The Stata Journal (*yyyy*) *vv*, Number *ii*

Identifying packages used in Stata code

Lydia Reiner
Independent researcher
TX, USA
lr397@cornell.edu

Lars Vilhuber
Cornell University
Ithaca, NY, USA
lars.vilhuber@cornell.edu

Abstract. Many researchers work on personal machines, and have accumulated many installed packages over the years. Reproducibility of code requires that all packages used within an analysis be listed. While the robust solution is to install packages once, in a project-specific directory, most users appear to use their system-wide installations to store packages. When the time comes to create a reproducible package, this package helps such users to identify the packages actually used in their code, by parsing the Stata code and listing the most likely packages being used. We outline two use cases for this package.

Keywords: st0001, packagesearch, reproducibility, replication package

1 Introduction

1.1 Background

Community-provided packages, published through this Journal, the Boston Statistical Software Components (`SSC`) archive (Cox 2010), and individually hosted "net installable" websites, are one of the strengths of Stata. They can be conveniently installed through point-and-click interfaces and Stata commands, and are readily available to augment the capabilities of Stata. In Dec 2021, the top 10 Stata packages on `SSC` had a total of 265953.5 downloads.

The inspiration for this package comes from research work under the AEA Data Editor, verifying countless Stata-based replication packages. Amongst replication packages submitted to the AEA journals, 73 percent use Stata at least in part, ahead of Matlab (22 percent, see Vilhuber et al. 2020).

Many fail to provide replication requirements, or said requirements are incomplete. This results in running Stata code without the proper packages installed, leading to a time-consuming process of code breaking, installing the necessary missing package, then restarting the process until all packages have been identified. With this project, we aim to identify the necessary Stata packages in provided .do files before code is run, saving time and frustration for authors and replicators alike.

More broadly, this could help mitigate small aspects of the replication crisis by allowing both authors and replicators identify packages used in code. It could be extra relevant in cases where provided code is unable to be run, such as when confidential data is used as an input.

1.2 Description

The `packagesearch` command has four basic components:

1. Collects and cleans list of all packages hosted at SSC (using the `ssc whatshot` command.

2. Parses each .do file in a specified directory (and subdirectories), then cleans and appends them. This step involves removing commented lines and collapsing the contents of each .do file into unique words.

3. Matches the parsed files against the list of existing SSC packages

4. Exports the results of the match. Results are ranked in terms of their popularity at SSC, with less popular packages having a higher probability of false positivity.

1.3 Further Discussion

The output of this command is a 'candidatepackages' file which gives a list of potential SSC packages required by the code based on the results of the match. If code was run, the user can then cross-reference the contents of this 'candidatepackages' file with the actual packages required by the code.

Currently, the process yields many more packages than are actually used by the code ("false positives") due to a variety of factors. This includes user-contributed packages that are built on top of an existing command (e.g- `bys`) , or packages with names that are commonly found in Stata code files but in other applications (e.g- `white, dash, cluster`).

As such, for each candidate package in the output file we give the probability of said package being a false positive. This probability is inversely related to the package's rank at SSC (i.e, the package's popularity). For example, a more popular package such as `estout` will have a much lower probability of false positivity than lesser known (and therefore lesser utilized) packages.

1.4 Limitations (Room for Improvement?)

As described above, the `packagesearch` command only scans .do files for packages found at SSC. As such, it currently cannot handle packages from Stata Journal or those obtained via `net install`. We gladly accept any contributions (Github repository linked here) to the project that may help expand the reach of the command.

We are currently running this command on any Stata-based replication package submitted to the AEA Data Editor, and using the results (both genuine packages found by the match and information on false positives) to further fine tune the process. We hope that in the future this will allow us to further refine the command and its functionality.

2 User's guide to stata.sty

`stata.sty` is a LaTeX package containing macros and environments to help authors produce documents containing Stata output and syntax diagrams.

2.1 Citing the Stata manuals

The macros for generating references to the Stata manuals are given in table 1.

