plym. Arrow, Plym. Champ, Plym. Horizon, Plym
> . Sapporo, Plym. Volare, Pont. Catalina, Pont. Firebird, Pont. Grand Prix, Po
> nt. Le Mans, Pont. Phoenix, Pont. Sunbird, Renault Le Car, Subaru, Toyota Cel
> ica, Toyota Corolla, Toyota Corona, VW Dasher, VW Diesel, VW Rabbit, VW Sciro
> cco, Volvo 260
. vallist make, word
AMC AMC AMC Audi Audi BMW Buick Buick Buick Buick Buick Buick Buick Cad. Cad. C
> ad. Chev. Chev. Chev. Chev. Chev. Chev. Datsun Datsun Datsun Datsun Dodge Dod
> ge Dodge Dodge Fiat Ford Ford Honda Honda Linc. Linc. Linc. Mazda Merc. Merc.
>  Merc. Merc. Merc. Merc. Olds Olds Olds Olds Olds Olds Olds Peugeot Plym. Ply
> m. Plym. Plym. Plym. Pont. Pont. Pont. Pont. Pont. Pont. Renault  Toyota Toyo
> ta Toyota VW VW VW VW Volvo
```

```
. vallist gratio
2.190000057220459 2.240000009536743 2.259999990463257 2.27999997138977 2.410000
> 085830688 2.430000066757202 2.47000002861023 2.52999997138977 2.5599999427795
> 41 2.730000019073486 2.75 2.869999885559082 2.930000066757202 2.9400000572204
> 59 2.97000002861023 2.980000019073486 3.049999952316284 3.059999942779541 3.0
> 79999923706054 3.150000095367432 3.200000047683716 3.210000038146972 3.230000
> 019073486 3.299999952316284 3.369999885559082 3.539999961853028 3.54999995231
> 6284 3.579999923706054 3.640000104904175 3.700000047683716 3.72000002861023 3
> .730000019073486 3.740000009536743 3.77999997138977 3.809999942779541 3.89000
> 0104904175
. vallist gratio , format(%3.2f)
2.19 2.24 2.26 2.28 2.41 2.43 2.47 2.53 2.56 2.73 2.75 2.87 2.93 2.94 2.97 2.98
>  3.05 3.06 3.08 3.15 3.20 3.21 3.23 3.30 3.37 3.54 3.55 3.58 3.64 3.70 3.72 3
> .73 3.74 3.78 3.81 3.89
```

Saved results

`r(list)` contains the list of distinct values.

gr46	Display a graph of nonlinear effects

Jeroen Weesie, Utrecht University, Netherlands, J.Weesie@fss.uu.nl

Abstract: After estimating a model in which the effect of some variable v was modeled via a number of functions of v (e.g., polynomials or splines in v), `graphf` displays the fitted effect of v against v, optionally with various forms of confidence bands.

Keywords: interpretation, nonlinear effects, confidence bands, bootstrap.

Introduction

Suppose you estimate a model which includes various transformations $f_1(x), f_2(x), \ldots, f_k(x)$ of a variable x, in order to assess (estimate) a nonlinear effect of variable x. The easiest way to evaluate what the estimated model asserts about the total effect $F(x) = \sum b_j f_j(x)$ of variable x is to display it in a graph of $\widehat{F}(x)$ versus x, possibly with a confidence band around $\widehat{F}(x)$. This insert describes a simple utility to do this.

Syntax

`graphf` *var* [*varlist*] [`,` `eq(`*str*`)` `cb(``none` | `value` | `envelope` | `bs [`*#*`])` `level(`*#*`)`]

Description

`graphf` is a post-estimation command that produces a graph of the fitted values and confidence band of the effect of a variable *var*, modeled via a linear predictor in (usually nonlinear) transformations of *var*. For instance, if the effect of working experience (`exp`) is modeled via a linear term `exp` and a quadratic term `exp2`, then *var* = `exp` and *varlist* = `exp exp2`. Variables in *varlist* that do not occur in the model are skipped with a warning.

If *varlist* is not specified, `graphf` will determine itself the *varlist* as the variables in the (selected equation of the) model that are a function of *var*, that is, do not vary within levels of *var*.

Options

`eq(`*str*`)` specifies the name of the equation in which the variables occur. If `eq()` is not specified, `graphf` uses the first equation of the model.

`cb(`*value*`)` specifies the type of confidence band to be displayed. Valid values are

`none`	No band is displayed (the default).
`value`	Specifies a value-wise confidence band that consists of the spline-connected lower and upper limits of the value-wise confidence intervals.
`envelope`	Ripley's envelope confidence band is displayed. This band is obtained as the value-wise minimum and maximum of the fitted values from 19 draws from the estimated multivariate distribution of the estimators for the coefficients in the model estimated last.
`bs [`*#*`]`	A parametric bootstrap confidence band, assuming multivariate normality of the parameter estimates. The value-wise confidence interval is estimated by the bias-corrected method. The number of Monte Carlo simulations # defaults to 1000.

`level(#)` specifies the confidence level, in percent, for the value-wise confidence intervals. The default is `level(95)` or as set by `set level`; see [R] **level**.

Example

We demonstrate the command using artificial data on income (`inc`) as a function of formal education (`edu`) and work experience (`exp`). The data were generated with the following code fragment:

```
. set seed 123456789
. set obs 400
. gen edu = invnorm(uniform())
. gen exp = int(30*uniform())
. label var exp "work experience"
. gen inc = edu + log(0.1+10*exp) + 1*invnorm(uniform())
```

We expect a nonlinear effect of `exp`. First, we will fit regression models with quadratic and cubic effects for `exp` via orthogonal polynomials.

```
. orthpoly exp, degree(3) gen(exp1 exp2 exp3)
. regress inc edu exp exp2
```

Source	SS	df	MS		Number of obs =	400
					F(3, 396) =	219.82
Model	1268.16207	3	422.720692		Prob > F =	0.0000
Residual	761.533804	396	1.92306516		R-squared =	0.6248
					Adj R-squared =	0.6220
Total	2029.69588	399	5.08695709		Root MSE =	1.3867

inc	Coef.	Std. Err.	t	P>|t|	[95% Conf.	Interval]
edu	1.052901	.0729641	14.43	0.000	.9094554	1.196346
exp	.1418497	.007852	18.07	0.000	.1264129	.1572866
exp2	-.6389055	.0693693	-9.21	0.000	-.7752837	-.5025274
_cons	2.395447	.1339128	17.89	0.000	2.132179	2.658716

What does the fitted quadratic effect of working experience on income look like? Remember, `exp2` is not simply the square of `exp`. An easy way to assess and interpret the nonlinear effect of `exp` on `inc` is making a plot of `exp` against the fitted effect in `exp`. This is accomplished by invoking the command `graphf exp` which will graph the effect of `exp`.

```
. graphf exp
function of exp : exp exp2
```

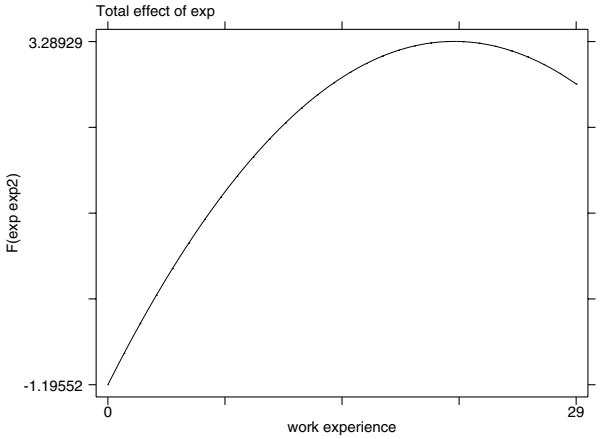

Figure 1. Fitted quadratic effect of working experience on income.

Note that you did not have to specify the variables (`exp` and `exp2`) that you used to model the effect of working experience. `graphf` selects itself the variables in the model that are a function of `exp`. In this case, `graphf` discovered that the model includes the variables `exp` and `exp2`, and thus it has plotted `exp` against .1418 `exp` − .6389 `exp2`.

According to this plot, the effect of working experience decreases for larger levels of working experience. This is hard to believe. Probably, the quadratic formulation of the experience effect is to blame here. We fit a cubic model for experience.

```
. regress inc edu exp exp2 exp3
```

Source	SS	df	MS		Number of obs = 400
					F(4, 395) = 214.18
Model	1389.18895	4	347.297237		Prob > F = 0.0000
Residual	640.506932	395	1.62153654		R-squared = 0.6844
					Adj R-squared = 0.6812
Total	2029.69588	399	5.08695709		Root MSE = 1.2734

inc	Coef.	Std. Err.	t	P>|t|	[95% Conf.	Interval]
edu	1.053465	.0670001	15.72	0.000	.9217432	1.185186
exp	.1418463	.0072102	19.67	0.000	.1276711	.1560214
exp2	-.6388893	.0636991	-10.03	0.000	-.764121	-.5136575
exp3	.5500613	.0636698	8.64	0.000	.4248873	.6752354
_cons	2.39551	.122967	19.48	0.000	2.153759	2.637262

Again, we will graph the effect of experience on income. This time, we add a confidence band for the estimated effect. Specifying `cb(value)` displays the value-wise bands, $\widehat{F}(x) \pm \Phi^{-1}(\alpha)(\mathrm{se}(\widehat{F}(x))$, where Φ is the standard normal cumulative distribution function. Note that α defaults to the value set via `set level` and can be overruled with the option `level`.

```
. graphf exp, cb(value)
function of exp : exp exp2 exp3
```

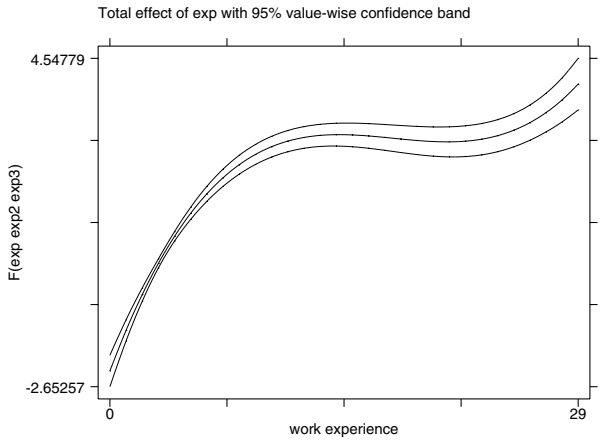

Figure 2. Cubic effect of experience on income with pointwise confidence bands.

The nonconcave estimated form of the experience effect does not make good theoretical sense. I still suspect misspecification. We next try linear splines (piecewise linear functions) as produced by the `mkspline` command,

```
. mkspline Exp 4 = exp
. regress inc edu Exp*
```

Source	SS	df	MS		Number of obs = 400
					F(5, 394) = 167.32
Model	1379.85553	5	275.971106		Prob > F = 0.0000
Residual	649.84035	394	1.64934099		R-squared = 0.6798
					Adj R-squared = 0.6758
Total	2029.69588	399	5.08695709		Root MSE = 1.2843

inc	Coef.	Std. Err.	t	P>|t|	[95% Conf.	Interval]
edu	1.065626	.0675651	15.77	0.000	.9327928	1.198459
Exp1	.6608829	.0429611	15.38	0.000	.5764212	.7453446
Exp2	-.0794035	.0392677	-2.02	0.044	-.1566039	-.002203
Exp3	.1601726	.0400491	4.00	0.000	.081436	.2389092
Exp4	-.0020411	.0444742	-0.05	0.963	-.0894775	.0853954
_cons	.279721	.206746	1.35	0.177	-.1267423	.6861844

We graph the fitted effect of experience, this time with a confidence band estimated by Ripley's envelope method. Here, the confidence band of $\widehat{F}(x) = F(x, \widehat{\beta})$ is estimated as the value-wise maximum and minimum of $F(x, \beta_i)$, where the β_i, i=1, ..., 19, are random variates generated from a multivariate normal distribution with the mean and variance of $\widehat{\beta}$ from the regression model.

```
. graphf exp, cb(envelope)
function of exp : Exp1 Exp2 Exp3 Exp4
```

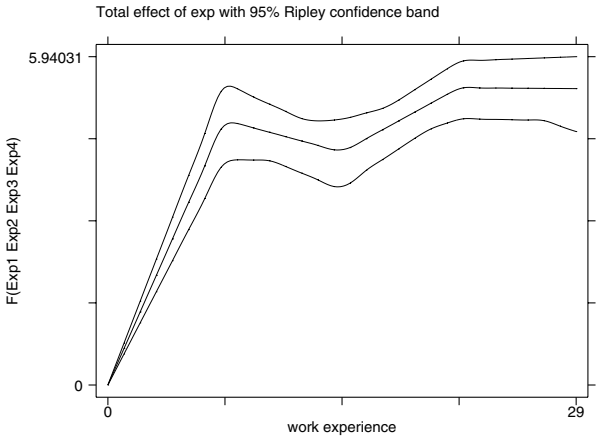

Figure 3. Cubic effect of experience on income with confidence bands using the Ripley envelope method.

While the estimated experience is not monotonic, the confidence band still allows monotonic concave effects, so I don't wonder much anymore. The estimated shape suggests a logarithmic or square-root effect of experience on income. (This could hardly be a surprise as the data were generated under this assumption.)

Finally, I estimate a quadratic spline of experience in the truncated power series form (see, for example, Section 9.3 of Seber and Wild 1989) with knots at 7.5, 15, and 22.5.

```
. drop Exp*
. gen Exp0 = (exp-15)^2
. forvalues i = 1/3 gen Exp‘i’ = max(0,exp-7.5*‘i’)^2
. regress inc edu Exp*
```

Source	SS	df	MS		
Model	1355.8444	5	271.168881	Number of obs =	400
Residual	673.851473	394	1.71028293	F(5, 394) =	158.55
				Prob > F =	0.0000
Total	2029.69588	399	5.08695709	R-squared =	0.6680
				Adj R-squared =	0.6638
				Root MSE =	1.3078

inc	Coef.	Std. Err.	t	P>|t|	[95% Conf.	Interval]
edu	1.075485	.0692212	15.54	0.000	.9393955	1.211574
Exp0	-.0230213	.0014832	-15.52	0.000	-.0259374	-.0201052
Exp1	-.0072883	.0029849	-2.44	0.015	-.0131566	-.00142
Exp2	.0486917	.0103251	4.72	0.000	.0283926	.0689908
Exp3	-.0389595	.0209896	-1.86	0.064	-.0802251	.0023062
_cons	5.629674	.1858246	30.30	0.000	5.264343	5.995006

Now, we display the graph with a confidence band estimated based on a parametric bootstrap of 1000 replications from the multivariate normal distribution also used by the envelope method (see Davison and Hinkley 1997).

(*Continued on next page*)

```
. graphf exp, cb(bs)
function of exp : Exp0 Exp1 Exp2 Exp3
```

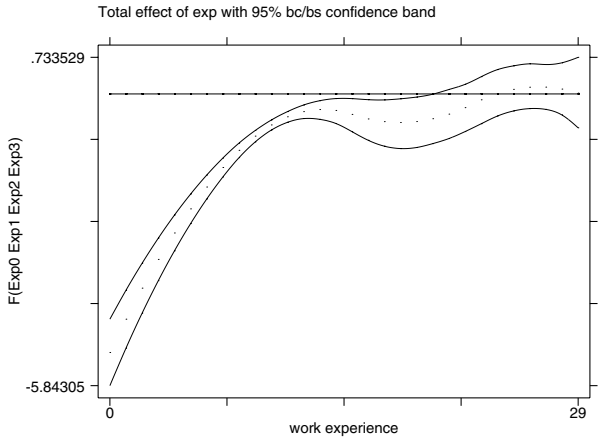

Figure 4. Cubic effect of experience on income with confidence bands from parametric bootstrap.

So far, the command **graphf** was illustrated in the simple context of a linear regression analysis. **graphf** can also be used with other estimation commands such as **logit** or **stcox**, and with multiple-equation models such as **mlogit** or **reg3** by selecting the appropriate equation via the **eq()** option.

In the examples shown here, **graphf** was able to figure out which variables in the models were the transformations of **exp**. How does **graphf** perform this "magic"? Remember, in mathematical terms a set (x, y) can be interpreted as a function if "for each x there is only one y". This is easy to verify with variables as well. This method is, of course, not foolproof, especially if your data comprise continuous (as opposed to categorical) variables. In such a case, **graphf** may include too many variables, and you may overrule **graphf** by specifying the *varlist* yourself, as in

