The Stata Journal

The *Stata Journal* publishes reviewed papers together with shorter notes or comments, regular columns, book reviews, and other material of interest to Stata users. Examples of the types of papers include 1) expository papers that link the use of Stata commands or programs to associated principles, such as those that will serve as tutorials for users first encountering a new field of statistics or a major new technique; 2) papers that go "beyond the Stata manual" in explaining key features or uses of Stata that are of interest to intermediate or advanced users of Stata; 3) papers that discuss new commands or Stata programs of interest either to a wide spectrum of users (e.g., in data management or graphics) or to some large segment of Stata users (e.g., in survey statistics, survival analysis, panel analysis, or limited dependent variable modeling); 4) papers analyzing the statistical properties of new or existing estimators and tests in Stata; 5) papers that could be of interest or usefulness to researchers, especially in fields that are of practical importance but are not often included in texts or other journals, such as the use of Stata in managing datasets, especially large datasets, with advice from hard-won experience; and 6) papers of interest to those who teach, including Stata with topics such as extended examples of techniques and interpretation of results, simulations of statistical concepts, and overviews of subject areas.

The *Stata Journal* is indexed and abstracted by *CompuMath Citation Index*, *Current Contents/Social and Behavioral Sciences*, *RePEc: Research Papers in Economics*, *Science Citation Index Expanded* (also known as *SciSearch*, *Scopus*, and *Social Sciences Citation Index*.

For more information on the *Stata Journal*, including information for authors, see the webpage

http://www.stata-journal.com

Subscriptions are available from StataCorp, 4905 Lakeway Drive, College Station, Texas 77845, telephone 979-696-4600 or 800-STATA-PC, fax 979-696-4601, or online at

http://www.stata.com/bookstore/sj.html

Subscription rates listed below include both a printed and an electronic copy unless otherwise mentioned.

U.S. and Canada		Elsewhere	
Printed & electronic		**Printed & electronic**	
1-year subscription	$ 98	1-year subscription	$138
2-year subscription	$165	2-year subscription	$245
3-year subscription	$225	3-year subscription	$345
1-year student subscription	$ 75	1-year student subscription	$ 99
1-year university library subscription	$125	1-year university library subscription	$165
2-year university library subscription	$215	2-year university library subscription	$295
3-year university library subscription	$315	3-year university library subscription	$435
1-year institutional subscription	$245	1-year institutional subscription	$285
2-year institutional subscription	$445	2-year institutional subscription	$525
3-year institutional subscription	$645	3-year institutional subscription	$765
Electronic only		**Electronic only**	
1-year subscription	$ 75	1-year subscription	$ 75
2-year subscription	$125	2-year subscription	$125
3-year subscription	$165	3-year subscription	$165
1-year student subscription	$ 45	1-year student subscription	$ 45

Back issues of the *Stata Journal* may be ordered online at

http://www.stata.com/bookstore/sjj.html

Individual articles three or more years old may be accessed online without charge. More recent articles may be ordered online.

http://www.stata-journal.com/archives.html

The *Stata Journal* is published quarterly by the Stata Press, College Station, Texas, USA.

Address changes should be sent to the *Stata Journal*, StataCorp, 4905 Lakeway Drive, College Station, TX 77845, USA, or emailed to sj@stata.com.

The *Stata Journal* (ISSN 1536-867X) is a publication of Stata Press. Stata, Stata, Stata Press, Mata, Mata, and NetCourse are registered trademarks of StataCorp LP.

The Stata Journal (2013)
13, Number 1, pp. 39–64

Sar: Automatic generation of statistical reports using Stata and Microsoft Word for Windows

Giovanni L. Lo Magno
Department of Agriculture and Forestry Sciences
University of Palermo
Palermo, Italy
lomagno.gl@virgilio.it

Abstract. The output provided by most Stata commands is plain text not suitable to be presented or published. After the numerical and graphical outputs are obtained, the user has to copy them into a word processor to complete the editing process. Some Stata commands help you to obtain well-formatted output, especially tabulated results in LaTeX or other formats, but they are not a complete solution nor are they friendly tools. Stata automatic report (Sar) is an easy-to-use macro for Microsoft Word for Windows that allows a powerful integration between Stata and Word. With Sar, the user can retrieve numerical results and graphs from Stata and automatically insert them into a well-formatted Word document, exploiting all the functions of Word. This process is managed by Word while Stata is executed in the background. Sar requires Stata commands and some specific Sar commands to be written in ordinary Word comments. Thus the report is well documented, and this can encourage the sharing of the workflow of data analysis and the reproducibility of the research. With Sar, the user can create an automatic report, that is, a Word document that can be automatically updated if data have changed. Sar works only on Windows systems.

Keywords: pr0055, Sar, Stata Automation object, report automation, Microsoft Word, reproducible research, Automation, OLE

1 Introduction

Presenting results is an essential step in the workflow of data analysis (Long 2008); nevertheless, it is often not well integrated with the other steps of the process. Once the statistical analysis is completed, results must be copied manually from Stata to another software to create presentations, reports, articles, books, or webpages. An efficient workflow should be automated, allowing the user to focus on core issues and save time.

Several user-written Stata commands are available to produce output that is readily usable in a LaTeX environment (Knuth 1986; Lamport 1994) or in editing software like a word processor. The `listtex` command by Newson (2003) uses values from a Stata dataset to create rows of a table in several formats: LaTeX, HTML, or plain text with separators. The `textab` command by Hardin (1995) generates TeX code to format a table containing values from a Stata dataset, so it allows some options to customize the table. The `estout` command by Jann (2005) is a tool for creating tables from stored

estimates. The `esttab` command by Jann (2007) simplifies the usage of the `estout` command. A dedicated command to the creation of a table of regression results is the `outreg` command by Gallup (2012).

Generally, the existing commands present the following limitations: 1) most of them are LaTeX oriented, and many users think working in LaTeX is difficult or boring; 2) their syntax is complex and not easy to remember; 3) they do not adopt a "what you see is what you get" approach.

In this article, I present a software called Stata automatic report (Sar), which the current version is 1.1. Sar is not a Stata command, but a macro for Microsoft Word, written in the Visual Basic for Applications (VBA) programming language. In short, a macro is a program executed by Word itself, often used to automate repetitive tasks or to extend the functions of Word.

Sar is used to facilitate the creation of Word documents that present numerical results obtained through Stata. To do this, Sar directly retrieves data from Stata, exploiting Stata Automation (for details, see http://www.stata.com/automation/), and automatically inserts them in the Word document. This basic feature avoids the process of manually copying numerical output from Stata to Word.