Table 1: Stata manual references

Example	Result
\bayesref{bayes}	[BAYES] **bayes**
\cmref{cmchoiceset}	[CM] **cmchoiceset**
\dref{Data types}	[D] **Data types**
\dsgeref{dsge}	[DSGE] **dsge**
\ermref{eregress}	[ERM] **eregress**
\fnref{Statistical functions}	[FN] **Statistical functions**
\fmmref{fmm:~betareg}	[FMM] **fmm: betareg**
\grefa{Graph Editor}	[G-1] **Graph Editor**
\grefb{graph}	[G-2] **graph**
\grefci{line_options}	[G-3] ***line_options***
\grefdi{connectstyle}	[G-4] ***connectstyle***
\gsref{6~Using the Data Editor}	[GS] **6 Using the Data Editor**
\irtref{irt}	[IRT] **irt**
\lassoref{Lasso intro}	[LASSO] **Lasso intro**
\metaref{meta}	[META] **meta**
\meref{me}	[ME] **me**
\mreff{Intro}	[M-0] **Intro**
\mrefa{Ado}	[M-1] **Ado**
\mrefb{Declarations}	[M-2] **Declarations**
\mrefc{mata clear}	[M-3] **mata clear**
\mrefd{Matrix}	[M-4] **Matrix**
\mrefe{st_view($\,$)}	[M-5] **st_view( )**
\mrefg{Glossary}	[M-6] **Glossary**
\miref{mi impute}	[MI] **mi impute**
\mvref{cluster}	[MV] **cluster**
\pref{syntax}	[P] **syntax**
\pssrefa{Intro}	[PSS-1] **Intro**
\pssrefb{power}	[PSS-2] **power**
\pssrefc{ciwidth}	[PSS-3] **ciwidth**
\pssrefd{Unbalanced designs}	[PSS-4] **Unbalanced designs**
\pssrefe{Glossary}	[PSS-5] **Glossary**
\pssref{Subject and author index}	[PSS] **Subject and author index**
\rptref{Dynamic documents intro}	[RPT] **Dynamic documents intro**
\rref{regress}	[R] **regress**
\spref{Intro}	[SP] **Intro**
\stref{streg}	[ST] **streg**
\svyref{svy:~tabulate oneway}	[SVY] **svy: tabulate oneway**
\tsref{arima}	[TS] **arima**
\uref{1~Read this---it will help}	[U] **1 Read this—it will help**
\xtref{xtreg}	[XT] **xtreg**

2.2 Stata syntax

Here is an example syntax display:

regress *depvar* [*indepvars*] [*if*] [*in*] [*weight*] [, **noconstant** **hascons** **tsscons**
vce(*vcetype*) **level**(#) **beta** **eform**(*string*) **depname**(*varname*) *display_options*
noheader **notable** **plus** **mse1** **coeflegend**]

This syntax is generated by

```
\begin{stsyntax}
\dunderbar{reg}ress
    \depvar\
    \optindepvars\
    \optif\
    \optin\
    \optweight\
    \optional{,
    \underbar{nocons}tant
    \underbar{h}ascons
    tsscons
    vce({\it vcetype\/})
    \underbar{l}evel(\num)
    \underbar{b}eta
    \underbar{ef}orm(\ststring)
    \dunderbar{dep}name(\varname)
    {\it display\_options}
    \underbar{nohe}ader
    \underbar{notab}le
    plus
    \underbar{ms}e1
    \underbar{coefl}egend}
\end{stsyntax}
```

Each command should be formatted using a separate `stsyntax` environment. Table 2 contains an example of each syntax macro provided in `stata.sty`.