```
. graphf exp Exp0 Exp1 Exp2 Exp3
 (output omitted)
```

In this case, **graphf** will test whether the variables **Exp0**, **Exp1**, **Exp2**, and **Exp3** are constant within levels of **exp**.

References

Davison, A.C. and D.V. Hinkley. 1997. *Bootstrap Methods and their Applications*. Cambridge: Cambridge University Press.

Seber, G. A. F. and C. J. Wild. 1989. *Nonlinear Regression*. New York: John Wiley & Sons.

sg160	On boundary-value likelihood-ratio tests

Roberto G. Gutierrez, Stata Corporation, rgutierrez@stata.com
Shana Carter, Stata Corporation, scarter@stata.com
David M. Drukker, Stata Corporation, ddrukker@stata.com

Abstract: Several Stata commands perform likelihood-ratio tests for the presence of overdispersion or random effects. By their very nature, these tests are boundary tests in that they test whether a positive-valued parameter is strictly greater than zero. In such cases, having the null space on the boundary makes standard likelihood ratio theory invalid and special consideration must be given to ensure that correct p-values are obtained. In particular, the correct p-values are shown to be one half of what was previously thought. This insert describes the problem in general terms and outlines the modifications to the affected commands that have resulted under Stata 7.

Keywords: likelihood-ratio test, chi-square, boundary test, overdispersion, random effects, variance components, frailty models.

Implicit boundary tests in Stata

Consider as a motivating example the Stata command **nbreg**, which fits the gamma–Poisson mixture

$$f(y_i|\nu_i) = \frac{(\nu_i\mu_i)^{y_i}\, e^{-\nu_i\mu_i}}{\Gamma(y_i+1)}$$

where $\mu_i = \exp(\mathbf{x}_i\beta + \text{offset}_i)$ and ν_i is an unobserved parameter with a $\text{gamma}(1/\alpha, 1/\alpha)$ density. Since ν has a mean of 1 and a variance of α, as the overdispersion α goes to zero, the model reduces to Poisson. A natural comparison of the two models is the likelihood-ratio test of $H_o : \alpha = 0$ versus $H_1 : \alpha > 0$ with $\alpha = 0$ being on the boundary of the parameter space.

In the context of panel data (the **xt** family of commands), boundary tests arise from random effects models where the variance component is estimated via maximum likelihood. For example, **xtreg** fits models of the form

$$y_{it} = \alpha + \mathbf{x}_{it}\beta + \nu_i + \epsilon_{it}$$

where the ν_i's are random effects. The likelihood-ratio test of $H_o : \sigma_\nu^2 = 0$ versus the right-sided alternative is a boundary test. The other affected **xt** commands perform an analogous boundary test for the presence of an overall group random effect.

In both the above situations, a critical regularity condition is violated. The null parameter space is no longer interior to the full parameter space, and thus the usual result which states that the likelihood-ratio test statistic tends towards χ^2_1 in distribution is untrue. As a result, the p-values given by these estimation routines pre-Stata 7 are incorrect.

In addition, with the release of Stata 7, **streg** now allows a **frailty()** option to fit parametric survival models with overdispersion. Therefore, the hazard function given a latent multiplicative random effect is

$$h(t|\alpha_i) = \alpha_i h(t)$$

where $h(t)$ is any of the parametric hazard functions already estimated by **streg**, and α_i is random with a mean of 1 and a variance of θ and is distributed as either gamma (**frailty(gamma)**) or inverse-Gaussian (**frailty(invgaussian)**). In this case, the likelihood-ratio test of $H_o : \theta = 0$ versus $H_1 : \theta > 0$ also falls on the boundary of the parameter space, and thus requires the same special consideration as **nbreg** and as the **xt** commands.

Limiting distribution of the likelihood-ratio test statistic

Examinations of the large-sample properties of likelihood-ratio tests under nonstandard conditions can be found as early as Moran (1970) and Chant (1974), with Self and Liang (1987) giving more general and rigorous consideration. Miller (1977) also studies this problem specifically when testing for the presence of a single random effect.

For the purpose of deriving limiting distributions, Self and Liang (1987) distinguish likelihood-ratio tests according to the number of parameters that fall into the four categories generated by whether the parameter is of primary interest and by whether the parameter value is on the boundary under the null.

The implicit likelihood-ratio tests under our consideration are mutually common in the sense that they all consist of exactly one boundary parameter that is of primary interest, with all other parameters being nuisance parameters that are interior under the null. This corresponds to Case 5 of Self and Liang (1987) in which case the limiting distribution of the likelihood-ratio test statistic is shown to be a 50:50 mixture of the χ^2_0 (point mass at zero) and χ^2_1 distributions. This mixture is denoted as $\bar{\chi}^2_{01}$. Thus, any upper-tail area that is calculated with respect to this limiting distribution will be equal to one-half times the corresponding area calculated with respect to the χ^2_1 distribution.

Updates

With the release of Stata 7, the commands **nbreg**, **xtclog**, **xtintreg**, **xtlogit**, **xtnbreg**, **xtpois**, **xtprobit**, **xtreg**, **xttobit**, and **zinb** have been updated so that the p-values of the boundary tests are now based on a 50:50 mixture of the χ^2_0 and χ^2_1 distributions, i.e., the $\bar{\chi}^2_{01}$ distribution. This distribution was also utilized when the **frailty()** option was added to **streg**.

Example

Consider the cancer data which ships with Stata 7 to which we fit a Gompertz regression model with gamma heterogeneity.

```
. use /usr/local/stata7/cancer
(Patient Survival in Drug Trial)
. stset studytime died
     failure event:  died ~= 0 & died ~= .
obs. time interval:  (0, studytime]
 exit on or before:  failure
------------------------------------------------------------------------------
       48  total obs.
        0  exclusions
------------------------------------------------------------------------------
       48  obs. remaining, representing
       31  failures in single record/single failure data
      744  total analysis time at risk, at risk from t =         0
                            earliest observed entry t =         0
                                 last observed exit t =        39
```

```
. streg drug age, dist(gompertz) frailty(gamma) nolog
        failure _d:  died
  analysis time _t:  studytime

Gompertz regression -- log relative-hazard form
                    Gamma frailty

No. of subjects =           48                     Number of obs   =        48
No. of failures =           31
Time at risk    =          744
                                                   LR chi2(2)      =     37.20
Log likelihood  =   -42.044123                     Prob > chi2     =    0.0000

------------------------------------------------------------------------------
          _t | Haz. Ratio   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
        drug |   .1827866   .1011623    -3.07   0.002     .0617805    .5408011
         age |   1.129536   .0496724     2.77   0.006     1.036257    1.231211
-------------+----------------------------------------------------------------
       gamma |   .1129468   .0507817     2.22   0.026     .0134166    .2124771
    /ln_the  |  -1.628188   2.458232    -0.66   0.508    -6.446234    3.189859
-------------+----------------------------------------------------------------
       theta |    .196285   .4825141                      .0015865    24.28501
------------------------------------------------------------------------------
Likelihood ratio test of theta=0: chibar2(01) =     0.19 Prob>=chibar2 = 0.330
```

The test statistic for the likelihood-ratio test of $H_o : \theta = 0$ is labeled as `chibar2(01)`, and the level of significance is obtained as $\Pr(\chi^2_1 > 0.19)/2$. In situations where the estimated θ is so small as to be indistinguishable from zero, the test statistic is set to zero to reflect this. In that case, a p-value of $\Pr(\chi^2_1 > 0)/2 = 0.5$ would be misleading, and thus a p-value of 1 is reported instead. For this reason, the level of significance is labeled as `Prob>=chibar2` instead of being a strict inequality.

Simulation

Consider the simulation run below which empirically examines the distribution of the likelihood-ratio test statistic present in `xtreg, mle`.

```
. capture program drop boundary
. program define boundary
  1.         version 7
  2.         if "`1'" == "?" {
  3.                 global S_1 "chi"
  4.                 exit
  5.         }
  6.         drop _all
  7.         set obs 5000
  8.         gen x1 = -1 + 2*uniform()
  9.         gen x2 = -1 + 2*uniform()
 10.         gen xb = -1 + 0.5*x1 + 1.5*x2
 11.         gen y = xb + invnorm(uniform())
 12.         gen u = int(_n/5)
 13.         xtreg y x1 x2, i(u) mle
 14.         post `1' (e(chi2_c))
 15. end
. set seed 10001
. simul boundary, reps(1000)
. sort chi
. summ
    Variable |       Obs        Mean    Std. Dev.       Min        Max
-------------+--------------------------------------------------------
         chi |      1000    .4819788    1.105139          0    14.0836
. count if chi==0
  496
. summ if _n>r(N)
    Variable |       Obs        Mean    Std. Dev.       Min        Max
-------------+--------------------------------------------------------
         chi |       504    .9563071    1.403981   .0000406    14.0836
```

For each of 1,000 replications, a dataset is generated with arbitrary panels, and a linear regression model with random effects is fit using maximum likelihood. The test statistic, which tests the presence of a random effect, is then recorded for each replication, and the results of the simulation are summarized.

The summary of the simulation results show that the empirical distribution of the test statistic is indeed consistent with that of a $\bar{\chi}^2_{01}$. Note that roughly half of the time the test statistic is zero, and when not zero has a mean and a standard deviation consistent with that of a χ^2_1 random variable. In addition, a χ^2 quantile-quantile plot of the nonzero test statistics is given below.

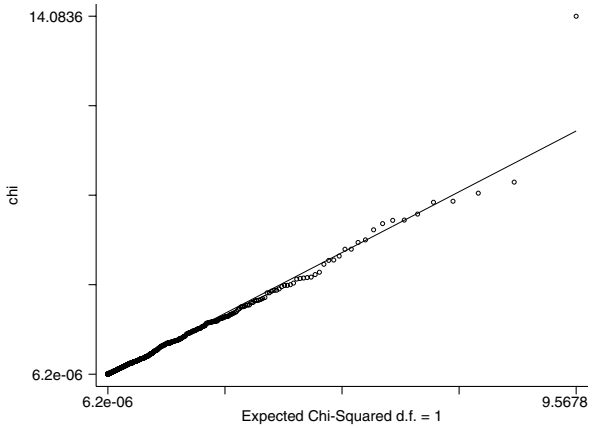

Figure 1. χ^2 quantile-quantile plot of nonzero values of `chi`.

Acknowledgment

The authors wish to thank Jeroen Weesie, Utrecht University, Netherlands, for bringing this matter to our attention.

References

Chant, D. 1974. On asymptotic tests of composite hypotheses in nonstandard conditions. *Biometrika* 61: 291–298.

Miller, J. J. 1977. Asymptotic properties of maximum likelihood estimates in the mixed model of the analysis of variance. *Annals of Statistics* 5: 746–762.

Moran, P. A. P. 1970. Maximum-likelihood estimation in nonstandard conditions. *Proceedings of the Cambridge Philosophical Society* 70: 441–450.

Self, S. G. and K.-Y. Liang. 1987. Asymptotic properties of maximum likelihood estimators and likelihood-ratio tests under nonstandard conditions. *Journal of the American Statistical Association* 82: 605–610.

sg161	Analysis of the turning point of a quadratic specification

Jeroen Weesie, Utrecht University, Netherlands, J.Weesie@fss.uu.nl

Abstract: `wherext` assists in interpreting results from a model with a quadratic specification.

Keywords: interpretation, quadratic, bootstrap.

Introduction

Suppose you estimated some model (`regress`, `logit`, `heckman`, etc.) with a quadratic specification for some variable, that is, you include both a linear and quadratic term. `wherext` helps interpret the results. For a related command on more general nonlinear specifications, please refer to `graphf` given in Weesie (2001). `wherext` displays the range of the variable, the value *argext* at which the combination of the linear and quadratic terms are extremal ("turning point"), and provides a confidence interval for *argext* based on the delta method or on a parametric bootstrap. Depending on the location of *argext* relative to the range of the variable, you can now easily determine whether the total effect of a variable is always positive (negative) or whether it changes sign at a meaningful value.

Syntax

`wherext` *linear-var quadratic-var* [`, eq(`*str*`) bootstrap reps(`*#*`) kdensity`[`(`*options*`)`] `level(`*#*`)`]

Description

`wherext` is a post-estimation command that helps interpret the results from estimating a model that includes a variable v and a quadratic term in v. `wherext` displays the range of v, the value of v (called *argext*) at which the linear plus quadratic terms are extreme (turning point), and a standard error and confidence interval of *argext*.

The quadratic term in v may take the general form $a + bv + cv^2$; `wherext` verifies that it is indeed quadratic.

The standard error and confidence interval are computed via the delta method, and optionally by Monte Carlo simulation.

Options

`eq(`*str*`)` specifies the name of the equation in which the variables occur. If not specified, the first equation is assumed.

`bootstrap` specifies that a simulation estimate of the confidence interval of *argext* is computed. The simulator assumes that the coefficients of the linear and quadratic terms are distributed as bivariate normal with mean and variance obtained from the estimation command.

`reps(`*#*`)` specifies the number of Monte Carlo simulations to be performed. The default is 10,000.

`kdensity`[`(`*options*`)`] specifies that a kernel density estimate of the distribution of *argext* is displayed. The graph is overlaid with a normal distribution based on the delta method. Options for the `kdensity` command can be provided as an argument to the `kdensity` option of `wherext`, without the comma. For example, `kdensity(parzen)` specifies that option `parzen` is issued when invoking `kdensity`.

`level(`*#*`)` specifies the confidence level, in percent, for confidence intervals. The default is `level(95)` or as set by `set level`.

Example

We demonstrate `wherext` using artificial data on income (`inc`) as a function of formal education (`edu`) and work experience (`exp`). We expect a nonlinear effect of `exp`. We first fit a regression model:

```
. regress inc edu exp exp2
      Source |       SS       df       MS              Number of obs =     400
-------------+------------------------------           F(  3,   396) =  210.72
       Model |  10638.797      3  3546.26567            Prob > F      =  0.0000
    Residual |  6664.43908   396  16.8293916            R-squared     =  0.6148
-------------+------------------------------           Adj R-squared =  0.6119
       Total |  17303.2361   399  43.3665065            Root MSE      =  4.1024

------------------------------------------------------------------------------
         inc |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
         edu |    1.161939   .2158516     5.38   0.000     .7375808    1.586298
         exp |     .861577   .0858501    10.04   0.000     .6927981    1.030356
        exp2 |   -.0113556   .0027453    -4.14   0.000    -.0167528   -.0059584
       _cons |     1.68029   .5508591     3.05   0.002     .5973158    2.763264
------------------------------------------------------------------------------
```

To obtain information about the turning point of the experience effect, we invoke `wherext` with the linear and quadratic terms.