Stata Automation is a framework allowing Windows applications to interact with Stata. To use Stata Automation, you have to install the Stata Automation object (see section 2). Because Sar exploits Stata Automation, it can run only on Windows machines.

Sar allows the creation of automatic reports, that is, Word documents that can update themselves if data change. Stata commands used to produce the statistical output and the Sar commands used to manage formatting have to be written in Word comments. These comments are associated with portions of text (for example, tables) that are updated with data retrieved from Stata according to the commands written in the comments. This way, you can obtain a well-documented and self-explanatory document. You also have the great advantage of using Word to edit the layout of the report in a "what you see is what you get" approach.

Only comments with the user initials "sar" (in lowercase) are processed by Sar, so you can continue to use ordinary comments in your Word document (see Word documentation to learn how to change user initials).

This article is organized as follows. In section 2, I will explain how to install the Sar macro and how to enable the interprocess communication between Word and Stata. In section 3, I will show the use of Sar for the first time, with a simple but enlightening example. In section 4, I will explain how to set the format of numerical outputs obtained through Sar. In section 5, I will show how to create tables, with particular attention to tables of estimates from regression analysis. In section 6, I will explain how to automatically insert graphs—created in Stata—into the Word document. In section 7, I will present an advanced feature of Sar: the use of Sar programs. In section 8, I will describe how to use Sar in interactive mode, an alternative way to retrieve data from Stata. In section 9, I will explain the syntax of all the Sar commands and provide some

brief descriptions of them. In section 10, I will describe some limitations of Sar, and in section 11, I will close the article with my conclusions.

2 What you need to use Sar

Sar works only on Microsoft Word for Windows. Both Stata and Microsoft Word must be installed on your computer.

To use Sar, you need to download the Word macro-enabled template file `Stata automatic report 1.1.dotm` from http://www.stata-journal.com/software/sj13-1/pr0055/Stata automatic report 1.1.dotm and copy it into the Word Startup folder. This file contains the Sar 1.1 macro. To find the Startup folder path, open Word, click on **Customize Quick Access Toolbar** > **More Commands...** > **Advanced** in the Word options dialog window, and then click on the **File Locations...** button.

You also need to install the Stata Automation object. See http://www.stata.com/automation/, section 3.1, for instructions. See that same URL for general information about the Stata Automation object.

Once the macro has been copied into the Word Startup folder, make sure the macro was found and loaded by Word (please see the Word documentation). To easily run the Sar macro, you should create a button on the customizable Quick Access Toolbar of Word, as in figure 1. Do the following steps: 1) click on the arrow to the right of the icons on the Quick Access Toolbar and choose **More Commands...**; 2) select `Macros` from the *Choose commands from:* list; 3) select the `Stata Automatic Report` macro, then click on **Add**; and 4) click on the **Modify** button to assign an icon to the button and change its name to `Stata automatic report 1.1`.

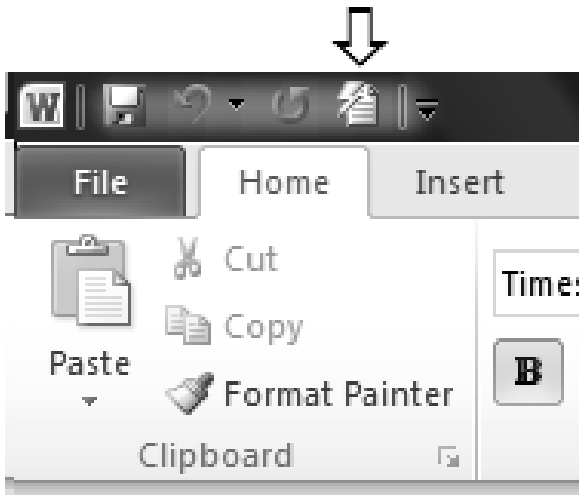

Figure 1. Customize the Quick Access Toolbar in Word

Optionally, you can create a keyboard shortcut to run the Sar macro. To create a keyboard shortcut, see the Word documentation.

3 A quick look at how Sar works

First of all, I will show you how Sar works with the help of a simple example. Suppose you want to create an automatic report where the mean of the `price` variable of `auto.dta` is discussed. You have to create a Word file where a portion of text, henceforth called "placeholder" text, is commented with a mix of Stata and Sar commands. I suggest you use the conventional character "X" as a general placeholder. After the Sar macro has been launched, the placeholder will be replaced by the numerical value obtained from processing the Stata and Sar commands. Before you execute Sar, the Word file should appear like that in figure 2.

Figure 2. A simple automatic report (before Sar is executed)

Notice that the comment in figure 2 is signed with the initials "sar" in lowercase (the 1 is automatically added by Word to designate this as the first comment by "sar"). Only comments with these initials will be processed by Sar, so you can continue to use ordinary comments when revising your document provided that you use different initials for them. To change the user initials, go to **File** > **Options** (figure 3).

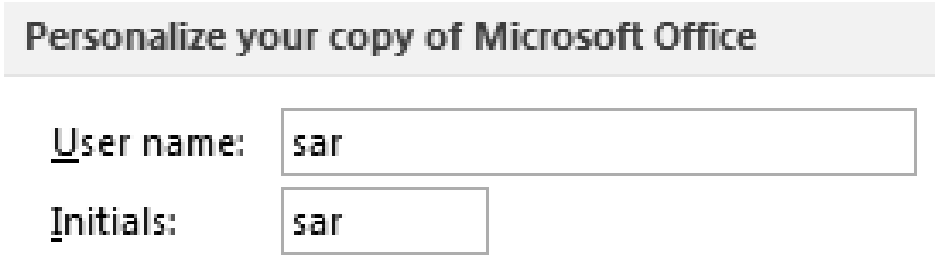

Figure 3. Setting user initials to work with Sar

The first two commands in the comment of figure 2 (`sysuse` and `summarize`) will be executed by Stata. They load `auto.dta` and calculate summary statistics for the `price` variable. The third command is the Sar `@print` command, which asks Word to replace the placeholder with the `r(mean)` value retrieved from the Stata environment. Telling the difference between Sar commands and Stata commands is easy: every Sar command begins with the `@` symbol.

After Sar has been executed, the report will appear as the one in figure 4.