Table 2: Stata syntax elements

Macro	Result	Macro	Result
\LB	[	\ifexp	if
\RB	]	\optif	[*if*]
\varname	*varname*	\inrange	in
\optvarname	[*varname*]	\optin	[*in*]
\varlist	*varlist*	\eqexp	=*exp*
\optvarlist	[*varlist*]	\opteqexp	[=*exp*]
\newvarname	*newvar*	\byvarlist	by *varlist*:
\optnewvarname	[*newvar*]	\optby	[by *varlist*:]
\newvarlist	*newvarlist*	\optional{text}	[text]
\optnewvarlist	[*newvarlist*]	\optweight	[*weight*]
\depvar	*depvar*	\num	#
\optindepvars	[*indepvars*]	\ststring	*string*
\opttype	[*type*]		

\underbar is a standard macro that generates underlines. The \dunderbar macro from stata.sty generates the underlines for words with descenders. For example,

- {\tt \underbar{reg}ress} generates regress
- {\tt \dunderbar{reg}ress} generates regress

The plain TEX macros \it, \sl, and \tt are also available. \it should be used to denote "replaceable" words, such as *varname*. \sl can be used for emphasis but should not be overused. \tt should be used to denote words that are to be typed, such as command names.

When describing the options of a new command, the \hangpara and \morehang commands provide a means to reproduce a paragraph style similar to that of the Stata reference manuals. For example,

level(#) specifies the confidence level, as a percentage, for confidence intervals. The default is level(95) or as set by set level; see [U] **20.8 Specifying the width of confidence intervals**.

was generated by

```
\hangpara
{\tt level(\num)} specifies the confidence level, as a percentage,
for confidence intervals.  The default is {\tt level(95)} or as set by {\tt
set level}; see \uref{20.8~Specifying the width of confidence intervals}.
```

2.3 Stata output

When submitting *Stata Journal* articles that contain Stata output, also submit a do-file and all relevant datasets that reproduce the output (do not forget to set the random-number seed when doing simulations). Results should be reproducible. Begin examples by loading the data. Code should be written to respect a linesize of 80 characters. The following is an example of the `stlog` environment containing output from simple linear regression analysis on two variables in `auto.dta`:

```
. sysuse auto
(1978 Automobile Data)
. regress mpg weight
      Source |       SS       df       MS              Number of obs =      74
-------------+------------------------------           F(  1,    72) =  134.62
       Model |  1591.9902     1  1591.9902              Prob > F      =  0.0000
    Residual |  851.469256    72  11.8259619            R-squared     =  0.6515
-------------+------------------------------           Adj R-squared =  0.6467
       Total |  2443.45946    73  33.4720474            Root MSE      =  3.4389

------------------------------------------------------------------------------
         mpg |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
      weight |  -.0060087   .0005179   -11.60   0.000    -.0070411   -.0049763
       _cons |   39.44028   1.614003    24.44   0.000     36.22283    42.65774
------------------------------------------------------------------------------
```

The above listing was included using

```
\begin{stlog}
\input{output1.log.tex}\nullskip
\end{stlog}
```

where `output1.log.tex` is a Stata log file converted to include TEX macros by using the `sjlog` command (more on `sjlog` shortly). `\nullskip` adjusts the spacing around the log file.

On occasion, it is convenient (maybe even necessary) to be able to omit some of the output or let it spill onto the next page. Here is a listing containing the details of the following discussion:

```
\begin{stlog}
. sysuse auto
(1978 Automobile Data)
{\smallskip}
. regress mpg weight
{\smallskip}
\oom
{\smallskip}
\clearpage
\end{stlog}
```

The `\oom` macro creates a short message indicating omitted output in the following example, and the `\clearpage` macro inserts a page break.

```
. sysuse auto
(1978 Automobile Data)
. regress mpg weight
  (output omitted)
```

The output in `output1.log.tex` was generated from the following `output.do`:

```
* output.do
set more off
capture log close

sjlog using output1, replace
sysuse auto
regress mpg weight
sjlog close, replace

sort weight
predict yhat
set scheme sj
scatter mpg yhat weight, c(. l) s(x i)
graph export output1.eps, replace

exit
```

`output.do` generates a `.smcl` file, `.log` file, and `.log.tex` file using `sjlog`. The actual file used in the above listing was generated by

```
. sjlog type output.do
```

`sjlog.ado` is provided in the Stata package for `sjlatex`. `sjlog` is a Stata command that helps generate log output to be included in LaTeX documents using the `stlog` environment. If you have installed the `sjlatex` package, see the help file for `sjlog` for more details. The lines that make up the table output from `regress` are generated from line-drawing macros defined in `stata.sty`; these were macros written using some font metrics defined in `?`.