```
. wherext exp exp2
range of exp                                  = [0,31]
exp+exp2 has maximum in argext                = 37.93618
Std Error of argext (delta method)            = 5.597306
95% confidence interval for argext            = (26.96566,  48.9067)
```

Apparently, experience has a positive effect on income, but the effect decreases with increasing levels of experience. The point at which extra experience affects income negatively is estimated to be 37.9 years. `wherext` refers to this point as the turning point. This is well beyond the range of experiences over which we have data; the maximal labor market experience in the data is 31 years. Thus, we would probably like to treat the possible negative effect of experience as an artificial result of extrapolating the model. Whether we can be sure depends on how sure we can be about the location of the turning point. Using the delta method (that is, linearly approximating the nonlinear function of the turning point in the regression coefficients), `wherext` computed the standard error of *argext* as 5.6. A symmetric confidence interval for *argext* is formed based on the assumption that *argext* is normally distributed. In this case, the confidence interval for *argext* clearly overlaps with the relevant range of experience. It looks like we may well have to face the possibility that experience has a negative effect for high levels of experience.

It is hard to make sense of this theoretically. Before we try, let's take a closer look at the methods used. The turning point *argext* $=$ `_b[exp2]`$/2$`_b[exp]` depends quite nonlinearly on the regression coefficient, especially if `b[exp2]` is small. We may indeed cast doubt on the quality of the normal approximation to the distribution of *argext*, even if we have reasonable confidence in the adequacy of the normal approximation of the distribution of the regression coefficients. To assess the distribution of *argext*, we can turn to a parametric bootstrap that assumes that the coefficients `_b[exp]` and `_b[exp2]` are indeed distributed as bivariate normal (see Davison and Hinkley 1997; see also King et al. 2000).

```
. wherext exp exp2, boot rep(1000) kdensity seed(123)
range of exp                                    = [0,31]
exp+exp2 has maximum in argext                  = 37.93618
Std Error of argext (delta method)              = 5.597306
95% confidence interval for argext              = (26.96566,48.9067)
Parametric bootstrap statistics (assuming bivariate normality)
```

Variable	Reps	Observed	Bias	Std. Err.	[95% Conf.	Interval]	
argext	1000	37.93618	1.845084	11.201	15.956	59.91636	(N)
					30.33916	57.58017	(P)
					30.20184	56.46141	(BC)

N = normal, P = percentile, BC = bias-corrected
seed used 123

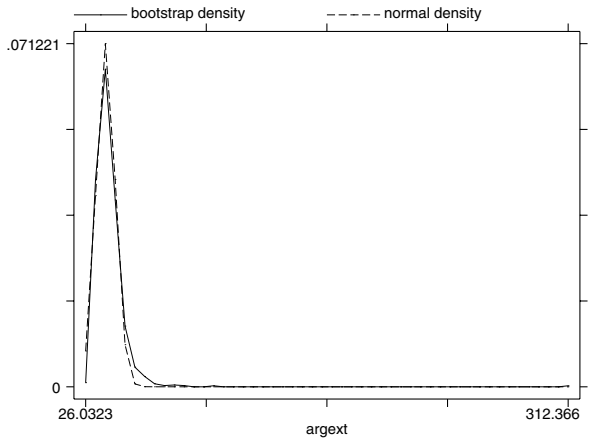

Figure 1. Bootstrap distribution of *argext*.

`wherext` displays a kernel density estimate of the distribution of *argext*, overlaid with the normal distribution obtained from the delta method. (Note: this is not the normal distribution that `kdensity` uses.) Figure 1 leads us to question the adequacy of the symmetric confidence intervals formed under the assumption that *argext* is normally distributed. The percentile and bias-corrected estimates of the confidence interval of *argext* are strongly asymmetric. According to these confidence intervals, the turning point of experience likely does not occur in the observed range of experience, and hence the theoretical puzzle does not need to be addressed.

Some readers may now be worried about the adequacy of the assumption that the regression coefficients are normally distributed. We can make a further step and perform a nonparametric bootstrap based on resampling observations. Such a bootstrap takes a considerable amount of time, since it has to estimate the models many times over. The parametric bootstrap, in contrast, only has to draw from a bivariate distribution and compute the turning point for each draw. The latter is quite fast.

```
. bs "regress inc edu exp exp2" "-0.5*_b[exp]/_b[exp2]", rep(1000)
command:     regress inc edu exp exp2
statistic:   -0.5*_b[exp]/_b[exp2]
(obs=400)
Bootstrap statistics
```

Variable	Reps	Observed	Bias	Std. Err.	[95% Conf.	Interval]	
bs1	1000	37.93618	1.891745	8.962951	20.34781	55.52455	(N)
					30.68794	60.43152	(P)
					30.14097	58.88775	(BC)

N = normal, P = percentile, BC = bias-corrected

We see that there is a good similarity between the parametric and nonparametric bootstrap results. This is often the case with smooth models from the class of generalized linear models, but not with "less linear" models, such as sample selection models. Thus, in a future version of `wherext`, an option to perform the nonparametric bootstrap automatically may be included.

References

Davison, A.C. and D.V. Hinkley. 1997. *Bootstrap Methods and their Applications*. Cambridge: Cambridge University Press.

King, G., M. Tomz, and J. Wittenberg. 2000. Making the most of statistical analyses: improving interpretation and presentation. *American Journal of Political Science* 44: 341–355.

Weesie, J. 2001. gr46: Display a graph of nonlinear effects. *Stata Technical Bulletin* 60: 11–15.

sg162	Tools for spatial data analysis

Maurizio Pisati, University of Milano Bicocca, Italy, maurizio.pisati@galactica.it

Abstract: A suite of commands for spatial data analysis is presented and illustrated.

Keywords: exploratory spatial data analysis, hotspots, Moran scatterplot, spatial autocorrelation, spatial clusters, spatial correlogram, spatial regression, spatial weights matrix, test for spatial dependence.

Syntax

```
spatwmat [using filename] , name(weights_matrix) [ drop(numlist) xcoord(varname) ycoord(varname)
    band(numlist) friction(#) binary standardize eigenval(eigen_matrix) ]

spatgsa varlist , weights(matrix) [ moran geary go twotail ]

spatcorr varname , bands(numlist) xcoord(varname) ycoord(varname) [ geary cumulative
    twotail graph needle savegraph(filename [, replace]) ]

spatlsa varname , weights(matrix) [ moran geary go1 go2 id(varname) twotail sort
    graph(moran | go1 | go2) symbol(id | n) map(filename) xcoord(varname) ycoord(varname)
    savegraph(filename [, replace]) ]

spatdiag , weights(matrix)

spatreg depvar [indepvars] , weights(weights_matrix) eigenval(eigen_matrix) model(lag | error)
    [ nolog robust level(#) ]
```

Description

`spatwmat` imports or generates the spatial weights matrices required by `spatgsa`, `spatlsa`, `spatdiag`, and `spatreg`. As an option, `spatwmat` also generates the eigenvalues matrix required by `spatreg`.

`spatgsa` computes three global spatial autocorrelation statistics: Moran's I, Geary's c, and Getis and Ord's G. For each requested statistic and each variable in *varlist*, `spatgsa` computes and displays in tabular form the statistic itself, the expected value of the statistic under the null hypothesis of global spatial independence, the standard deviation of the statistic, the z-value, and the corresponding one- or two-tailed p-value.

`spatcorr` computes and optionally plots Moran's I or Geary's c spatial correlogram based on two or more consecutive or cumulative distance bands. For each distance band, `spatcorr` computes and displays in tabular form the requested statistic, the expected value of the statistic under the null hypothesis of global spatial independence, the standard deviation of the statistic, the z-value, and the corresponding one- or two-tailed p-value.

`spatlsa` computes four local spatial autocorrelation statistics: Moran's I_i, Geary's c_i, Getis and Ord's G_i, and Getis and Ord's G_i^*. For each requested statistic and each location in the analysis, `spatlsa` computes and displays in tabular form the statistic itself, the expected value of the statistic under the null hypothesis of local spatial independence, the standard deviation of the statistic, the z-value, and the corresponding one- or two-tailed p-value. As an option, `spatlsa` also displays a Moran scatterplot, a map of Moran scatterplot values, a map of G_i z-values, or a map of G_i^* z-values.

`spatdiag` carries out several diagnostic tests for spatial dependence in OLS regression. For each test, `spatdiag` computes and displays in tabular form the relevant statistic, the degrees of freedom of the test, and the corresponding p-value. `spatdiag` can be used only after `regress`.

`spatreg` estimates the spatial lag and the spatial error regression models by maximum likelihood.

Options for spatwmat

`using` *filename* requests that the matrix *weights_matrix* be imported from *filename*, an external Stata datafile which includes exactly N cases and N variables containing the user-defined spatial weights (missing values are not allowed).

`name(`*weights_matrix*`)` is required. It specifies the name of the $N \times N$ spatial weights matrix to be generated, where N denotes the number of locations in the analysis.

`drop(`*numlist*`)`, if used with option `using` *filename*, specifies the cases (rows) and variables (columns) to be dropped before generating matrix *weights_matrix*.

`xcoord(`*varname*`)` is required if option `using` *filename* is not specified. It specifies the name of the variable containing the x-coordinate of each location in the analysis.

`ycoord(`*varname*`)` is required if option `using` *filename* is not specified. It specifies the name of the variable containing the y-coordinate of each location in the analysis. Both the x-coordinate and the y-coordinate must be expressed in projected units, for example, meters, kilometers, miles, or arbitrary digitizing units.

`band(`*numlist*`)` is required if option `using` *filename* is not specified. It specifies the lower and upper bounds of the distance band within which location pairs must be considered neighbors, that is, spatially contiguous, and, therefore, assigned a nonzero spatial weight. By default, `spatwmat` generates spatial weights as

$$w_{ij} = \begin{cases} 0, & \text{if } d_{ij} \leq lb \text{ or } d_{ij} > ub \\ 1/d_{ij}^{f}, & \text{if } lb < d_{ij} \leq ub \end{cases}$$

where (i, j) denotes the location pair, d_{ij} denotes the Euclidean distance between locations i and j, lb denotes the lower bound of the specified distance band, ub denotes the upper bound of the specified distance band, and f denotes a positive friction parameter (by default $f = 1$).

`friction(#)` specifies the friction parameter to be used in the computation of spatial weights (see above).

`binary` requests that a binary weights matrix be generated. To this aim, all nonzero spatial weights are set to 1.

`standardize` requests that a row-standardized weights matrix be generated. To this aim, all nonzero spatial weights are rescaled so that within each row their sum equals 1.

`eigenval(`*eigen_matrix*`)` requests that an additional $N \times 1$ matrix be generated containing the eigenvalues of *weights_matrix*. This matrix is required by `spatreg`.

Options for spatgsa

`weights(`*matrix*`)` is required. It specifies the name of the spatial weights matrix to be used in the computation of the requested global spatial autocorrelation statistics. This matrix must have been generated by `spatwmat`.

`moran` requests that Moran's I and the related quantities of interest be computed and displayed.

`geary` requests that Geary's c and the related quantities of interest be computed and displayed.

`go` requests that Getis and Ord's G and the related quantities of interest be computed and displayed. This option requires that the spatial weights matrix specified by the `weights` option be a nonstandardized symmetric binary weights matrix.

`twotail` requests that two-tailed p-values be computed and displayed instead of the default one-tailed p-values.

To run `spatgsa`, it is necessary to specify at least one of the `moran`, `geary`, and `go` options.

Options for spatcorr

`bands(`*numlist*`)` is required. It specifies the sequence of lower and upper bounds of the distance bands to be used in the calculation of the spatial correlogram. For example, if `bands(0 2 4 6)` is specified, and `cumulative` is not specified, then the spatial correlogram will be computed on the consecutive distance bands: $(0, 2]$, $(2, 4]$, $(4, 6]$. On the other hand, if option `cumulative` is specified, then the spatial correlogram will be computed on the cumulative distance bands: $(0, 2]$, $(0, 4]$, $(0, 6]$.

`xcoord(`*varname*`)` is required. It specifies the name of the variable containing the x-coordinate of each location in the analysis.

`ycoord(`*varname*`)` is required. It specifies the name of the variable containing the y-coordinate of each location in the analysis. Both the x-coordinate and the y-coordinate must be expressed in projected units, for example, meters, kilometers, miles, or arbitrary digitizing units.

`geary` requests that Geary's c spatial correlogram be computed and optionally plotted instead of the default Moran's I spatial correlogram.

`cumulative` requests that cumulative distance bands be used instead of the default consecutive distance bands.

`twotail` requests that two-tailed p-values be computed and displayed instead of the default one-tailed p-values.

`graph` requests that the spatial correlogram be plotted.

`needle` requests that the spatial correlogram be plotted using vertical lines from zero (for Moran's I) or one (for Geary's c) to the computed statistics instead of lines that connect the computed statistics.

`savegraph(`*filename* `[, replace])` requests that the graph be saved in *filename*. If *filename* exists, an error will occur unless `replace` is also specified. *filename* must have one of the extensions `.ps`, `.eps`, `.prn`, or `.wmf`.

Options for spatlsa

`weights(`*matrix*`)` is required. It specifies the name of the spatial weights matrix to be used in the computation of the requested local spatial autocorrelation statistics. This matrix must have been generated by `spatwmat`.

`moran` requests that Moran's I_i and the related quantities of interest be computed and displayed.

`geary` requests that Geary's c_i and the related quantities of interest be computed and displayed.

`go1` requests that Getis and Ord's G_i and the related quantities of interest be computed and displayed. This option requires that the spatial weights matrix specified by the `weights` option be a nonstandardized symmetric binary weights matrix.

`go2` requests that Getis and Ord's G_i^* and the related quantities of interest be computed and displayed. This option requires that the spatial weights matrix specified by the `weights` option be a nonstandardized symmetric binary weights matrix.

`id(`*varname*`)` specifies the name of the variable containing the identifier of each location in the analysis.

`twotail` requests that two-tailed p-values be computed and displayed instead of the default one-tailed p-values.

`sort` requests that statistics be displayed in ascending order based on z-values instead of the default case order.

`graph(moran | go1 | go2)` requests that a graphical representation of results be displayed.

`graph(moran)`, without option `map(`*filename*`)`, requests that a Moran scatterplot be displayed. This option requires that the spatial weights matrix specified by the `weights` option be a row-standardized weights matrix.

`graph(moran)`, with option `map(`*filename*`)`, requests that a map of Moran scatterplot values be displayed. This option requires that the spatial weights matrix specified by the `weights` option be a row-standardized weights matrix.

`graph(go1)` requests that a map of G_i z-values be displayed. This option requires that the `map` option also be specified.

`graph(go2)` requests that a map of G_i^* z-values be displayed. This option requires that the `map` option also be specified.

`symbol(id | n)` requests that in the Moran scatterplot locations be represented by an identifier instead of the default graphic symbol.

`symbol(id)` requests that locations be identified by the values of the variable specified by option `id(`*varname*`)`.

`symbol(n)` requests that locations be identified by the case number.

`map(`*filename*`)` specifies the file containing the coordinates of the polygon(s) that delineate the geographical area of analysis. (For a description of the file format, see below.) This option requires that options `xcoord(`*varname*`)` and `ycoord(`*varname*`)` be specified.

`xcoord(`*varname*`)` specifies the name of the variable containing the x-coordinate of each location in the analysis.

`ycoord(`*varname*`)` specifies the name of the variable containing the y-coordinate of each location in the analysis. Both the x-coordinate and the y-coordinate must be expressed in projected units, for example, meters, kilometers, miles, or arbitrary digitizing units.

`savegraph(`*filename* `[, replace])` requests that the graph be saved in *filename*. If *filename* exists, an error will occur unless `replace` is also specified. *filename* must have one of the extensions `.ps`, `.eps`, `.prn`, or `.wmf`.

To run `spatlsa`, it is necessary to specify at least one of the `moran`, `geary`, `go1`, or `go2` options.

Options for spatdiag

`weights(`*matrix*`)` is required. It specifies the name of the spatial weights matrix to be used in the computation of the diagnostics. This matrix must have been generated by `spatwmat`.

Options for spatreg

`weights(`*matrix*`)` is required. It specifies the name of the spatial weights matrix to be used in the estimation of the requested spatial regression model. This matrix must have been generated by `spatwmat`.

`eigenval(`*eigen_matrix*`)` is required. It specifies the name of the eigenvalues matrix to be used in the estimation of the requested spatial regression model. This matrix must have been generated by `spatwmat`.

`model(lag | error)` is required. It specifies the type of spatial regression model to be estimated.

`model(lag)` requests that the spatial lag model be estimated.

`model(error)` requests that the spatial error model be estimated.

`nolog` requests that reporting of the iteration log be suppressed.

`robust` requests that the Huber/White/sandwich estimator of variance be used instead of the traditional calculation.

`level(#)` specifies the confidence level, in percent, for confidence intervals. The default is `level(95)` or as set by `set level`.

Using spatwmat

One of the main distinguishing features of spatial data analysis is that it takes into account the spatial arrangement of the observational units, which we will call locations (Anselin 1992a). This spatial arrangement is represented by a spatial weights matrix W whose elements w_{ij} express the presence or absence (binary weights matrix) or the degree (nonbinary weights matrix) of potential spatial interaction between each possible pair of locations.

The primary purpose of `spatwmat` is to generate the $N \times N$ spatial weights matrices required by `spatgsa`, `spatlsa`, `spatdiag`, and `spatreg`, where N denotes the number of locations in the analysis. `spatwmat` can accomplish this task either by importing a user-defined matrix stored in an external Stata datafile, or by using the spatial coordinates of locations supplied by the user. As an option, `spatwmat` also generates the $N \times 1$ eigenvalues matrix required by `spatreg`.

To illustrate the practical application of `spatwmat`, as well as the other commands discussed in this insert, I will use data on 49 contiguous planning neighborhoods in Columbus, Ohio (Anselin 1988).