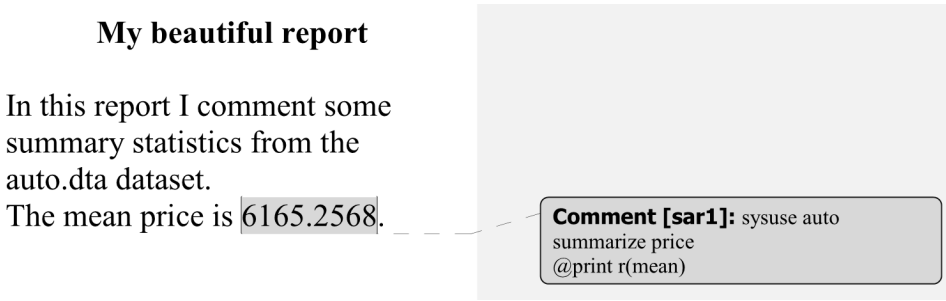

Figure 4. A simple automatic report (after Sar is executed)

The placeholder text was replaced by the value 6165.2568. If the value of the `price` variable changes, you can easily update your report simply by running Sar again. The comment will remember the statistical analysis you carried out, helping you to document your analysis and allowing other people to reproduce the results. Now your document is automated and self-explanatory.

4 Formatting numerical output

Of course, you would like to be able to format the numerical values retrieved from the Stata environment, for example, by choosing the number of digits after the decimal separator. You can do this by using the `@format` command. This command must be followed by a numeric format string as its unique argument, specified according to the same rules used in the Stata `format` command. For example, you can type `@format %6.1f` to set a numerical output with a period as the decimal separator and only one digit after the decimal separator. All numerical outputs will be formatted according to the numeric format specified in the command syntax, but the effect of the `@format` command is maintained until a new `@format` command specifies a different numeric format.

In the example in figure 5, the automatic report shows the average price of the cars in `auto.dta`, the number of observations, and the number of foreign cars.

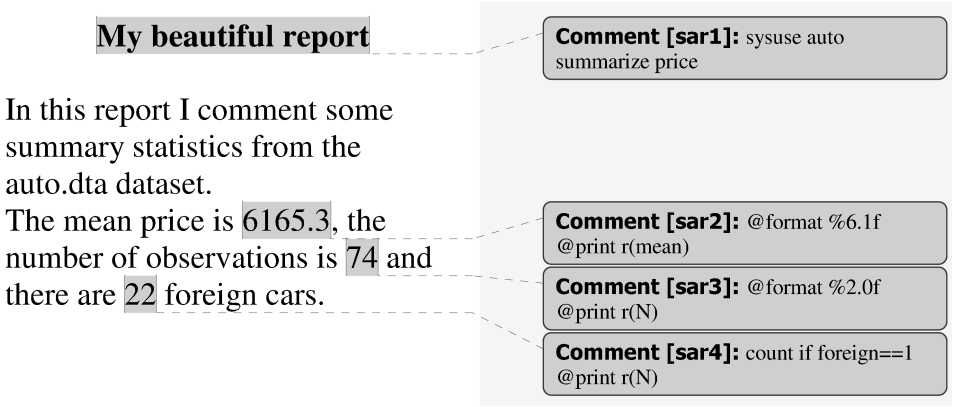

Figure 5. Using the `@format` command

Notice that the first Stata commands are placed in a comment associated with the title of the report, making the document easier to read. Stata and Sar commands can be placed in different comments, and they are always executed sequentially.

No leading spaces are added to the numerical output even if they are expected according to the Stata formatting rules. For example, in the second comment, the effect of `@format %6.1f` would be the same as that of `@format %21.1f`. This simplifies the editing work in Word.

Try to change the `@format` command in the second comment to `@format %6,1f` to obtain a numerical output with a comma as the decimal separator.

5 Creating tables

Most reports have statistical tables, and statisticians know how boring it can be to create them. Sar helps you to easily create tables, filling them with numerical values and labels. The basic commands to do this are `@filltable`, `@matrixrownames`, and `@matrixcolnames`.

The `@filltable` command must always be inserted into a comment associated with a table. In its simplest syntax, the command requires three arguments (there are five in its complete syntax). The first is the data, retrieved from the Stata environment, that you want to insert into the table; the second and the third, respectively, are the table row and column from which the filling process starts.

A well-formatted table needs row and column labels, too. Results stored in a matrix with row and column names are often obtained in Stata; row and column names are nat-

ural candidates to use as labels for your table. You can automatically insert these names into your table with the `@matrixrownames` (for row names) and `@matrixcolnames` (for column names) commands. Similarly to the `@filltable` command, in the simplest syntax for `@matrixrownames` and `@matrixcolnames`, the first argument is a matrix (but never a scalar) and the second and the third arguments, respectively, are the row and the column of the first table cell that has to be filled with the names of the matrix.

In figure 6, we use the `@filltable` and `@matrixrownames` commands to create a table presenting results from a regression analysis. First of all, we obtain the `beta` matrix of estimated regression coefficients as the transposed matrix[1] of `e(b)` in the first Sar comment. The `@filltable` command in figure 6 has the column vector `beta` as its first argument, the number `2` as its second argument, and the number `2` as its third argument: the command prints the matrix `beta` starting from the table cell in the second row and second column. The `@matrixrownames beta 2 1` command prints the row names of the matrix `beta` starting from the table cell in the second row and first column.

The Word table associated with the `@filltable` and `@matrixrownames` commands needs to be big enough to contain the vector that the commands insert into it—Sar will not automatically add the needed rows and columns to the table.

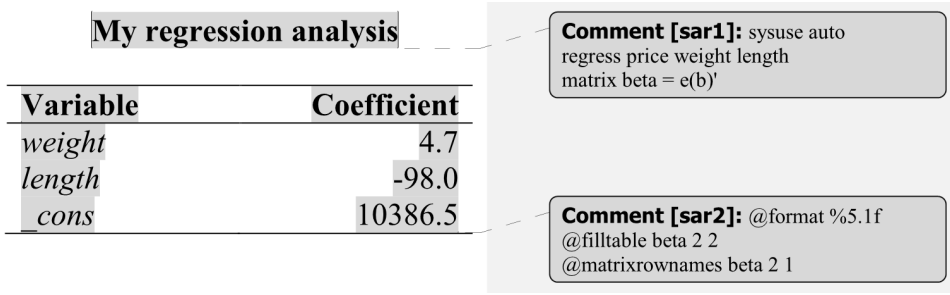

Figure 6. Using the `@filltable` and `@matrixrownames` commands

Now we want to add standard errors in parentheses under each estimated coefficient. To do this, we have to replace our 4×5 table with a 7×5 table (we add three rows, one for each standard error). To obtain the final table (shown in figure 7), we need to alternate estimated coefficients and standard errors in the second column of our new table and alternate row names with white cells in the first column.

The `@filltable` command has a total of five arguments (the last two are optional). So far, I have explained the first three. The fourth and the fifth arguments are called, respectively, row step and column step: they set how many rows (columns) have to be skipped between a row (column) and the next row (column) when you write the matrix

1. Note that the apex used as operator of transposition (') is not the one directly obtained by trying to type it in Word (‘). If you want to get the correct apex after typing the wrong one in Word, you have to press *Ctrl+Z*.

in the table. Their default values are (`0`, `0`), for which no row or no column is skipped. In the first `@filltable` command in figure 7, we use a row step of `1` and a column step of `0` (this last has no effect in this example because `beta` is a column vector); so the coefficients are printed in the table skipping one row between values. The same row step and column step are used for the second `@filltable` command, which prints the standard errors (the standard error vector `sd` is obtained in the first Sar comment by using two lines of Mata code). A row step of `1` is also specified as the fourth argument of the `@matrixrownames` command, which takes only four arguments (a column step argument would be useless).

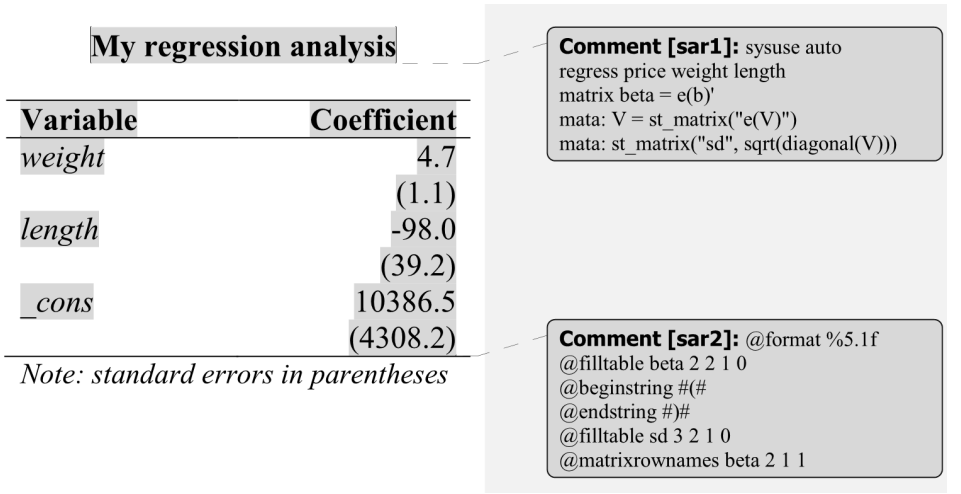

Figure 7. A table of regression results with beta coefficients and standard errors[2]

In figure 7, standard errors are listed in parentheses, which we obtain using the `@beginstring` and `@endstring` commands. Each of these commands takes only one argument given by a string delimited by two sharps (#). The `@beginstring` command sets the string you want to put before numerical output printed in tables, whereas the `@endstring` command sets the string you want to place after numerical output. In our example, we use an opening parenthesis as the opening string and a closing parenthesis as the closing string. The effect of these commands persists until the `@resetstring` command (with no arguments) is used.

2. Note that the quotation marks used in the first Sar comment are " and not “. If you try to type them from your keyboard, Word will automatically create the latter. To obtain the correct one, press *Ctrl+Z* after you have typed the wrong quotation marks in Word.

6 Graphics

Sar can retrieve not only numbers but also graphs created in Stata. For this purpose, you can use the `@graph` command, which allows you to insert into Word the last graph displayed in Stata. The `@graph` command is typically used immediately after a Stata `graph` command, such as `graph twoway`, `graph matrix`, or `graph bar`.

The `@graph` command must be associated with a placeholder, which can be text or an image (but not a table). Similarly to the `@print` command, the placeholder will be replaced by what is retrieved from Stata, which in the case of the `@graph` command is an image. From a technical point of view, the graph is exported by Stata into a temporary file in your system temporary folder, and then the image is inserted into Word (the name of the temporary file is `sartemp`, with the extension depending on the file format chosen in the `@graph` command). This data exchange process is hidden to the user.

In the example in figure 8, we create an ordinary graph in Stata by using the `twoway` command. This command is immediately followed by the Sar `@graph` command, which will retrieve the last displayed graph and insert it into Word. The `@graph` command requires three arguments. The first is the image file format used in the data exchange process between Stata and Word; the format must be one of the following: `.eps`, `.png`, `.tif`, `.wmf`, or `.emf`. In the example, the file format is `.tif`, a raster (not vector) file format. The choice of the format can affect the visual appearance of the image. The second and the third arguments, respectively, are the width and the height of the image as it will be displayed in Word.

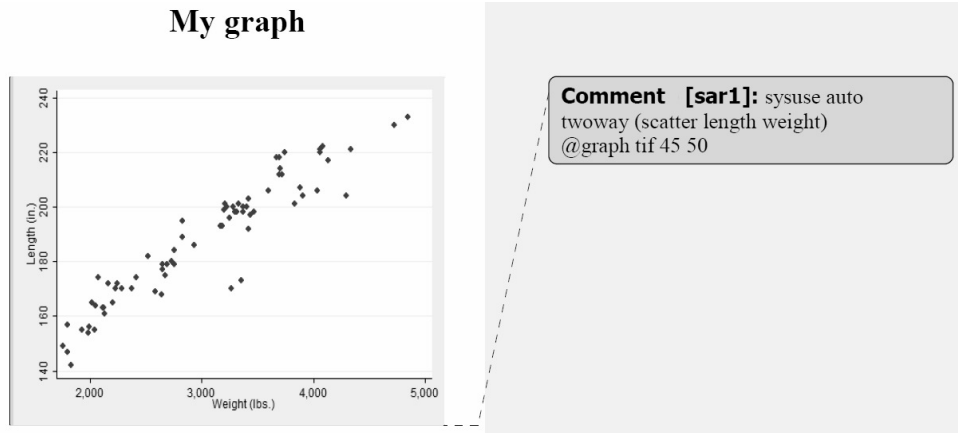

Figure 8. A graph inserted into Word by Sar

The `@graph` command also can be used after a Stata `graph use` command. For example, in the following Sar and Stata code,

```
twoway (scatter length weight)
graph save scatter1.gph, replace
twoway (scatter price mpg)
graph save scatter2.gph, replace
graph use scatter1.gph
@graph tif 100 100
```

the graph that will be inserted into Word is `scatter1.gph` and not the last-created `scatter2.gph`.