```
. use columbusdata.dta
. describe
Contains data from columbusdata.dta
  obs:            49
 vars:             6                          30 Jan 2001 14:44
 size:         1,225 (100.0% of memory free)
```

variable name	storage type	display format	value label	variable label
id	byte	%8.0g		Neighborhood id value
hoval	float	%9.0g		Housing value (in $1,000)
income	float	%9.0g		Household income (in $1,000)
crime	float	%9.0g		Residential burglaries & vehicle thefts per 1,000 households
x	float	%9.0g		x coordinate of centroid (in arbitrary digitizing units)
y	float	%9.0g		y coordinate of centroid (in arbitrary digitizing units)

```
Sorted by:  id
```

Before using these data to carry out the analyses of interest, we have to generate the proper spatial weights matrix. As mentioned above, a way to accomplish this task consists of importing a user-defined matrix stored in an external Stata datafile.

```
. spatwmat using columbusswm.dta, name(W)
The following matrix has been created:
1. Imported binary weights matrix W
   Dimension: 49x49
```

As we can see, when everything is finished, `spatwmat` informs us that a 49×49 spatial weights matrix named `W` has been created; moreover, `spatwmat` has detected that the imported matrix is made up of binary weights. In case we would like to analyze only the first ten neighborhoods, we could use option `drop(`*numlist*`)`.

```
. spatwmat using columbusswm.dta, name(W) drop(11/49)
The following matrix has been created:
1. Imported binary weights matrix W
   Dimension: 10x10
. matrix list W, nonames
```

```
symmetric W[10,10]
  0
  1  0
  0  1  0
  0  0  1  0
  1  0  0  0  0
  1  1  0  0  1  0
  0  1  1  0  0  1  0
  0  0  0  0  0  0  1  0
  0  0  0  0  0  0  1  0  0
  0  0  0  0  0  0  0  0  0  0
```

If we specify the `standardize` option, we obtain a row-standardized spatial weights matrix.

```
. spatwmat using columbusswm.dta, name(W) drop(11/49) standardize
The following matrix has been created:
1. Imported binary weights matrix W (row-standardized)
   Dimension: 10x10
. matrix list W, nonames format(%4.2f)
W[10,10]
  0.00  0.33  0.00  0.00  0.33  0.33  0.00  0.00  0.00  0.00
  0.25  0.00  0.25  0.00  0.00  0.25  0.25  0.00  0.00  0.00
  0.00  0.33  0.00  0.33  0.00  0.00  0.33  0.00  0.00  0.00
  0.00  0.00  1.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00
  0.50  0.00  0.00  0.00  0.00  0.50  0.00  0.00  0.00  0.00
  0.25  0.25  0.00  0.00  0.25  0.00  0.25  0.00  0.00  0.00
  0.00  0.20  0.20  0.00  0.00  0.20  0.00  0.20  0.20  0.00
  0.00  0.00  0.00  0.00  0.00  0.00  1.00  0.00  0.00  0.00
  0.00  0.00  0.00  0.00  0.00  0.00  1.00  0.00  0.00  0.00
  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00
```

Alternatively, `spatwmat` can be used to generate several kinds of distance-based spatial weights matrices. For example,

```
. spatwmat, name(W) xcoord(x) ycoord(y) band(0 3) binary
The following matrix has been created:
1. Distance-based binary weights matrix W
   Dimension: 49x49
   Distance band: 0 < d <= 3
   Friction parameter: 1
   Minimum distance: 0.1
   1st quartile distance: 1.0
   Median distance: 1.7
   3rd quartile distance: 2.4
   Maximum distance: 4.8
   Largest minimum distance: 0.67
   Smallest maximum distance: 2.53
```

As we can see from the output, the above command has generated a 49×49 binary weights matrix named `W` whose elements take value 1 if locations are within a distance of 3 digitizing units from each other, and value 0 otherwise. The output also offers some additional information that may help specify the proper distance band. For instance, the value corresponding to the largest minimum distance suggests that if we specify an upper bound smaller than 0.67, there will be at least one location with no neighbors:

```
. spatwmat, name(W) xcoord(x) ycoord(y) band(0 0.66) binary
The following matrix has been created:
1. Distance-based binary weights matrix W
   Dimension: 49x49
   Distance band: 0 < d <= .66
   Friction parameter: 1
   Minimum distance: 0.1
   1st quartile distance: 1.0
   Median distance: 1.7
   3rd quartile distance: 2.4
   Maximum distance: 4.8
   Largest minimum distance: 0.67
   Smallest maximum distance: 2.53
   Beware! 1 location has no neighbors
   You are advised to extend the distance band
```

Finally, **spatwmat** can be used to generate the eigenvalues matrix required by **spatreg**:

```
. spatwmat using columbusswm.dta, name(W) standardize eigenval(E)
The following matrices have been created:
1. Imported binary weights matrix W (row-standardized)
   Dimension: 49x49
2. Eigenvalues matrix E
   Dimension: 49x1
```

Using spatgsa

Spatial autocorrelation can be defined as the phenomenon that occurs when the spatial distribution of the variable of interest exhibits a systematic pattern (Cliff and Ord 1981). **spatgsa** computes three measures of global spatial autocorrelation: Moran's I (Moran 1948), Geary's c (Geary 1954), and Getis and Ord's G (Getis and Ord 1992). Moran's I is given by

$$I = \frac{\sum_{i=1}^{N}\sum_{j=1}^{N} w_{ij} Z_i Z_j}{S_0 m_2}$$

where w_{ij} denotes the elements of the spatial weights matrix W corresponding to the location pair (i, j), $Z_i = Y_i - \overline{Y}$, Y_i denotes the value taken on by the variable Y of interest at location i; $\bar{Y}$ denotes the mean of variable Y, $S_0 = \sum_i \sum_j w_{ij}$, and $m_2 = \sum_i Z_i^2/N$.

Under the null hypothesis of no global spatial autocorrelation, the expected value of I is given by

$$E(I) = -1/(N-1)$$

If I is larger than its expected value, then the overall distribution of variable Y can be seen as characterized by positive spatial autocorrelation, meaning that the value taken on by Y at each location i tends to be similar to the values taken on by Y at spatially contiguous locations. On the other hand, if I is smaller than its expected value, then the overall distribution of variable Y can be seen as characterized by negative spatial autocorrelation, meaning that the value taken on by Y at each location i tends to be different from the values taken on by Y at spatially contiguous locations. Inference is based on z-values, computed by subtracting $E(I)$ from I and dividing the result by the standard deviation of I

$$z_I = \frac{I - E(I)}{sd(I)}$$

The formula for the standard deviation of I depends on the assumptions about the data and the nature of spatial autocorrelation. **spatgsa** computes $sd(I)$ under the total randomization assumption (for details and formulas, see Sokal et al. 1998); under this assumption z_I follows a normal distribution (asymptotically), so that its significance can be evaluated by means of a standard normal table (Anselin 1992a).

Geary's c is computed by

$$c = (N-1)\frac{\sum_{i=1}^{N}\sum_{j=1}^{N} w_{ij}(Z_i - Z_j)^2}{2NS_0 m_2}$$

Under the null hypothesis of no global spatial autocorrelation, the expected value of c equals 1. If c is larger than 1, then the overall distribution of variable Y can be seen as characterized by negative spatial autocorrelation; on the other hand, if c is smaller than 1, then the overall distribution of variable Y can be seen as characterized by positive spatial autocorrelation. As in the case of Moran's I, inference is based on z-values, computed by subtracting 1 from c and dividing the result by the standard deviation of c

$$z_c = \frac{c - 1}{sd(c)}$$

spatgsa computes $sd(c)$ under the total randomization assumption (for details and formulas, see Sokal et al. 1998), which implies that z_c is asymptotically distributed as a standard normal variate (Anselin 1992a).

Getis and Ord's G is computed by

$$G = \frac{\sum_{i \neq j} w_{ij} Y_i Y_j}{\sum_{i \neq j} Y_i Y_j}$$

where variable Y can take only positive values, and w_{ij} denotes the elements of a nonstandardized symmetric binary weights matrix where $w_{ii} = 0$. Under the null hypothesis of no global spatial autocorrelation, the expected value of G equals

$$E(G) = \frac{S_0}{N(N-1)}$$

G measures the overall spatial clustering of values of Y in a peculiar way and, therefore, its interpretation is somewhat different from that of I and c. Specifically, if G is larger than its expected value, then the overall distribution of variable Y can be seen as characterized by positive spatial autocorrelation with a prevalence of high-valued clusters. On the other hand, if G is smaller than its expected value, then the overall distribution of variable Y is still characterized by positive spatial autocorrelation, but with a prevalence of low-valued clusters (Getis and Ord 1992). As in the other cases, inference is based on z-values, computed by subtracting $E(G)$ from G and dividing the result by the standard deviation of G

$$z_G = \frac{G - E(G)}{sd(G)}$$

Asymptotically, z_G approximates a standard normal variate (for details and formulas, see Getis and Ord 1992).

Let us now use the Columbus data to see how `spatgsa` works in practice. Since we want to compute all the statistics discussed above, we decide to use the nonstandardized symmetric binary weights matrix stored in file `columbusswm.dta`.

```
. use columbusdata.dta, clear
. spatwmat using columbusswm.dta, name(W)
The following matrix has been created:
1. Imported binary weights matrix W
   Dimension: 49x49
. spatgsa hoval income crime, weights(W) moran geary go
Measures of global spatial autocorrelation
Weights matrix
Name: W
Type: Imported (binary)
Row-standardized: No
Moran's I
```

Variables	I	E(I)	sd(I)	z	p-value*
hoval	0.220	-0.021	0.085	2.824	0.002
income	0.413	-0.021	0.086	5.067	0.000
crime	0.521	-0.021	0.087	6.212	0.000

Geary's c

Variables	c	E(c)	sd(c)	z	p-value*
hoval	0.805	1.000	0.138	-1.411	0.079
income	0.716	1.000	0.131	-2.165	0.015
crime	0.584	1.000	0.109	-3.835	0.000

Getis & Ord's G

Variables	G	E(G)	sd(G)	z	p-value*
hoval	0.098	0.099	0.006	-0.188	0.425
income	0.098	0.099	0.005	-0.057	0.477
crime	0.126	0.099	0.006	4.714	0.000

*1-tail test

To carry out a two-tailed test, we simply have to add option `twotail` to the above command.

Using spatcorr

In some cases, it is useful to evaluate global spatial autocorrelation at several levels of spatial separation, for example, for different distance bands. One way to accomplish this task is to use **spatgsa** repeatedly, each time specifying a different distance-based spatial weights matrix. Alternatively, we can use **spatcorr** to compute and optionally plot a Moran's I or Geary's c spatial correlogram based on two or more consecutive or cumulative distance bands (Cliff and Ord 1981; Bailey and Gatrell 1995; Chen and Getis 1998). For each distance band, **spatcorr** first generates a row-standardized binary weights matrix, and then uses this matrix to compute the requested statistic and the related quantities of interest.

To see how **spatcorr** works in practice, let us go back to the Columbus data. To compute a Moran's I spatial correlogram for variable crime based on five consecutive distance bands of width 1, the proper command is

```
. use columbusdata.dta, clear
. spatcorr crime, bands(0(1)5) xcoord(x) ycoord(y)
Moran's I spatial correlogram
Residential burglaries & vehicle thefts per 1,000 households
```

Distance bands	I	E(I)	sd(I)	z	p-value*
(0-1]	0.416	-0.021	0.062	7.105	0.000
(1-2]	-0.211	-0.021	0.041	-4.691	0.000
(2-3]	-0.410	-0.021	0.050	-7.756	0.000
(3-4]	0.157	-0.021	0.100	1.779	0.038
(4-5]	0.812	-0.021	0.203	4.095	0.000

*1-tail test

Were we interested in cumulative distance bands, we could use

```
. spatcorr crime, bands(0(1)5) xcoord(x) ycoord(y) cumulative
Moran's I spatial correlogram
Residential burglaries & vehicle thefts per 1,000 households
```

Distance bands	I	E(I)	sd(I)	z	p-value*
(0-1]	0.416	-0.021	0.062	7.105	0.000
(0-2]	0.036	-0.021	0.026	2.169	0.015
(0-3]	-0.093	-0.021	0.010	-6.955	0.000
(0-4]	-0.037	-0.021	0.004	-4.073	0.000
(0-5]	-0.021	-0.021	0.000	0.014	0.494

*1-tail test

Finally, we can ask **spatcorr** to plot the spatial correlogram:

```
. spatcorr crime, bands(0(1)5) xcoord(x) ycoord(y) graph
```
(*output omitted*)

which gives the graph in Figure 1.

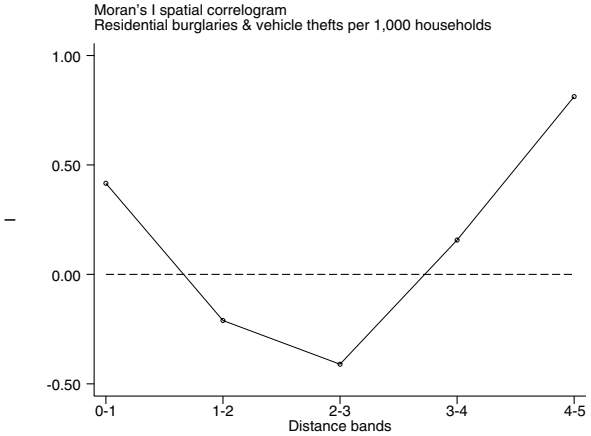

Figure 1. Moran's I spatial correlogram for the **crime** variable.

Using spatlsa

The measures of global spatial autocorrelation discussed above offer an "average" picture of the spatial distribution of the variable of interest and, therefore, may hide interesting features of the phenomenon under study. To overcome this limitation, during the last decade several measures of local spatial autocorrelation have been devised. Such statistics can serve different purposes. When applied to datasets lacking global spatial autocorrelation, local statistics may be able to reveal one or more limited areas exhibiting significant deviation from spatial randomness. When applied to datasets where global spatial autocorrelation is present, local statistics may help identify the locations that contribute most to the overall pattern of spatial clustering (Sokal et al. 1998). More generally, local statistics are employed to detect significant spatial clustering around individual locations, sometimes referred to as hotspots.

`spatlsa` computes four local spatial autocorrelation statistics: Moran's I_i, Geary's c_i, Getis and Ord's G_i, and Getis and Ord's G_i^* (for details and formulas, see Getis and Ord 1992; Anselin 1995; Sokal et al. 1998). To interpret the results displayed by `spatlsa`, the following remarks should be taken into account:

- The expected value and the standard deviation of both I_i and c_i are computed under the total randomization assumption.
- As shown by Sokal et al. (1998), significance testing for local spatial autocorrelation is problematic and, therefore, the reported p-values should be regarded just as an approximate indication of statistical significance, especially in the presence of global spatial autocorrelation. However, the statistics (especially in their standardized form) are informative when employed in an exploratory manner.
- Like their global counterparts, Moran's I_i and Geary's c_i spot both positive and negative spatial autocorrelation. Specifically, positive I_i z-values and negative c_i z-values indicate clustering of similar values of Y around location i, that is, positive local spatial autocorrelation, while negative I_i z-values and positive c_i z-values indicate clustering of dissimilar values of Y around location i, that is negative local spatial autocorrelation. On the other hand, Getis and Ord's G_i and G_i^* can detect only positive spatial autocorrelation. Specifically, positive G_i and G_i^* z-values indicate spatial clustering of high values of Y around location i, while negative G_i and G_i^* z-values indicate spatial clustering of low values of Y around location i.

To illustrate the practical use of `spatlsa`, let us consider again the Columbus data. To compute the Moran's I_i and Getis and Ord's G_i^* statistics for variable `crime`, we must enter the following commands:

```
. use columbusdata.dta, clear
. spatwmat using columbusswm.dta, name(W)
The following matrix has been created:
1. Imported binary weights matrix W
   Dimension: 49x49
. spatlsa crime, weights(W) moran go2
Measures of local spatial autocorrelation
Weights matrix
-------------------------------------------------------------------------------
Name: W
Type: Imported (binary)
Row-standardized: No
-------------------------------------------------------------------------------
Moran's Ii (Residential burglaries & vehicle thefts pe)
```

Location	Ii	E(Ii)	sd(Ii)	z	p-value*
1	1.586	-0.063	1.674	0.985	0.162
2	0.019	-0.083	1.912	0.054	0.479
3	0.460	-0.125	2.289	0.255	0.399
4	-7.442	-0.083	1.912	-3.850	0.000
5	1.474	-0.042	1.381	1.097	0.136
6	0.375	-0.083	1.912	0.240	0.405
7	2.429	-0.167	2.581	1.006	0.157
8	-0.363	-0.042	1.381	-0.233	0.408
9	4.409	-0.125	2.289	1.981	0.024
10	0.161	-0.083	1.912	0.128	0.449
11	-0.324	-0.063	1.674	-0.156	0.438
12	2.703	-0.063	1.674	1.652	0.049
13	4.959	-0.083	1.912	2.638	0.004
14	2.495	-0.063	1.674	1.528	0.063
15	0.935	-0.042	1.381	0.707	0.240
16	4.846	-0.104	2.113	2.342	0.010
17	3.414	-0.208	2.815	1.287	0.099
18	0.195	-0.146	2.443	0.140	0.444

19	-0.024	-0.125	2.289	0.044	0.482
20	1.180	-0.104	2.113	0.607	0.272
21	-0.496	-0.083	1.912	-0.216	0.415
22	0.214	-0.083	1.912	0.156	0.438
23	-0.210	-0.146	2.443	-0.026	0.490
24	3.465	-0.125	2.289	1.569	0.058
25	-0.103	-0.104	2.113	0.000	0.500
26	1.090	-0.063	1.674	0.689	0.246
27	0.814	-0.083	1.912	0.470	0.319
28	0.272	-0.083	1.912	0.186	0.426
29	0.035	-0.125	2.289	0.070	0.472
30	2.539	-0.125	2.289	1.164	0.122
31	8.793	-0.188	2.704	3.321	0.000
32	11.771	-0.146	2.443	4.877	0.000
33	10.351	-0.125	2.289	4.577	0.000
34	8.276	-0.104	2.113	3.965	0.000
35	1.600	-0.146	2.443	0.714	0.238
36	9.910	-0.167	2.581	3.904	0.000
37	4.529	-0.146	2.443	1.913	0.028
38	7.290	-0.104	2.113	3.498	0.000
39	0.501	-0.083	1.912	0.306	0.380
40	3.692	-0.125	2.289	1.667	0.048
41	2.400	-0.063	1.674	1.471	0.071
42	2.763	-0.104	2.113	1.357	0.087
43	0.465	-0.063	1.674	0.315	0.376
44	2.114	-0.083	1.912	1.149	0.125
45	2.035	-0.042	1.381	1.503	0.066
46	6.216	-0.104	2.113	2.991	0.001
47	2.304	-0.042	1.381	1.698	0.045
48	2.499	-0.042	1.381	1.840	0.033
49	2.173	-0.042	1.381	1.604	0.054

Getis & Ord's G2i (Residential burglaries & vehicle thefts pe)

Location	G2i	E(G2i)	sd(G2i)	z	p-value*
1	0.057	0.082	0.019	-1.340	0.090
2	0.099	0.102	0.021	-0.132	0.448
3	0.167	0.143	0.024	1.013	0.156
4	0.116	0.102	0.021	0.661	0.254
5	0.038	0.061	0.016	-1.433	0.076
6	0.086	0.102	0.021	-0.772	0.220
7	0.218	0.184	0.026	1.285	0.099
8	0.062	0.061	0.016	0.069	0.473
9	0.203	0.143	0.024	2.521	0.006
10	0.079	0.102	0.021	-1.138	0.128
11	0.053	0.082	0.019	-1.540	0.062
12	0.044	0.082	0.019	-2.003	0.023
13	0.043	0.102	0.021	-2.851	0.002
14	0.048	0.082	0.019	-1.821	0.034
15	0.037	0.061	0.016	-1.511	0.065
16	0.071	0.122	0.022	-2.325	0.010
17	0.189	0.224	0.028	-1.263	0.103
18	0.154	0.163	0.025	-0.375	0.354
19	0.145	0.143	0.024	0.079	0.469
20	0.158	0.122	0.022	1.595	0.055
21	0.108	0.102	0.021	0.281	0.389
22	0.111	0.102	0.021	0.435	0.332
23	0.157	0.163	0.025	-0.235	0.407
24	0.183	0.143	0.024	1.701	0.044
25	0.119	0.122	0.022	-0.158	0.437
26	0.061	0.082	0.019	-1.133	0.129
27	0.082	0.102	0.021	-0.982	0.163
28	0.091	0.102	0.021	-0.517	0.303
29	0.147	0.143	0.024	0.189	0.425
30	0.202	0.143	0.024	2.501	0.006
31	0.281	0.204	0.027	2.806	0.003
32	0.250	0.163	0.025	3.454	0.000
33	0.222	0.143	0.024	3.328	0.000
34	0.181	0.122	0.022	2.630	0.004
35	0.246	0.163	0.025	3.271	0.001
36	0.275	0.184	0.026	3.475	0.000
37	0.209	0.163	0.025	1.828	0.034

38	0.181	0.122	0.022	2.625	0.004
39	0.116	0.102	0.021	0.661	0.254
40	0.182	0.143	0.024	1.661	0.048
41	0.113	0.082	0.019	1.663	0.048
42	0.177	0.122	0.022	2.464	0.007
43	0.099	0.082	0.019	0.958	0.169
44	0.065	0.102	0.021	-1.780	0.038
45	0.033	0.061	0.016	-1.760	0.039
46	0.063	0.122	0.022	-2.675	0.004
47	0.029	0.061	0.016	-1.975	0.024
48	0.029	0.061	0.016	-1.975	0.024
49	0.032	0.061	0.016	-1.803	0.036

*1-tail test

Sometimes, it might be useful to have the results displayed in ascending order of the z-values:

```
. spatlsa crime, weights(W) moran sort
Measures of local spatial autocorrelation
Weights matrix
```

Name: W
Type: Imported (binary)
Row-standardized: No

Moran's Ii (Residential burglaries & vehicle thefts pe)

Location	Ii	E(Ii)	sd(Ii)	z	p-value*
4	-7.442	-0.083	1.912	-3.850	0.000
8	-0.363	-0.042	1.381	-0.233	0.408
21	-0.496	-0.083	1.912	-0.216	0.415
11	-0.324	-0.063	1.674	-0.156	0.438
23	-0.210	-0.146	2.443	-0.026	0.490
25	-0.103	-0.104	2.113	0.000	0.500
19	-0.024	-0.125	2.289	0.044	0.482
2	0.019	-0.083	1.912	0.054	0.479
29	0.035	-0.125	2.289	0.070	0.472
10	0.161	-0.083	1.912	0.128	0.449
18	0.195	-0.146	2.443	0.140	0.444
22	0.214	-0.083	1.912	0.156	0.438
28	0.272	-0.083	1.912	0.186	0.426
6	0.375	-0.083	1.912	0.240	0.405
3	0.460	-0.125	2.289	0.255	0.399
39	0.501	-0.083	1.912	0.306	0.380
43	0.465	-0.063	1.674	0.315	0.376
27	0.814	-0.083	1.912	0.470	0.319
20	1.180	-0.104	2.113	0.607	0.272
26	1.090	-0.063	1.674	0.689	0.246
15	0.935	-0.042	1.381	0.707	0.240
35	1.600	-0.146	2.443	0.714	0.238
1	1.586	-0.063	1.674	0.985	0.162
7	2.429	-0.167	2.581	1.006	0.157
5	1.474	-0.042	1.381	1.097	0.136
44	2.114	-0.083	1.912	1.149	0.125
30	2.539	-0.125	2.289	1.164	0.122
17	3.414	-0.208	2.815	1.287	0.099
42	2.763	-0.104	2.113	1.357	0.087
41	2.400	-0.063	1.674	1.471	0.071
45	2.035	-0.042	1.381	1.503	0.066
14	2.495	-0.063	1.674	1.528	0.063
24	3.465	-0.125	2.289	1.569	0.058
49	2.173	-0.042	1.381	1.604	0.054
12	2.703	-0.063	1.674	1.652	0.049
40	3.692	-0.125	2.289	1.667	0.048
47	2.304	-0.042	1.381	1.698	0.045
48	2.499	-0.042	1.381	1.840	0.033
37	4.529	-0.146	2.443	1.913	0.028
9	4.409	-0.125	2.289	1.981	0.024
16	4.846	-0.104	2.113	2.342	0.010
13	4.959	-0.083	1.912	2.638	0.004
46	6.216	-0.104	2.113	2.991	0.001
31	8.793	-0.188	2.704	3.321	0.000

```
  38 |   7.290   -0.104    2.113    3.498    0.000
  36 |   9.910   -0.167    2.581    3.904    0.000
  34 |   8.276   -0.104    2.113    3.965    0.000
  33 |  10.351   -0.125    2.289    4.577    0.000
  32 |  11.771   -0.146    2.443    4.877    0.000
-----+-------------------------------------------
*1-tail test
```

As an option, **spatlsa** computes and displays a Moran scatterplot (Anselin 1995). This option, however, requires that a row-standardized weights matrix be used:

```
. spatwmat using columbusswm.dta, name(WS) standardize
The following matrix has been created:
1. Imported binary weights matrix WS (row-standardized)
   Dimension: 49x49
. spatlsa crime, weights(WS) moran graph(moran) symbol(n)
  (output omitted)
```

The results of this are given in Figure 2.

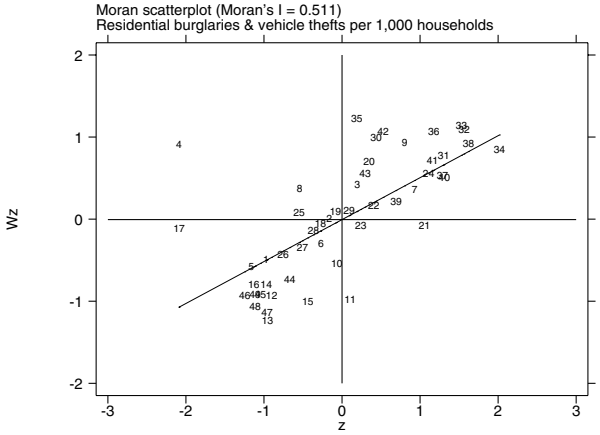

Figure 2. Moran scatterplot for variable **crime**.

The Moran scatterplot is a plot of Wz versus z, where W denotes a row-standardized spatial weights matrix and $z = (Y - \bar{Y})/sd(Y)$. The oblique line represents the linear regression line obtained by regressing Wz on z, and its slope equals Moran's I (in this case $I = 0.511$). As we can see, the Moran scatterplot is divided into four quadrants, each of which represents a different kind of spatial association:

- The upper right quadrant represents spatial clustering of high values of Y around high-value locations. In general, these locations are associated with positive values of I_i, G_i, and G_i^*, and negative values of c_i.
- The lower left quadrant represents spatial clustering of low values of Y around low-value locations. In general, these locations are associated with positive values of I_i and negative values of c_i, G_i, and G_i^*.
- The lower right quadrant represents spatial clustering of low values of Y around high-value locations. In general, these locations are associated with negative values of I_i and positive values of c_i
- The upper left quadrant represents spatial clustering of high values of Y around low-value locations. In general, these locations are associated with negative values of I_i and positive values of c_i.

As an option, **spatlsa** can display a map of Moran scatterplot values:

```
. spatlsa crime, w(WS) moran graph(moran) map(columbusboundary.dta) x(x) y(y)
  (output omitted)
```

which gives the map in Figure 3.

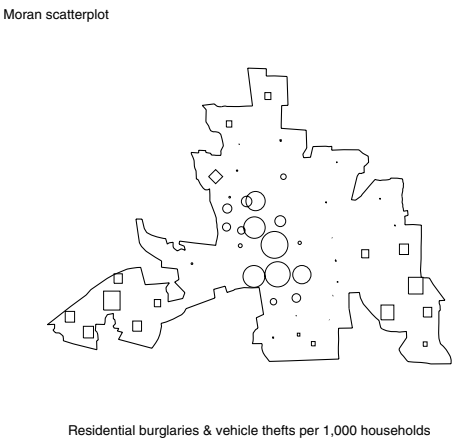

Figure 3. Map of Moran scatterplot values for variable crime.

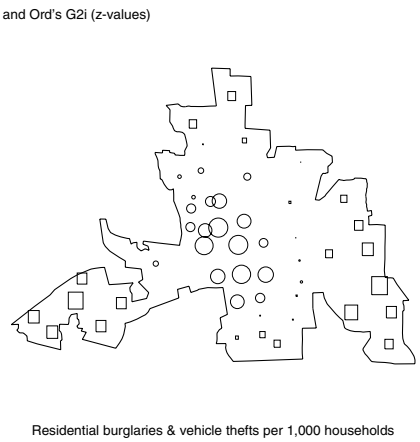

Figure 4. Map of $G_i^* z$-values for variable crime.

In this map, locations belonging to quadrant 1 of the Moran scatterplot (high-high spatial association) are represented by a circle, locations belonging to quadrant 2 (low-low spatial association) are represented by a square, locations belonging to quadrant 3 (high-low spatial association) are represented by a triangle, and locations belonging to quadrant 4 (low-high spatial association) are represented by a diamond; symbol size is proportional to the corresponding I_i z-value. The external file used to draw the boundaries of the geographical area object of analysis (in this case `columbusboundary.dta`) must include the following variables: `_ID`, which contains the identifier of the polygon(s) that make up the geographical area object of analysis, `_X`, which contains the x-coordinates of the polygon(s), and `_Y`, which contains the y-coordinates of the polygon(s). The coordinates of the polygon(s) must be expressed in the same units in which the coordinates of the locations object of analysis are expressed. Moreover, the coordinates of each polygon must be arranged so as to correspond to consecutive nodes.

With `spatlsa` it is also possible to map G_i or G_i^* z-values:

```
. spatlsa crime, w(W) go2 graph(go2) map(columbusboundary.dta) x(x) y(y)
(output omitted)
```

which gives the map in Figure 4.

In this case, locations associated with positive G_i or G_i^* z-values are represented by a circle, while locations associated with negative G_i or G_i^* z-values are represented by a square; symbol size is proportional to the corresponding z-value.

Using spatdiag

The indication of a significant pattern of spatial clustering given by spatial autocorrelation statistics represents only the first step in the analysis of spatial data. Such statistics show that the values taken on by the variable Y of interest at the different locations are more spatially clustered than they would be under a random assignment, but they do not explain why such clustering occurs (Anselin 1992b).

To answer this question, one could use the standard OLS regression model

$$Y = X\beta + \epsilon$$

where Y denotes an $N \times 1$ vector of observations on the variable Y (the dependent variable), X denotes an $N \times k$ matrix of observations on the explanatory variables, β denotes a $k \times 1$ vector of regression parameters, and ϵ denotes a vector of normally distributed, homoskedastic, and uncorrelated errors. However, when the observations are spatial units, that is, locations, the standard OLS regression model may be misspecified because of the presence of spatial dependence among observations. In this insert, two kinds of spatial dependence will be considered. The first takes the form of a spatial autoregressive process in the error term and corresponds to the following spatial regression model

$$Y = X\beta + \epsilon$$

where $\epsilon = \lambda W\epsilon + \mu$, λ denotes the spatial autoregressive parameter, μ denotes a vector of homoskedastic and uncorrelated errors, and all the other terms are defined as above (Anselin and Hudak 1992). We refer to this model as the *spatial error model*.

The second kind of spatial dependence takes the form of a mixed regressive spatial autoregressive process and corresponds to the following spatial regression model

$$Y = \rho WY + X\beta + \mu$$

where ρ denotes the spatial autoregressive parameter, WY denotes the spatially lagged dependent variable, and all the other terms are defined as above (Anselin and Hudak 1992). We refer to this model as the *spatial lag model*.

spatdiag performs several tests for both kinds of spatial dependence. Before carrying out these tests, an OLS regression model must be estimated using **regress**. Returning to the Columbus data, suppose we want to express variable **crime** (residential burglaries and vehicle thefts per 1,000 households) as a linear and additive function of the explanatory variables **hoval** (housing value in U.S. thousands of dollars), and **income** (household income in U.S. thousands of dollars):

```
. use columbusdata.dta, clear
. spatwmat using columbusswm.dta, name(W) standardize
The following matrix has been created:
1. Imported binary weights matrix W (row-standardized)
   Dimension: 49x49
. regress crime hoval income
```

Source	SS	df	MS	Number of obs =	49
				F(2, 46) =	28.39
Model	7423.32674	2	3711.66337	Prob > F =	0.0000
Residual	6014.89281	46	130.758539	R-squared =	0.5524
				Adj R-squared =	0.5329
Total	13438.2195	48	279.962907	Root MSE =	11.435

crime	Coef.	Std. Err.	t	P>|t|	[95% Conf.	Interval]
hoval	-.2739315	.1031987	-2.65	0.011	-.4816597	-.0662033
income	-1.597311	.3341308	-4.78	0.000	-2.269881	-.9247405
_cons	68.61896	4.735486	14.49	0.000	59.08692	78.151

```
. spatdiag, weights(W)
Diagnostic tests for spatial dependence in OLS regression
Fitted model
crime = hoval + income
Weights matrix
Name: W
Type: Imported (binary)
Row-standardized: Yes
Diagnostics
```

Test	Statistic	df	p-value
Spatial error:			
Moran's I	2.955	1	0.003
Lagrange multiplier	5.723	1	0.017
Robust Lagrange multiplier	0.079	1	0.778
Spatial lag:			
Lagrange multiplier	9.364	1	0.002
Robust Lagrange multiplier	3.720	1	0.054

As we can see from the output, **spatdiag** carries out three tests for spatial error dependence (Moran's I_λ, simple Lagrange multiplier LM_λ, and robust Lagrange multiplier LM_λ^*) and two tests for spatial lag dependence (simple Lagrange multiplier LM_ρ and robust Lagrange multiplier LM_ρ^*). I_λ and LM_λ test for the null hypothesis that $\lambda = 0$, while LM_ρ tests that $\rho = 0$. When testing for $\lambda = 0$, however, I_λ and LM_λ also respond to nonzero ρ; likewise, when testing for $\rho = 0$, LM_ρ also corresponds to nonzero λ. The robust tests LM_λ^* and LM_ρ^* have been devised to avoid this problem. For details and formulas pertaining to all these tests, see Anselin and Hudak (1992) and Anselin et al. (1996).

Using spatreg

spatreg estimates the spatial error and the spatial lag regression models (see above) by maximum likelihood, using Stata's **ml** routine:

```
. use columbusdata.dta, clear
. spatwmat using columbusswm.dta, name(W) eigenval(E) standardize
```

```
The following matrices have been created:
1. Imported binary weights matrix W (row-standardized)
   Dimension: 49x49
2. Eigenvalues matrix E
   Dimension: 49x1
. spatreg crime hoval income, weights(W) eigenval(E) model(error)
initial:       log likelihood = -187.42512
rescale:       log likelihood = -187.42512
rescale eq:    log likelihood = -187.42512
Iteration 0:   log likelihood = -187.42512
Iteration 1:   log likelihood = -183.49132
Iteration 2:   log likelihood = -183.38161
Iteration 3:   log likelihood = -183.38047
Iteration 4:   log likelihood = -183.38047
Weights matrix
 Name: W
 Type: Imported (binary)
 Row-standardized: Yes
Spatial error model                               Number of obs   =        49
                                                  Variance ratio  =     0.321
                                                  Squared corr.   =     0.536
Log likelihood = -183.38047                       Sigma           =      9.78
```

	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
crime						
hoval	-.3022502	.0905532	-3.34	0.001	-.4797312	-.1247692
income	-.941312	.3702766	-2.54	0.011	-1.667041	-.2155832
_cons	59.89322	5.883702	10.18	0.000	48.36137	71.42506
lambda	.5617903	.1524222	3.69	0.000	.2630482	.8605323

```
Wald test of lambda=0:                 chi2(1) =  13.585 (0.000)
Likelihood ratio test of lambda=0:     chi2(1) =   7.994 (0.005)
Lagrange multiplier test of lambda=0:  chi2(1) =   5.723 (0.017)
Acceptable range for lambda: -1.536 < lambda < 1.000
```

In addition to the usual statistics, the output produced by **spatreg** for the spatial error model reports the following information:

- Variance ratio: this is a pseudo R^2 statistic equal to $\text{Var}(\widehat{Y})/\text{Var}(Y)$, where $\text{Var}(\widehat{Y})$ denotes the variance of the predicted values of the dependent variable, and $\text{Var}(Y)$ denotes the variance of the observed values of dependent variable (Anselin 1992a).
- Squared correlation: this is another pseudo R^2 statistic equal to the squared correlation between the predicted and the observed values of the dependent variable (Anselin 1992a).
- Sigma: this is the maximum likelihood root MSE.
- The results for three tests of $\lambda = 0$ (Wald, likelihood-ratio, and LM_λ). Although these three tests are asymptotically equivalent, they tend to produce different results in finite samples. In most cases, the ordering of the test statistics in terms of their magnitude is $W \geq LR \geq LM_\lambda$ (Anselin 1988). The likelihood-ratio test is carried out only if option **robust** is not specified.
- Acceptable range for λ: the lower and the upper limits of this range correspond to the inverse of the minimum and maximum eigenvalues of the spatial weights matrix W.

The output produced by **spatreg** for the spatial lag model is exactly the same as that produced for the spatial error model, with the exception that the results for three tests of $\rho = 0$ are reported:

```
. spatreg crime hoval income, weights(W) eigenval(E) model(lag)
initial:       log likelihood = -187.42512
rescale:       log likelihood = -187.42512
rescale eq:    log likelihood = -187.42512
Iteration 0:   log likelihood = -187.42512
Iteration 1:   log likelihood = -182.65034
Iteration 2:   log likelihood = -182.39202
Iteration 3:   log likelihood = -182.39043
Iteration 4:   log likelihood = -182.39043
```

```
Weights matrix
 Name: W
 Type: Imported (binary)
 Row-standardized: Yes
Spatial lag model                                 Number of obs   =        49
                                                  Variance ratio  =     0.615
                                                  Squared corr.   =     0.652
Log likelihood = -182.39043                       Sigma           =      9.77
```

crime	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
crime						
hoval	-.2659263	.0882217	-3.01	0.003	-.4388376	-.0930149
income	-1.031616	.3284158	-3.14	0.002	-1.675299	-.3879326
_cons	45.07925	7.871214	5.73	0.000	29.65195	60.50654
rho	.4310232	.1236179	3.49	0.000	.1887366	.6733099

```
Wald test of rho=0:                     chi2(1) =  12.157 (0.000)
Likelihood ratio test of rho=0:         chi2(1) =   9.974 (0.002)
Lagrange multiplier test of rho=0:      chi2(1) =   9.364 (0.002)
Acceptable range for rho: -1.536 < rho < 1.000
```

For details on maximum likelihood estimation of the spatial error and the spatial lag regression models, see Anselin (1988) and Anselin and Hudak (1992).

Saved Results

`spatgsa` saves in `r()`:

Matrices

`r(Moran)`	Moran's I and related quantities of interest
`r(Geary)`	Geary's c and related quantities of interest
`r(GetisOrd)`	Getis and Ord's G and related quantities of interest

`spatcorr` saves in r():

Matrices

`r(Moran)`	Moran's I and related quantities of interest
`r(Geary)`	Geary's c and related quantities of interest

`spatlsa` saves in r():

Matrices

`r(Moran)`	Moran's I_i and related quantities of interest
`r(Geary)`	Geary's c_i and related quantities of interest
`r(GetisOrd1)`	Getis and Ord's G_i and related quantities of interest
`r(GetisOrd2)`	Getis and Ord's G_i^* and related quantities of interest

`spatlsa` saves in r():

Matrices

`r(stats)`	test statistics and related quantities of interest

`spatreg` saves in e():

Scalars

`e(rc)`	return code
`e(ll)`	log-likelihood
`e(rank)`	rank of e(V)
`e(rank0)`	rank of e(V) for constant-only model
`e(ll_0)`	log-likelihood for model without spatial autoregressive parameter
`e(chi2)`	χ^2
`e(k)`	number of estimated parameters
`e(ic)`	number of iterations
`e(N)`	number of observations
`e(df_m)`	degrees of freedom for χ^2 test
`e(p)`	p-value for χ^2 test
`e(k_eq)`	number of equations

`e(k_dv)`	number of dependent variables
`e(minEigen)`	1/(minimum eigenvalue)
`e(maxEigen)`	1/(maximum eigenvalue)
`e(Wald)`	Wald test statistic
`e(LM)`	simple Lagrange multiplier test statistic
`e(varRatio)`	variance ratio
`e(sqCorr)`	squared correlation between Y and $\hat{Y}$
Macros	
`e(depvar)`	name of the dependent variable
`e(cmd)`	`spatreg`
`e(opt)`	type of optimization
`e(title)`	type of spatial regression model
`e(user)`	name of likelihood-evaluator program
`e(chi2type)`	type of model χ^2 test
`e(vcetype)`	covariance estimation method
Matrices	
`e(b)`	coefficient vector
`e(V)`	variance–covariance matrix of the estimators
`e(resid)`	regression residuals
`e(yhat)`	predicted values of the dependent variable
`e(ilog)`	iteration log
Functions	
`e(sample)`	marks estimation sample

References

Anselin, L. 1988. *Spatial Econometrics: Methods and Models.* Dordrecht: Kluwer Academic.

——. 1992a. *SpaceStat Tutorial. A Workbook for Using SpaceStat in the Analysis of Spatial Data.* Regional Research Institute, West Virginia University.

——. 1992b. *Spatial Data Analysis with GIS: An Introduction to Application in the Social Sciences.* National Center for Geographic Information and Analysis, University of California, Santa Barbara.

——. 1995. Local indicators of spatial association - LISA. *Geographical Analysis* 27: 93–115.

Anselin, L. and S. Hudak. 1992. Spatial econometrics in practice. A review of software options. *Regional Science and Urban Economics* 22: 509–536.

Anselin, L., A. K. Bera, R. Florax, and M. J. Yoon. 1996. Simple diagnostic tests for spatial dependence. *Regional Science and Urban Economics* 26: 77–104.

Bailey, T. C. and A. C. Gatrell. 1995. *Interactive Spatial Data Analysis.* Harlow: Longman.

Chen, D. and A. Getis. 1998. *Point Pattern Analysis.* Department of Geography, San Diego State University.

Cliff, A. D. and J. K. Ord. 1981. *Spatial Processes: Models and Applications.* London: Pion.

Geary, R. 1954. The contiguity ratio and statistical mapping. *The Incorporated Statistician* 5: 115–145.

Getis, A. and J. K. Ord. 1992. The analysis of spatial association by use of distance statistics. *Geographical Analysis* 24: 189–206.

Moran, P. 1948. The interpretation of statistical maps. *Journal of the Royal Statistical Society, Series B* 10: 243–251.

Sokal, R. R., N. L. Oden, and B. A. Thomson. 1998. Local spatial autocorrelation in a biological model. *Geographical Analysis* 30: 331–354.

sts18 | A test for long-range dependence in a time series

Christopher F. Baum, Boston College, baum@bc.edu
Tairi Room, Boston College, room@bc.edu

Abstract: This insert implements the Hurst–Mandelbrot rescaled range statistic and the Lo (1991) modified rescaled range statistic to test for long-range dependence in a time series.

Keywords: fractional integration, long memory, rescaled range, time series.

Syntax

`lomodrs` *varname* [`if` *exp*] [`in` *range*] [`,` `maxlag(#)`]

`lomodrs` is for use with time-series data. You must `tsset` your data before using `lomodrs`; see [R] **tsset**.
varname may contain time-series operators; see [U] **14.4.3 Time-series varlists**.

Options

`maxlag(#)` specifies the maximum lag order for the test. By default, `maxlag` is calculated from the sample size and from the first-order autocorrelation coefficient of the *varname* using the data-dependent rule of Andrews (1991), assuming that the data-generating process is AR(1). If `maxlag` is set to zero, the test performed is the classical Hurst–Mandelbrot rescaled-range statistic.

Description

The model of an autoregressive fractionally integrated moving average process of a time series of order $(p,\ d,\ q)$, denoted by ARFIMA $(p,\ d,\ q)$, with mean μ, may be written using operator notation in terms of a white noise series ϵ having variance σ_ϵ^2 as

$$\Phi(L)(1-L)^d\,(y_t-\mu)=\Theta(L)\epsilon_t \tag{1}$$

where L is the backward-shift operator, $\Phi(L)=1-\phi_1L-\cdots-\phi_pL^p$, $\Theta(L)=1+\vartheta_1L+\cdots+\vartheta_qL^q$, and $(1-L)^d$ is the fractional differencing operator defined by

$$(1-L)^d=\sum_{k=0}^{\infty}\frac{\Gamma(k-d)L^k}{\Gamma(-d)\Gamma(k+1)} \tag{2}$$

with $\Gamma(\cdot)$ denoting the gamma (generalized factorial) function. The parameter d is allowed to assume any real value. The arbitrary restriction of d to integer values gives rise to the standard autoregressive integrated moving average (ARIMA) model. The stochastic process y_t is both stationary and invertible if all zeros of $\Phi(L)$ and $\Theta(L)$ lie outside the unit circle and $|d|<0.5$. The process is nonstationary for $d\geq 0.5$, as it possesses infinite variance, for example, see Granger and Joyeux (1980).

Assuming that $d\in[0,\ 0.5)$, Hosking (1981) showed that the autocorrelation function, $\rho(\cdot)$, of an ARFIMA process is proportional to k^{2d-1} as $k\to\infty$. Consequently, the autocorrelations of the ARFIMA process decay hyperbolically to zero as $k\to\infty$ in contrast to the faster, geometric decay of a stationary ARMA process. For $d\in(0,0.5)$, $\sum_{j=-n}^{n}|\rho(j)|$ diverges as $n\to\infty$, and the ARFIMA process is said to exhibit long memory, or long-range positive dependence. The process is said to exhibit intermediate memory (anti-persistence), or long-range negative dependence, for $d\in(-0.5,0)$.

The importance of long-range dependence in economic and financial time series was first studied by Mandelbrot (1972), who proposed the R/S (range over standard deviation) statistic, also known as the rescaled-range statistic, originally developed by Hurst (1951) in the context of hydrological studies. The R/S statistic is the range of the partial sums of deviations of a time series from its mean, rescaled by its standard deviation. For a sample $x_1,\ldots,x_n$,

$$Q_n=\frac{1}{s_n}\left[\max_{1\leq k\leq n}\sum_{j=1}^{k}(x_j-\bar{x}_n)-\min_{1\leq k\leq n}\sum_{j=1}^{k}(x_j-\bar{x}_n)\right]$$

where s_n is the maximum likelihood estimator of the standard deviation of x. The first bracketed term is the maximum of the partial sums of the first k deviations of x_j from the full-sample mean, which is nonnegative. The second bracketed term is the corresponding minimum, which is nonpositive. The difference of these two quantities is thus nonnegative, so that $Q_n>0$. Empirical studies have demonstrated that the R/S statistic has the ability to detect long-range dependence in the data. Like many other estimators of long-range dependence, though, the R/S statistic has been shown to be excessively sensitive to "short-range dependence," or short memory, features of the data. Lo (1991) shows that a sizable AR(1) component in the data generating process will seriously bias the R/S statistic. He modifies the R/S statistic to account for the effect of short-range dependence by applying a "Newey–West" correction (using a Bartlett window) to derive a consistent estimate of the long-range variance of the time series. For `maxlag` >0, the denominator of the statistic is computed as the Newey–West estimate of the long run variance of the series; see [R] **newey**.

Critical values for the test are taken from Table II of Lo (1991).

Saved results

`lomodrs` saves the following in `r()`:

Scalars

`r(lomodrs)`	test statistic
`r(N)`	degrees of freedom

Remarks

The description of the Hurst–Mandelbrot and Lo statistics draws heavily from Chapter 2 of Campbell et al. (1997).

Examples

Data from Terence Mills' *Econometric Analysis of Financial Time Series* on U.S. S&P 500 stock returns are analyzed.