7 Sar programs

A Sar program is, roughly speaking, a list of Sar and Stata commands. The code of a Sar program is defined between the `@program` and the `@end` commands, that is,

`@program myprog`
[...]
[*my_commands*]
[...]
`@end`

The first argument of the `@program` command is used to specify the name of the Sar program. In the example above, the name of the Sar program is `myprog`. This program can be executed simply by using the `@do` command with the program name as its first argument:

```
@do myprog
```

A program can be defined in a Word comment or in an external plain text file, called library in the Sar jargon. A library can contain many Sar programs. To load all the Sar programs defined in a library, you can use the `@loadlibrary` command with the file path of the library file in quotation marks as its unique argument. Here is an example:

```
@loadlibrary "C:\sar libraries\mylibrary.txt"
```

Programs can accept arguments. Arguments have to be specified in the `@program` command. For example, the following program,

```
@program outmatrix matrix
@matrixrownames §matrix§ 2 1
@matrixcolnames §matrix§ 1 2
@format %4.3f
@filltable §matrix§ 2 2
@end
```

has a unique argument labeled `matrix` and prints the user-specified matrix in a Word table. In the program code, each argument is written between two § symbols.[2] Before executing the program, Sar replaces the §`matrix`§ string with the argument provided by the user. We use the `outmatrix` Sar program in figure 9.

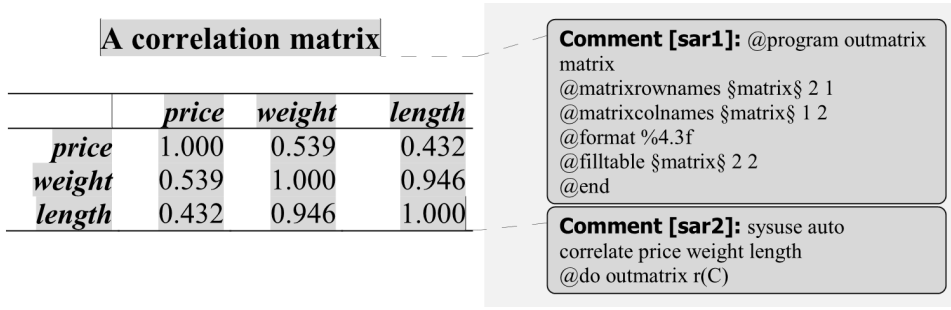

Figure 9. Using a Sar program and defining it in a Word comment

The `@do` command has always the Sar program name as its first argument. If the program requires some arguments, they must be included as arguments in the `@do` command. For example, in figure 9, the `@do` command calls the `outmatrix` Sar program with `r(C)` as an argument.

If the `outmatrix` program is defined in a library, it will have to be loaded through the `@loadlibrary` command, as it is done in figure 10.

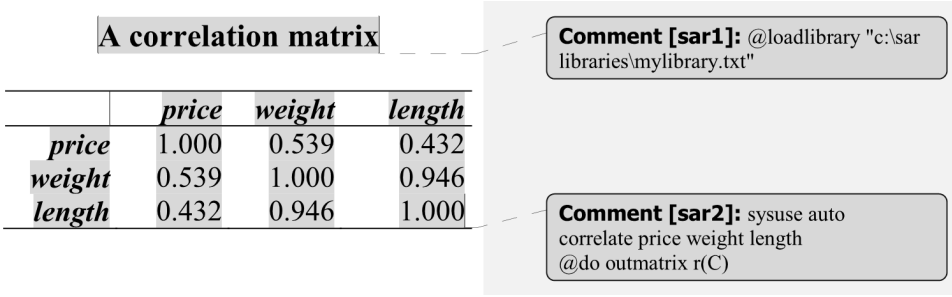

Figure 10. Loading a Sar program from a library

Using programs and libraries, the Sar code in the automatic report tends not to be verbose; it is very easy to understand what the code does. Moreover, a library can be reused and shared with colleagues.

2. To access the § symbol in Word, click **Insert** > **Symbol** > **More Symbols...**. In the Symbol dialog box, you will find the section sign symbol (§).

The following program produces a regression output with beta coefficients, standard errors, number of observations, and R-squared:

```
@program regressout
matrix beta = e(b)´
mata: V = st_matrix("e(V)")
mata: V = st_matrix("sd", sqrt(diagonal(V)))
@format %10.1f
@filltable beta 2 2 1 0
@matrixrownames beta 2 1 1
@beginstring #(#
@endstring #)#
@filltable sd 3 2 1 0
@resetstring
@format %3.0f
@filltable e(N) -2 2
@format %4.3f
@filltable e(r2) -1 2
@end
```

An example of its usage is given in figure 11. Because the `@print` command cannot be used to insert numerical values into a table, we print the number of observations and the R-squared by using the `@filltable` command. The last two `@filltable` commands in the `regressout` program have a negative number as the first argument. Negative numbers are used to indicate row and column coordinates according to a different coordinate system, where `-1` is the last row or column, `-2` is the second-to-last row or column, and so on. We use negative numbers to make the `regressout` program flexible and easy to use. This way you can use the program without being worried about how many variables you use in the regression analysis: the number of observations and the R-squared will always be printed, respectively, in the second-to-last row and in the last row of the table.

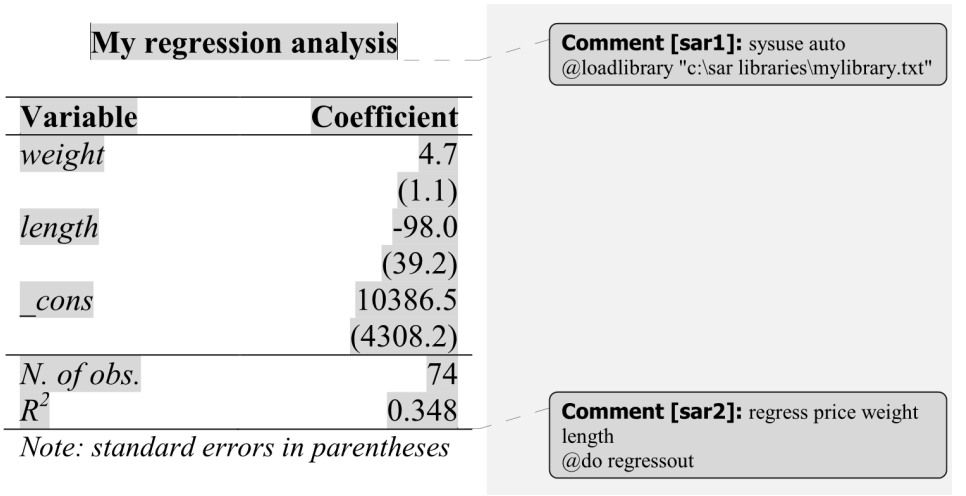

Figure 11. Output of a regression analysis with the `regressout` program

If you like the `regressout` Sar program, you can save it in a library and use it in all your automatic reports simply by loading the library with the `@loadlibrary` command. You can also create other Sar programs to carry out specific formatting tasks.

A Sar program cannot contain the definition of another program. Nested definitions of programs like the following are forbidden:

```
@program alpha
[ do_something_in_alpha ]
@program beta
[ do_something_in_beta ]
@end
[ do_something_in_alpha ]
@end
```

A Sar program cannot call another program with the `@do` command or load programs through the `@loadlibrary` command.

8 Using Sar in interactive mode

Suppose you do not have any need to create a document that can update itself if data change, and you like to launch Stata commands directly from Stata because you consider it a more comfortable environment. In such a case, you could consider using the `@interact` command, which allows you to halt Sar, open a Stata session in which you can interactively launch your commands, and return to Sar to use the obtained results in your document.

To use Sar in interactive mode, place the `@interact` command in a Word comment (the command does not require arguments). The elaborations managed by Sar will halt when the `@interact` command is found. At that moment, Sar will open the Stata software window, which remains at your disposal, to allow you to type your commands and interact with Stata. After you conclude your Stata session, remember not to manually close the Stata window: this will cause Sar to crash. To avoid this, you have to return to Word, where you will find a dialog window with a button to close Stata. If you correctly do this, the data objects created in the Stata session will be available for the Sar commands following the `@interact` command.

The `@interact` command can be useful when you are not sure about how to do a certain statistical analysis, so you need to interact with Stata to check intermediate results or maybe call help. Suppose that you have to obtain the well-known $X'X$ matrix required in regression analysis from a dataset given by X (the last column is a column vector of ones). You do not mind documenting your statistical analysis in a Word comment, so you choose to use the `@interact` command as shown in figure 12.

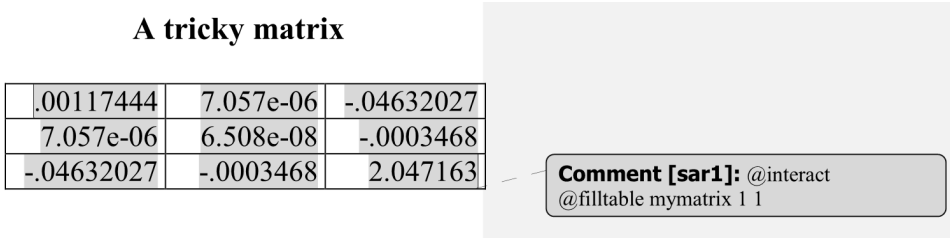

Figure 12. Using Sar in interactive mode

When this example is launched and Sar finds the `@interact` command, the Stata window will open and there you can type the following:

```
. sysuse auto
(1978 Automobile Data)
* suppose you do not remember how to use the mkmat command
. help mkmat
. mkmat mpg weight, matrix(X)
. count
  74
. matrix one = J(74, 1, 1)
. matrix X = X, one
. matrix mymatrix = invsym(X´ * X)
```

After typing these commands, remember to return to Word, where you will find a dialog window with a button named **Close Stata** that you must click on. If you try to manually close Stata, Sar will crash, and you will have to retype all your commands.

The matrix `mymatrix`, created in the previous Stata session, is now available in the Sar environment. The `@filltable` command in figure 12 retrieves the values of this matrix and uses them to fill the final table. If you are no longer interested in the Word comment, you can then delete it.

9 Syntax and description of the Sar commands

This section gives the syntax and a short description of each Sar command. In the given syntax of a Sar command, arguments between brackets are optional. Examples are shown only for commands with arguments.

9.1 @beginstring and @endstring

Syntax

`@beginstring #`*string*`#`

`@endstring #`*string*`#`

Description

The `@beginstring` command sets the string of characters you want to place before the numerical output of the `@filltable` command. The `@endstring` command sets the string of characters you want to place after the numerical output of the `@filltable` command. The *string* must be specified between two sharps (`#`).

Examples

The following code ensures that every numerical value inserted into a table by the `@filltable` command will begin and end with an open bracket:

```
@beginstring #[#
@endstring #]#
```

The following code cancels the effect of the previous code:

```
@beginstring ##
```

9.2 @cleartable

Syntax

`@cleartable`

Description

The `@cleartable` command clears the table associated with the comment where the command is written. It can only be used within Word comments associated with a single table.

The command has no arguments.

9.3 @do

Syntax

`@do` *SarProgram* [*arg1 arg2 arg3 ... argN*]

Description

The `@do` command executes a program previously loaded by the `@loadlibrary` command or defined in a Word comment through the `@program` and `@end` paradigm.

SarProgram specifies the program to be executed.

The optional arguments *arg1*, *arg2*, *arg3*, ..., *argN* specify the arguments to be passed to the program.

Example

Suppose you have defined a Sar program in a comment and you have called it `myprogram`. If you want to execute it, you can use the following code:

```
@do myprogram
```

9.4 @filltable

Syntax

`@filltable` *StataData startingRow startingCol* [*rowStep colStep*]

Description

The `@filltable` command inserts matrices, Stata results, scalars, and macros given by the *StataData* argument into a Word table. It can be used only in Word comments associated with a single table.

StataData is the data retrieved from the Stata environment used by the command to fill the table. It can be a matrix, a Stata result, a scalar, or a macro.

startingRow and *startingCol* indicate, respectively, the row and the column of the table cell from which *StataData* should begin to be printed. They have to be nonzero integers. If these values are negative, `-1` means the last row or column, `-2` means the second-to-last row or second-to-last column, and so on.

rowStep (*colStep*) indicates how many rows (columns) have to be skipped between a row (column) and the next row (column) while you are filling the table. When *rowStep* or *colStep* equals `0`, no blank row or column is left between printed rows or columns. When *rowStep* or *colStep* equals n, then n blank rows or columns are left between printed rows or columns. These arguments are optional, and they have to be nonnegative integers.