```
. use http://fmwww.bc.edu/ec-p/data/Mills2d/sp500a.dta
. tsset
        time variable:  year, 1871 to 1997
. lomodrs sp500ar
Lo Modified R/S test for sp500ar
Critical values for H0: sp500ar is not long-range dependent
90%: [ 0.861, 1.747 ]
95%: [ 0.809, 1.862 ]
99%: [ 0.721, 2.098 ]
Test statistic:  .780838  (1 lags via Andrews criterion)  N = 124
. lomodrs sp500ar, max(0)
Hurst-Mandelbrot Classical R/S test for sp500ar
Critical values for H0: sp500ar is not long-range dependent
90%: [ 0.861, 1.747 ]
95%: [ 0.809, 1.862 ]
99%: [ 0.721, 2.098 ]
Test statistic:  .799079    N = 124
. lomodrs sp500ar if tin(1946,)
Lo Modified R/S test for sp500ar
Critical values for H0: sp500ar is not long-range dependent
90%: [ 0.861, 1.747 ]
95%: [ 0.809, 1.862 ]
99%: [ 0.721, 2.098 ]
Test statistic:  1.08705  (0 lags via Andrews criterion)  N = 50
```

Applied to the full sample, the Lo modified R/S test rejects the null hypothesis of no long-range dependence at the 95% level. The Hurst–Mandelbrot test yields a similar inference. When the sample is restricted to the postwar era, the Lo test no longer can reject the null hypothesis at any level of significance.

References

Andrews, D., 1991. Heteroskedasticity and autocorrelation consistent covariance matrix estimation. *Econometrica* 59: 817–858.

Campbell, J. Y., A. W. Lo, and A. C. MacKinlay. 1997. *The Econometrics of Financial Markets*. Princeton: Princeton University Press.

Granger, C. W. J. and R. Joyeux. 1980. An introduction to long-memory time series models and fractional differencing. *Journal of Time Series Analysis* 1: 15–39.

Hosking, J. R. M. 1981. Fractional differencing. *Biometrika* 68: 165–176.

Hurst, H. 1951. Long term storage capacity of reservoirs. *Transactions of the American Society of Civil Engineers* 116: 770–799.

Lo, A. W., 1991. Long-term memory in stock market prices. *Econometrica* 59: 1279–1313.

Mandelbrot, B., 1972. Statistical methodology for non-periodic cycles: From the covariance to R/S analysis. *Annals of Economic and Social Measurement* 1: 259–290.

sts19 | Multivariate portmanteau (Q) test for white noise

Richard Sperling, The Ohio State University, rsperling@boo.net
Christopher F. Baum, Boston College, baum@bc.edu

Abstract: This routine performs a multivariate portmanteau (Q) test for white noise in a time series context.

Keywords: autocorrelation, white noise, time series, Q test.

Syntax

`wntstmvq` *varlist* [`if` *exp*] [`in` *range*] [`, lags(#) varlags(#)`]

`wntstmvq` is for use with with time-series data. You must `tsset` your data before using `wntstmvq`; see [R] **tsset**.

varlist may contain time-series operators; see [U] **14.4.3 Time-series varlists**.

Options

`lags(#)` specifies the number of sample autocorrelations, that is, the maximum lag order to be included in the test. If not specified, it takes on a default value of $\min(N/2-2, 40)$ where N is the number of available observations.

`varlags(#)` specifies the order of the VAR (vector autoregression) used to produce the series in *varlist*. If specified, `varlags` must not exceed `lags`.

Description

`wntstmvq` performs the multivariate Ljung–Box portmanteau (or Q) test for white noise in a set of time series. This test is a generalization of the univariate Ljung–Box portmanteau (Q) test implemented in Stata as `wntestq`. The multivariate form of the test was proposed by Hosking (1980) and others. Hosking (1981) demonstrated the equivalence of the several forms in the literature. The test implemented here is that described in Johansen (1995, 22). It is often applied to the residuals of a multivariate regression, such as a VAR (vector autoregression).

The null hypothesis of the multivariate test is that the autocorrelation functions of all series in *varlist* have no significant elements for lags one through that specified by the `lags` option. The `lags` parameter may be specified by the user. If the series in *varlist* are residuals from a vector autoregression, the *varlist* option should be specified to provide the order of the VAR.

As a function of s lags, the test statistic

$$LB(s) = T\,(T+2)\sum_{j=1}^{s}\frac{1}{T-j}\mathrm{tr}\left[\widehat{C}_{0j}\widehat{C}_{00}^{-1}\widehat{C}_{0j}'\widehat{C}_{00}^{-1}\right]$$

where

$$\widehat{C}_{0j} = T^{-1}\sum_{t=j+1}^{T}\widehat{\epsilon}_t\widehat{\epsilon}_{t-j}'$$

is distributed, under the null hypothesis, as χ^2 with degrees of freedom equal to $p^2(\texttt{lags}-\texttt{varlags})$ where p is the number of series in *varlist*. A rejection indicates that at least one series is not white noise.

Although portmanteau statistics are commonly applied in diagnosing time series models, some caution should be exercised with their use in a cointegration context. Jacobson (1995, 179) states "[O]ne should exercise some care when using the portmanteau statistic for evaluating the fit of a cointegration model. This observation is due to the facts that cointegration implies the presence of unit roots and an assumption underlying the properties of the portmanteau statistic is that of a stationary process disqualifying roots on the unit circle. There is to my knowledge no theoretical result justifying the use of portmanteau statistics in connection with potential unit roots. Nevertheless these tests are being used ..."

Some guidance for the application of tests of this nature in the vector autoregressive setting is given by Bender and Grouven (1993), who find that the number of lags must be carefully chosen in order to avoid significant loss of power of the test. They conduct a simulation study and tabulate appropriate choices of the number of lags as a function of sample size, model order, and model dimension.

Saved Results

`wntstmvq` saves the following scalars in `r()`:

`r(stat)`	test statistic	`r(k)`	number of series
`r(df)`	degrees of freedom	`r(s)`	maximum lag order
`r(p)`	p-value	`r(nobs)`	number of observations

Examples

The Grunfeld investment data (20 years of annual data on five U.S. corporations) are analyzed.

```
. set matsize 100
. use http://fmwww.bc.edu/ec-p/data/Greene2000/tbl15-1.dta,clear
. reshape wide i f c,i(year) j(firm)
 (output omitted)
. tsset year,yearly
        time variable:  year, 1935 to 1954
. mvreg i1 i2 i3  = f1 c1 f2 c2 f3 c3
 (output omitted)
```

```
. for num 1/3:predict epsX,equation(iX) r
  (output omitted)
. wntstmvq eps1-eps3
Multivariate Ljung-Box statistic (3 variables, 8 lags):   98.5122
Prob > chi2(72) = 0.0207
```

The three residual series from this multivariate regression model do not appear to be white noise.

References

Bender, R., and U. Grouven. 1993. On the choice of the number of residual autocovariances for the portmanteau test of multivariate autoregressive models. *Communications in Statistics-Simulation and Computation* 22: 19–32.

Hosking, J. R. M. 1980. The multivariate portmanteau statistic. *Journal of the American Statistical Association* 75: 602–08.

——. 1981. Equivalent forms of the multivariate portmanteau statistic. *Journal of the Royal Statistical Society, Series B* 43: 261–62.

Jacobson, T. 1995. On the determination of lag order in vector autoregressions of cointegrated systems. *Computational Statistics* 10: 177–92.

Johansen, S. 1995. *Likelihood-Based Inference in Cointegrated Vector Autoregressive Models*. New York: Oxford University Press.

sxd4	Sample size estimation for cluster designed samples

Joanne M. Garrett, University of North Carolina, joanne_garrett@med.unc.edu

Abstract: The command `sampclus` which estimates sample sizes required for the difference of means or proportions, adjusted for cluster size and intraclass correlation, is described and illustrated.

Keywords: cluster sampling, sample size.

Background

More and more frequently we see studies that use a cluster sampling design. In this design, a sample (preferably random) of natural groups of individuals is selected, rather than a random sample of the individuals themselves. We are interested in examining the individuals, not the groupings to which they belong. However, a key assumption for most statistical tests is that the observations are independent. This assumption may be violated using this sampling technique. Ideally, it would be preferable to have a random sample of observations, but at times this may not be cost effective or even feasible.

Consider a study of the effect of the presence of sciatica on the length of time to recovery for patients with acute low back pain. A random sample of physicians (clusters) is selected, and each physician then recruits a sample of low back pain patients (observations). The study question is interested in the patients, not the physicians. However, patients seeing the same physician may not be independent. There may be characteristics or practice patterns of Physician A that help patients to recover from their low back pain more quickly, while patients seeing Physician B may not receive the same benefits. Therefore, patients seeing Physician A may be more correlated to each other than we would expect had they been drawn randomly from the population, and thus are not independent.

This correlation factor is referred to as intraclass correlation. Nonindependence of the observations will not bias our point estimates (for example, change our estimated means or proportions), but can result in standard errors that are too small. We must correct the standard errors for any nonindependence of patients due to a physician effect. Using a cluster sample design does not necessarily mean intraclass correlation will be an issue. In order for it to affect our estimates, the factors causing the correlation must have some effect on the outcome we're studying. For instance, if Physician "A" encourages all her patients to get flu shots, and Physician "B" does not, we would not necessarily expect this practice to differentially affect how quickly patients recover from back pain. However, it is difficult to predict beforehand whether intraclass correlation will be a problem. Therefore, whenever we use a cluster sample design, we must assume nonindependence is possible and prepare for it accordingly.

Until recently, studies seemed to ignore any potential intraclass correlation problems, perhaps for as simple a reason as there were no easy ways on the computer to make the adjustment. Now, the correction is easy to do, so most authors of journal articles are very conscientious about reporting and adjusting for intraclass correlation. The method for doing so is particularly easy in Stata, using either the cluster option which is available for many of Stata's programs (for example, `regress` or `logistic`) or the `svy` commands. But, given that an inflation of the standard errors by definition reduces power, it follows that we will need larger sample sizes to detect the difference in effect that we hope to find when we design a study using a cluster method of sampling. Funding agencies have become much more critical of grant proposals that do not take this potential problem into account when selecting an adequate sample size.

In addition to the usual factors needed to estimate sample size (effect size, standard deviation, alpha, and power), there are two other factors that will directly affect how many observations we need to collect. The first is the amount of intraclass correlation present. In reality, for studies using a cluster design simply for feasibility reasons (as in the example of physicians

selecting patients), the intraclass correlations tend to be fairly small; often between 0 and 0.05, and rarely more than 0.1, although theoretically they could be higher. In fact, sometimes the corrected p-values are even smaller than without the correction (the intraclass correlation is negative). This makes no intuitive sense, since it implies that observations within a cluster are more different from each other than we might expect had they been chosen randomly.

The second additional factor affecting sample size is the number of observations per cluster. The larger the number of clusters and the fewer observations per cluster, the less the effect on the standard error estimates. In other words, a sample of 250 consisting of 50 clusters with 5 observations per cluster will have better power than the same sample size consisting of 25 clusters with 10 observations per cluster. In fact, with only one observation per cluster, we're back to a simple random sample.

If we can make a guess at these two factors, that is, intraclass correlation and cluster size, we can adjust our sample size calculations by multiplying the variance of the sample size formulas by the following correction (Kish 1965)

$$1 + \phi(m - 1)$$

where m =average number of observations per cluster, ϕ =intraclass correlation

If there is no intraclass correlation, that is, $\phi = 0$, this correction factor reduces to 1, thus causing no increase in the sample size estimate.

Syntax

`sampclus , { obsclus(#) | numclus(#) } [rho(#)]`

Description

`sampclus` estimates sample sizes required for the difference of means or proportions, adjusted for cluster size and intraclass correlation. If the cluster size is one or the intraclass correlation is zero, the sample sizes will be the same as those calculated using `sampsi`. `sampclus` uses the sample size estimates produced by `sampsi`, which must be specified first.

Options

`obsclus(#)` is the number of observations in each cluster. When specified, it will return corrected sample sizes and the minimum number of clusters needed. For example, `obsclus(10)` calculates corrected sample size and number of physicians (clusters) needed assuming each physician recruits 10 patients.

`numclus(#)` is the maximum number of clusters. When specified, it will return corrected sample sizes and the number of observations needed in each cluster. The number of observations per cluster is rounded up so that the number of clusters will not exceed `numclus`, but may be fewer. If too few clusters are requested for the other given parameters, `sampclus` will prompt for the minimum number of clusters possible. For example, `numclus(40)` calculates corrected sample size and number of patients needed per physician assuming 40 physicians (clusters) are chosen.

`rho(#)` is the intraclass correlation (from 0 to 1.0). The default is `rho(0)`, which causes no change in the original estimated sample sizes.

Note that one must choose either `obsclus` or `numclus`, but not both.

Illustrating difference of two means, selecting number of observations per cluster

In the study of sciatica and length of time to recovery from acute low back pain, we know that patients on average take 14 days (with a standard deviation of 20) to recover. We expect patients with sciatica to take an additional 7 days to improve (21 days). In designing a study to test this hypothesis, the sample sizes we would need to detect a 7 day difference in recovery time between the two groups, with a two-sided alpha of 0.05 and 80% power is

```
. sampsi 14 21, sd(20) p(.8) a(.05)
Estimated sample size for two-sample comparison of means
Test Ho: m1 = m2, where m1 is the mean in population 1
                  and m2 is the mean in population 2
Assumptions:
         alpha =   0.0500  (two-sided)
         power =   0.8000
            m1 =       14
            m2 =       21
           sd1 =       20
           sd2 =       20
         n2/n1 =     1.00
Estimated required sample sizes:
            n1 =      129
            n2 =      129
```

We would need 129 patients with sciatica and 129 patients without sciatica to detect a 7 day difference in recovery time.

Suppose we plan to collect our data taking a random sample of physicians and expect each physician to recruit 10 patients (cluster sample design). How would this affect our sample size? After running `sampsi`, we can use `sampclus` to answer this question.