Examples

The following code,

```
@filltable e(V) 4 6 3 1
```

fills a Word table with values taken from the matrix `e(V)` starting from the position in row `4` and column `6`. The row step is `3` and the column step is `1`, so the command will leave three blank rows between printed rows and one blank column between printed columns.

In the following code,

```
@filltable e(V) 4 6
```

the row step and the column step are not set, so the `@filltable` command will consider the default value of `0` for both arguments. No blank rows (columns) will be left between printed rows (columns).

9.5 @format

Syntax

`@format` %*fmt*

Description

The `@format` command sets the numerical format of the output obtained by the `@print` and `@filltable` commands. The set numerical format is preserved for the following `@print` and `@filltable` commands.

The %*fmt* argument has to be a numerical format written with the same rules used in the Stata `format` command (see [D] **format**).

Example

The following code sets a format with three decimal digits and a comma as decimal separator:

```
@format %5,3f
```

9.6 @graph

Syntax

`@graph` *graphicformat width height*

Description

The `@graph` command retrieves the last graph displayed in Stata and inserts it into the Word document. The command must be associated with a placeholder, which can be text or an image but not a table. The placeholder will be replaced by the last graph displayed in Stata. The `@graph` command is typically used immediately after a Stata graph command, such as `graph twoway`, `graph matrix`, or `graph bar`.

The *graphicformat* argument is the graphic format of the image exported by Stata and imported by Sar during the data exchange process between the two softwares. It can be one of the following: `.eps`, `.png`, `.tif`, `.wmf`, or `.emf`. The choice of the format can affect the visual appearance of the image. The *width* and the *height* arguments represent the width and the height of the graph displayed in the Word document.

All arguments must be provided.

Example

The following code inserts a scatterplot into Word with a width of 100 and a height of 80, using the `.png` file format:

```
sysuse auto
twoway (scatter length weight)
@graph png 100 80
```

9.7 @interact

Syntax

`@interact`

Description

The `@interact` command halts the execution of Sar to put Stata at your disposal. You can use Stata, interact with it, and create data objects (such as scalars or matrices) that will be available in the Sar environment after your Stata session has been closed. Remember not to manually close the Stata window: this will cause Sar to crash. You have to return to Word, where you will find a dialog window with a button to close Stata.

The command has no arguments.

9.8 @loadlibrary

Syntax

`@loadlibrary "`*LibraryFilePath*`"`

Description

The `@loadlibrary` command loads programs defined in a Sar library file. The path of the Sar library file is specified in the *LibraryFilePath* argument.

Example

Suppose you wrote some Sar programs in a plain text file named `utilities.txt` in the `C:\Sar` folder. You can load all the Sar programs defined in the file by using the following code:

```
@loadlibrary "C:\Sar\utilities.txt"
```

9.9 @matrixcolnames and @matrixrownames

Syntax

`@matrixcolnames` *StataMatrix startingRow startingCol* [*colStep*]

`@matrixrownames` *StataMatrix startingRow startingCol* [*rowStep*]

Description

The `@matrixcolnames` and `@matrixrownames` commands fill a Word table with, respectively, column names and row names of a Stata matrix. They can be used only in Word comments associated with a single table.

StataMatrix is the matrix retrieved from the Stata environment whose matrix row names are printed by `@matrixrownames` and whose matrix column names are printed by `@matrixcolnames`. This argument has to be a matrix.

startingRow and *startingCol* indicate, respectively, the row and the column of the table cell from which the row names or column names of *StataMatrix* begin to be printed. They have to be nonzero integers. If these values are negative, `-1` will indicate the last row or column, `-2` will indicate the second-to-last row or column, and so on.

colStep is an optional argument for `@matrixcolnames`. It indicates the column step according to which the table is filled. The default value is `0`. It has to be a nonnegative integer.

rowStep is an optional argument for `@matrixrownames`. It indicates the row step according to which the table is filled. The default value is `0`. It has to be a nonnegative integer.

Examples

The following code writes row names of matrix `e(b)` in a table starting from row two and column three:

```
@matrixrownames e(b) 2 3
```

If you add the row step argument, the row names will be written leaving five blank rows between printed rows:

```
@matrixrownames e(b) 2 3 5
```

9.10 @print

Syntax

`@print` *StataValue*

Description

The `@print` command, launched from a Word comment associated with a portion of text (a temporary text placeholder in Sar jargon), replaces its placeholder with the value of a Stata result, scalar, or macro retrieved from the Stata environment. The `@print` command cannot be used in a Word comment associated with a table. The *StataValue* argument must be a Stata result, scalar, or macro.

Example

Suppose you have created a scalar named `myresult` in Stata. To retrieve the value of the scalar, you can type

```
@print myresult
```

in a comment associated with the text you want to replace with the value.

9.11 @program and @end

Syntax

`@program` *programName* [*arg1 arg2 ... argN*]
[...]
[*Sar and Stata commands*]
[...]
`@end`

Description

The `@program` and `@end` paradigm is used to define a Sar program. This paradigm can be used in a Word comment or in a Sar library. A Sar program is, roughly speaking, a list of Sar and Stata commands. This list of commands is defined between the `@program` and the `@end` commands. After the commands are loaded in the Sar environment, they can be executed through the `@do` command.

The *programName* argument sets the name of the program.

The optional arguments *arg1*, *arg2*, ..., *argN* specify the arguments of the program defined by the `@program` and `@end` paradigm. When you want to use the values passed as arguments in your program, you have to use the §*arg1*§, §*arg2*§, ..., §*argN*§ callbacks inside your program code; before executing the program, Sar replaces every callback with the corresponding values of arguments.

The `@end` command closes a program definition. It has no arguments.

The following commands cannot be used within a Sar program: `@do`, `@loadlibrary`, `@interact`, and the `@program` and `@end` paradigm.

Examples

Consider the following Sar program:

```
@program printTransposedMatrix matrix row col
matrix mymatrix = §matrix§´
@fillmatrix mymatrix §row§ §col§
@end
```

Suppose you call it as follows:

```
@do printTransposedMatrix e(b) 2 2
```

The commands executed by Sar, after the call done by `@do`, will be

```
matrix mymatrix = e(b)´
@fillmatrix mymatrix 2 2
```

9.12 @resetstring

Syntax

```
@resetstring
```

Description

The `@resetstring` command sets the string of characters coming before and after the numerical output of the `@print` and `@filltable` commands to an empty string. When the `@resetstring` command is used, no characters are added before or after the numerical output. It is equivalent to the commands `@beginstring ##` and `@endstring ##`.

The command has no arguments.

See also the `@beginstring` and `@endstring` syntax and description.

9.13 @viewlog

Syntax

```
@viewlog
```

Description

The `@viewlog` command asks Sar to leave the Stata window open after the Sar macro is executed. This can be used to look at the log created by Stata computations. When `@viewlog` is used in a Word comment, a dialog window is opened after the execution of the Sar macro, allowing you to close the Stata window and definitively terminate the Sar macro.

The command has no arguments.

10 Some limitations of Sar

Sar works only in Windows.

Because Sar does not use Stata Automation in asynchronous mode (for details, see http://www.stata.com/automation/), you cannot use the following Stata commands: `program define`, `while`, `forvalues`, `foreach`, and `input`. These commands can still be used inside do-files or ado-files. The `exit` command is ignored.

Sar internally uses two auxiliary Stata local macros to retrieve results from Stata. The local macro `stataAutomaticReportValue` is used to retrieve Stata results, scalars, and macros. The local macro `stataAutomaticReportMatrix` is used to retrieve matrices. You have to avoid the use of these two auxiliary local macros to prevent conflicts between Sar and your Stata code.

Whenever a command written in a Sar comment modifies the Word document (for example, filling a table with the `@filltable` command), the comment is deleted and afterward rebuilt with the same text. This process is hidden to the user. Here there is no room for me to explain the technical reason why this process is necessary. You should consider this way of working with Sar to use Sar properly; otherwise, there could be unwanted consequences in the document.

The maximum number of programs that can be loaded in the Sar environment is 500. The maximum number of arguments of a Sar program is 50. These limits are not due to particular technical reasons. I set them simply thinking it is unlikely they could be exceeded.

Word comments referring to the same portion of text cause the Sar macro to crash and cause unwanted consequences in the Word document, so you have to avoid them.

The setting of global and local macros has no effect in Sar. So the following Stata commands will cause an error:

```
. global mypi "3.14"
. scalar mydoublepi = $mypi * 2
```

After the execution of the Sar macro, you cannot erase all the previous changes carried out in the document by Sar with the Word undo function (*Ctrl+Z*). I strongly recommend that you save the Word document executing Sar to prevent unwanted consequences to your document.

The following commands cannot be used within a Sar program: `@do`, `@loadlibrary`, `@interact`, and the `@program` and `@end` paradigm.

11 Conclusions

Sar makes preparing statistical reports in Microsoft Word for Windows easy. Thanks to Sar, you can exploit all the functions of Microsoft Word—including its "what you see is what you get" approach—and the power of Stata as a computational engine. Sar allows the creation of an automatic report, that is, a Word document in which numerical values generated by statistical analyses in Stata are automatically inserted into the document. Sar provides two advantages: the automatic report does not have to be modified if data change, and the report is integrated with the documentation about the statistical analysis that has been carried out. Sar allows you to easily share the workflow of data analysis and encourages reproducible research.

Automation and self-explanatory documents can also be obtained through the package `Sweave` (Leisch 2002) for the R statistical software. `Sweave` is a tool that allows you to embed R code in LaTeX documents, and it is part of every basic R installation. Unlike Sar, `Sweave` requires competence in the use of LaTeX and does not offer a graphic user interface for document editing; it essentially adopts a "what you see is what you mean" approach.

Another approach to the automatic generation of documents is described by Gini and Pasquini (2006). The authors give examples of how to generate LaTeX and HTML documents from Stata by using the `file` suite of commands (see [P] **file**). This approach is very flexible but, unlike Sar, requires writing LaTeX or HTML code.

Sar is a versatile tool. If you use it in interactive mode, you will quickly retrieve numerical results from Stata, and you will not need to manually copy them into the word processor. You can use Sar to create webpages or other types of documents managed by Word. Automation capabilities of Sar can be exploited to write homework, schoolwork, and tests with different random numerical values (all with a different but known seed of a pseudorandom generation process to allow a quick marking of each test). Moreover, "quick parts" can be stored in Word to create reusable tables with associated Sar comments and inserted into the document when necessary; thus you avoid having to reinvent the wheel any time a new document has to be edited (see Word documentation to find out how to create "quick parts").

Sar is an extensible system. To solve complex formatting tasks, you can add new useful functions defining Sar programs. By calling Sar programs, you can make your code less verbose and easier to read.

Like all software, Sar can be improved. Some of its limitations have been discussed in the previous section. If you want to modify the Sar macro, right-click on the `Stata automatic report 1.1.dotm` file, choose **Open**, and edit the macro in the VBA integrated development environment available in Word (see Word documentation to find out how to use it). Mastery in the use of the VBA language and the Stata Automation object is required.

12 References

Gallup, J. L. 2012. A programmer's command to build formatted statistical tables. *Stata Journal* 12: 655–673.

Gini, R., and J. Pasquini. 2006. Automatic generation of documents. *Stata Journal* 6: 22–39.

Hardin, J. 1995. dm29: Create TeX tables from data. *Stata Technical Bulletin* 25: 3–7. Reprinted in *Stata Technical Bulletin Reprints*, vol. 5, pp. 20–25. College Station, TX: Stata Press.

Jann, B. 2005. Making regression tables from stored estimates. *Stata Journal* 5: 288–308.

———. 2007. Making regression tables simplified. *Stata Journal* 7: 227–244.

Knuth, D. E. 1986. *The TEXbook*. Reading, MA: Addison–Wesley.

Lamport, L. 1994. *LATEX: A Document Preparation System*. 2nd ed. Reading, MA: Addison–Wesley.

Leisch, F. 2002. Sweave: Dynamic generation of statistical reports using literate data analysis. In *COMPSTAT 2002*, ed. W. Härdle and B. Rönz, 575–580. Physica Verlag: Heidelberg.

Long, J. S. 2008. *The Workflow of Data Analysis Using Stata*. College Station, TX: Stata Press.

Newson, R. 2003. Confidence intervals and p-values for delivery to the end user. *Stata Journal* 3: 245–269.

About the author

Giovanni L. Lo Magno is a research fellow in statistics at the University of Palermo in Italy. His research interests focus on gender differences in employment and statistical softwares. He is the author of Tabula, a graphic user interface front-end for Stata.