```
. sampclus, obsclus(10)
Sample Size Adjusted for Cluster Design
   n1 (uncorrected) = 129
   n2 (uncorrected) = 129
   Intraclass correlation     = 0
   Average obs. per cluster   = 10
   Minimum number of clusters = 26
   Estimated sample size per group:
      n1 (corrected) = 129
      n2 (corrected) = 129
```

We still need only 129 patients per group (sciatica versus no sciatica) and a total of 26 physicians (clusters). The intraclass correlation was not specified, thus defaulting to 0, causing no increase in sample size.

This time let's guess that there will be an intraclass correlation of about 0.1.

```
. sampclus, obsclus(10) rho(.1)
Sample Size Adjusted for Cluster Design
   n1 (uncorrected) = 129
   n2 (uncorrected) = 129
   Intraclass correlation     = .1
   Average obs. per cluster   = 10
   Minimum number of clusters = 50
   Estimated sample size per group:
      n1 (corrected) = 246
      n2 (corrected) = 246
```

The intraclass correlation has increased each sample from 129 to 246, and now requires at least 50 physicians.

Next, we will try reducing the observations per cluster to five, that is, each physician will recruit five rather than 10 patients, but maintain the same intraclass correlation.

```
. sampclus, obsclus(5) rho(.1)
Sample Size Adjusted for Cluster Design
   n1 (uncorrected) = 129
   n2 (uncorrected) = 129
   Intraclass correlation     = .1
   Average obs. per cluster   = 5
   Minimum number of clusters = 73
   Estimated sample size per group:
      n1 (corrected) = 181
      n2 (corrected) = 181
```

We are able to get by with fewer patients (181 per group), but must increase the number of physicians in the study to 73.

Finally, we know that only about 25% of low back pain patients tend to have sciatica, so we look at a sample size calculation for 10 observations per physician, correlation of 0.1, and three times as many patients without sciatica as with sciatica.

```
. sampsi 14 21, sd(20) p(.8) a(.05) ratio(3)
Estimated sample size for two-sample comparison of means
Test Ho: m1 = m2, where m1 is the mean in population 1
                  and m2 is the mean in population 2
Assumptions:
         alpha =   0.0500  (two-sided)
         power =   0.8000
            m1 =       14
            m2 =       21
           sd1 =       20
           sd2 =       20
         n2/n1 =     3.00
```

```
Estimated required sample sizes:

            n1 =       86
            n2 =      258

. sampclus, obsclus(10) rho(.1)

Sample Size Adjusted for Cluster Design

   n1 (uncorrected) = 86
   n2 (uncorrected) = 258

   Intraclass correlation     = .1

   Average obs. per cluster   = 10
   Minimum number of clusters = 66

   Estimated sample size per group:

      n1 (corrected) = 164
      n2 (corrected) = 492
```

Thus, we need 164 patients with sciatica and 492 without, and 66 physicians selecting 10 patients each.

Illustrating difference of two means, selecting number of clusters

Sometimes, we know ahead of time that the number of clusters may be limited, that is, we know we will not be able to recruit more than, say, 40 physicians. It is possible to specify the number of physicians (clusters) rather than the number of patients per physician. Then, `sampclus` will calculate the sample size needed for each group and the number of observations per cluster. We go back to the first power calculation from the sciatica study, and calculate the number of patients needed for each group (sciatica versus no sciatica) and the number of patients each physician must recruit, assuming 40 physicians and an intraclass correlation of 0.1.

```
. sampsi 14 21, sd(20) p(.8) a(.05)
  (output omitted)
. sampclus, numclus(40) rho(.1)

Sample Size Adjusted for Cluster Design

   n1 (uncorrected) = 129
   n2 (uncorrected) = 129

   Intraclass correlation     = .1

   Average obs. per cluster   = 17
   Minimum number of clusters = 40

   Estimated sample size per group:

      n1 (corrected) = 336
      n2 (corrected) = 336
```

We need 336 patients with sciatica, 336 patients without sciatica, and 40 physicians, each recruiting 17 patients, in order to correct our sample size estimate for an intraclass correlation of 0.1.

Illustrating difference of two proportions

Using the same study, let the outcome be the proportion of patients who have not recovered from their back pain by 4 weeks. We know that typically about 20% of acute low back pain patients have not recovered by four weeks. We think the percentage may be 30% for patients who also have sciatica. The sample sizes we would need to detect this difference between the two groups, with a two-sided alpha of 0.05 and 80% power is

```
. sampsi .2 .3, p(.8) a(.05)

Estimated sample size for two-sample comparison of proportions

Test Ho: p1 = p2, where p1 is the proportion in population 1
                  and p2 is the proportion in population 2
Assumptions:

         alpha =   0.0500  (two-sided)
         power =   0.8000
            p1 =   0.2000
            p2 =   0.3000
         n2/n1 =   1.00

Estimated required sample sizes:

            n1 =      313
            n2 =      313
```

We need 313 patients with sciatica and 313 patients without sciatica to see the 10% difference in the proportion of patients who have not recovered by four weeks.

Now, we look at the same example using 10 observations per cluster and an intraclass correlation of 0.1.

```
. sampclus, obsclus(10) rho(.1)
Sample Size Adjusted for Cluster Design
   n1 (uncorrected) = 313
   n2 (uncorrected) = 313
   Intraclass correlation     = .1
   Average obs. per cluster   = 10
   Minimum number of clusters = 119
   Estimated sample size per group:
      n1 (corrected) = 595
      n2 (corrected) = 595
```

Taking into account the cluster design of the study, our sample sizes have increased to 595 per group and a total of 119 physicians.

Finally, we try the calculation assuming only 40 physicians.

```
. sampclus, numclus(40) rho(.1)
For this rho, the minimum number of clusters possible is: 63
```

We get an error message telling us that, given these sets of sample size parameters, it is not possible to estimate a sample size with fewer than 63 clusters. We can either rerun `sampclus` with 63 clusters or try reducing the correlation. First, we will try using 63 clusters.

```
. sampclus, numclus(63) rho(.1)
Sample Size Adjusted for Cluster Design
   n1 (uncorrected) = 313
   n2 (uncorrected) = 313
   Intraclass correlation     = .1
   Average obs. per cluster   = 1410
   Minimum number of clusters = 63
   Estimated sample size per group:
      n1 (corrected) = 44415
      n2 (corrected) = 44415
```

Unfortunately, even with 63 clusters, the number of patients needed would be impossible to get, so we try going back to 40 clusters, and reduce the correlation to 0.05.

```
. sampclus, numclus(40) rho(.05)
Sample Size Adjusted for Cluster Design
   n1 (uncorrected) = 313
   n2 (uncorrected) = 313
   Intraclass correlation     = .05
   Average obs. per cluster   = 69
   Minimum number of clusters = 40
   Estimated sample size per group:
      n1 (corrected) = 1378
      n2 (corrected) = 1378
```

This is a more manageable number of patients than in the previous example, but even a small amount of intraclass correlation with only 40 clusters causes a fairly large increase in our estimated sample size compared to the uncorrected estimate. As with all sample size calculations, we may need to experiment with several combinations of values until we find a sample size that is both valid, as well as realistic.

References

Kish L. 1965. *Survey Sampling*. New York: John Wiley and Sons.

STB categories and insert codes

Inserts in the STB are presently categorized as follows:

General Categories:

an	announcements	*ip*	instruction on programming
cc	communications & letters	*os*	operating system, hardware, & interprogram communication
dm	data management		
dt	datasets	*qs*	questions and suggestions
gr	graphics	*tt*	teaching
in	instruction	*zz*	not elsewhere classified

Statistical Categories:

sbe	biostatistics & epidemiology	*ssa*	survival analysis
sed	exploratory data analysis	*ssi*	simulation & random numbers
sg	general statistics	*sss*	social science & psychometrics
smv	multivariate analysis	*sts*	time-series, econometrics
snp	nonparametric methods	*svy*	survey sampling
sqc	quality control	*sxd*	experimental design
sqv	analysis of qualitative variables	*szz*	not elsewhere classified
srd	robust methods & statistical diagnostics		

In addition, we have granted one other prefix, *stata*, to the manufacturers of Stata for their exclusive use.

Guidelines for authors

The Stata Technical Bulletin (STB) is a journal that is intended to provide a forum for Stata users of all disciplines and levels of sophistication. The STB contains articles written by StataCorp, Stata users, and others.

Articles include new Stata commands (ado-files), programming tutorials, illustrations of data analysis techniques, discussions on teaching statistics, debates on appropriate statistical techniques, reports on other programs, and interesting datasets, announcements, questions, and suggestions.

A submission to the STB consists of

1. An insert (article) describing the purpose of the submission. The STB is produced using plain TEX so submissions using TEX (or LATEX) are the easiest for the editor to handle, but any word processor is appropriate. If you are not using TEX and your insert contains a significant amount of mathematics, please FAX (979–845–3144) a copy of the insert so we can see the intended appearance of the text.

2. Any ado-files, `.exe` files, or other software that accompanies the submission.

3. A help file for each ado-file included in the submission. See any recent STB diskette for the structure a help file. If you have questions, fill in as much of the information as possible and we will take care of the details.

4. A do-file that replicates the examples in your text. Also include the datasets used in the example. This allows us to verify that the software works as described and allows users to replicate the examples as a way of learning how to use the software.

5. Files containing the graphs to be included in the insert. If you have used STAGE to edit the graphs in your submission, be sure to include the `.gph` files. Do not add titles (e.g., "Figure 1: ...") to your graphs as we will have to strip them off.

The easiest way to submit an insert to the STB is to first create a single "archive file" (either a `.zip` file or a compressed `.tar` file) containing all of the files associated with the submission, and then email it to the editor at `stb@stata.com` either by first using `uuencode` if you are working on a Unix platform or by attaching it to an email message if your mailer allows the sending of attachments. In Unix, for example, to email the current directory and all of its subdirectories:

```
tar -cf - . | compress | uuencode xyzz.tar.Z > whatever
mail stb@stata.com < whatever
```

International Stata Distributors

International Stata users may also order subscriptions to the *Stata Technical Bulletin* from our International Stata Distributors.

Company:	Applied Statistics & Systems Consultants Ltd.	Countries served:	Israel
		Phone:	+972 (0)4 6100101
Address:	P.O. Box 5	Fax:	+972 (0)4 6100101
	38900 CAESAREA, Israel	Email:	stata@watchwise.net
Company:	Axon Technology Company Ltd	Countries served:	Taiwan
Address:	9F, No. 259, Sec. 2	Phone:	+886-(0)2-27045535
	Ho-Ping East Road	Fax:	+886-(0)2-27541785
	TAIPEI 106, Taiwan	Email:	hank@axon.axon.com.tw
Company:	Chips Electronics	Countries served:	Indonesia, Brunei, Malaysia, Singapore
Address:	Kelapa Puyuh IV KB 23	Phone / Fax:	62 - 21 - 452 17 61
	Kelapa Gading Permai	Mobile Phone:	62 - 81 - 884 95 84
	JAKARTA 14240, Indonesia	Email:	puyuh23@indo.net.id
Company:	Dittrich & Partner Consulting	Countries served:	Germany, Austria, Czech Republic, Hungary, Poland
Address:	Kieler Straße 17		
	5. floor	Phone:	+49 2 12 / 26 066 - 0
	D-42697 Solingen, Germany	Fax:	+49 2 12 / 26 066 - 66
URL:	http://www.dpc.de	Email:	sales@dpc.de
Company:	IEM	Countries served:	South Africa, Botswana, Lesotho, Namibia, Mozambique, Swaziland, Zimbabwe
Address:	P.O. Box 2222		
	PRIMROSE 1416, South Africa		
		Phone:	+27-11-8286169
		Fax:	+27-11-8221377
		Email:	iem@hot.co.za
Company:	JasonTech Inc.	Countries served:	Korea
Address:	181-3 Hansang B/D, Bangyidong	Phone:	+82-(0)2-420-6700
	Songpaku	Fax:	+82-(0)2-420-8600
	Seoul 138-050, Korea	Email:	info@jat.co.kr
Company:	MercoStat Consultores	Countries served:	Uruguay, Argentina, Brazil, Paraguay
Address:	9 de junio 1389	Phone:	598-2-613-7905
	CP 11400 MONTEVIDEO, Uruguay	Fax:	598-2-613-7905
		Email:	mercost@adinet.com.uy
Company:	Metrika Consulting	Countries served:	Sweden, Baltic States, Denmark, Finland, Iceland, Norway
Address:	Mosstorpsvagen 48		
	183 30 Taby STOCKHOLM, Sweden	Phone:	+46-708-163128
URL:	http://www.metrika.se	Fax:	+46-8-7924747
		Email:	sales@metrika.se
Company:	MultiON Consulting, SA de CV	Countries served:	Mexico
Address:	Insurgentes Sur 1236-301	Phone:	52 (5) 559-4050 Ext 190
	Mexico, DF, 03200, Mexico	Fax:	52 (5) 559-4048
		Email:	multion@multion.com.mx

(*List continued on next page*)

International Stata Distributors

(*Continued from previous page*)

Company:	Ritme Informatique	Countries served:	France, Belgium, Luxembourg
Address:	34, boulevard Haussmann	Phone:	+33 (0)1 42 46 00 42
	75009 Paris, France	Fax:	+33 (0)1 42 46 00 33
URL:	http://www.ritme.com	Email:	info@ritme.com
Company:	Scientific Solutions S.A.	Countries served:	Switzerland
Address:	Avenue du Général Guisan, 5	Phone:	41 (0)21 711 15 20
	CH-1009 Pully/Lausanne, Switzerland	Fax:	41 (0)21 711 15 21
		Email:	info@scientific-solutions.ch
Company:	Smit Consult	Countries served:	Netherlands
Address:	Doormanstraat 19	Phone:	+31 416-378 125
	5151 GM Drunen, Netherlands	Fax:	+31 416-378 385
URL:	http://www.smitconsult.nl	Email:	info@smitconsult.nl
Company:	Survey Design & Analysis Services Pty Ltd	Countries served:	Australia, New Zealand
Address:	PO Box 1206	Phone:	+61 (0)3 9878 7373
	Blackburn North VIC 3130, Australia	Fax:	+61 (0)3 9878 2345
URL:	http://survey-design.com.au	Email:	sales@survey-design.com.au
Company:	Timberlake Consultants	Countries served:	United Kingdom, Eire
Address:	Unit B3 Broomsleigh Bus. Park	Phone:	+44 (0)208 697 3377
	Worsley Bridge Road	Fax:	+44 (0)208 697 3388
	LONDON SE26 5BN, United Kingdom	Email:	info@timberlake.co.uk
URL:	http://www.timberlake.co.uk		
Company:	Timberlake Consultants Srl	Countries served:	Italy
Address:	Via Baden Powell, 8	Phone:	+39 (0)864 210101
	67039 SULMONA (AQ), Italy	Fax:	+39 (0)864 206014
URL:	http://www.timberlake.it	Email:	timberlake@arc.it
Company:	Timberlake Consulting S.L.	Countries served:	Spain
Address:	Calle Mendez Nunez, 1, 3	Phone:	+34 95 421 9369
	41011 Sevilla, Spain	Fax:	+34 95 422 0648
		Email:	timberlake@zoom.es
Company:	Timberlake Consultores, Lda.	Countries served:	Portugal
Address:	Praceta Raúl Brandao, n°1, 1°E	Phone:	351 (0)1 471 73 47
	2720 ALFRAGIDE, Portugal	Fax:	+351 (0)1 471 73 47
		Email:	timberlake.co@mail.telepac.pt
Company:	Vishvas Marketing-Mix Services	Countries served:	India
Address:	C\O S. D. Wamorkar	Phone:	+91-251-440087
	"Prashant" Vishnu Nagar, Naupada	Fax:	+91-22-5378552
	THANE - 400602, India	Email:	vishvas@vsnl.com