By default, `stlog` sets an 8-point font for the log. Use the `auto` option to turn this behavior off, allowing you to use the current font size, or change it by using `\fontsize{#}{#}\selectfont`. The call to `stlog` with the `auto` option looks like `\begin[auto]{stlog}`.

Here is an example where we are using a 12-point font.

```
. sjlog type output.do
```

2.4 About tables

Tables should be created using the standard LaTeX methods. See `?` for a discussion and examples. Tables should be included in the main text rather than at the end of the document. Tables should be called out in the text prior to appearance.

There are many user-written commands that produce LaTeX output, including tables. Christopher F. Baum has written `outtable`, a Stata command for creating LaTeX tables from Stata matrices. Ben Jann's well-known `estout` command can also produce LaTeX output. To find other user-written commands that produce LaTeX output, try

```
. net search latex
```

Tables with notes

Table 3 shows the order and format to use for notes to tables.

Table 3: Industrial clusters

China		United States	
Core of cluster	Size (in # of units)	Core of cluster	Size (in # of units)
Construction	28[a]	Public administration and defense; compulsory social security	30[b]
Food, beverages, and tobacco	3	Food, beverages, and tobacco	2
Textiles and textile products	2	Chemicals and chemical products	1
Chemicals and chemical products	1	Basic metals and fabricated metal	1
Transport equipment	1	Transport equipment	1
$L_a = 0.602$***		$L_a = 0.567$	
$L_w = 0.828$**		$L_w = 0.837$	
$L_m = 0.335$*		$L_m = 0.287$	
$K^* = 5$		$K^* = 5$	
$K = 35$		$K = 35$	

SOURCE: Pew Research Center.
NOTE: U.S. industrial clusters based on U.S. input–output flows of goods expressed in millions of dollars between 35 ISIC industries from the WIOD data. The minimum number of clusters `k()` was set equal to five. The algorithm returns L_a, L_w, and L_m, which refer to the average of the internal relative flows, the population-weighted average of the internal relative flows, and the minimum of the internal relative flows, respectively. K^* and K refer to the number of defined regional clusters and the number of distinct starting units, respectively.
[a] This note pertains only to row 1 column 2.
[b] This note pertains only to row 1 column 4.
*** denotes $p < 0.01$; ** denotes $p < 0.05$; * denotes $p < 0.1$.

Order of notes should be

1. source notes
2. notes applying to the whole table
3. notes applying to specific parts of the table

4. notes on significance levels

Special notes:

- Use `\centering` because the `center` environment adds unnecessary vertical spacing.
- Place the `\begin{threeparttable}` line above the caption.

Tables should be included in the main text rather than at the end of the document. Tables should be called out in the text prior to appearance.

2.5 Encapsulated PostScript (EPS)

You can include figures by using either `\includegraphics` or `\epsfig`.

```
\begin{figure}[h!]
\begin{center}
\includegraphics{eps/output1.eps}
\end{center}
\caption{Scatterplot with simple linear regression line}
\label{fig}
\end{figure}

\begin{figure}[h!]
\begin{center}
\epsfig{file=output1}
\end{center}
\caption{Scatterplot with simple linear regression line}
\label{fig}
\end{figure}
```

Figure 1 is included using `\epsfig` from the `epsfig` package.

The graph was generated by running `output.do`, the do-file given in section 2.3. The `epsfig` package is described in `?`.

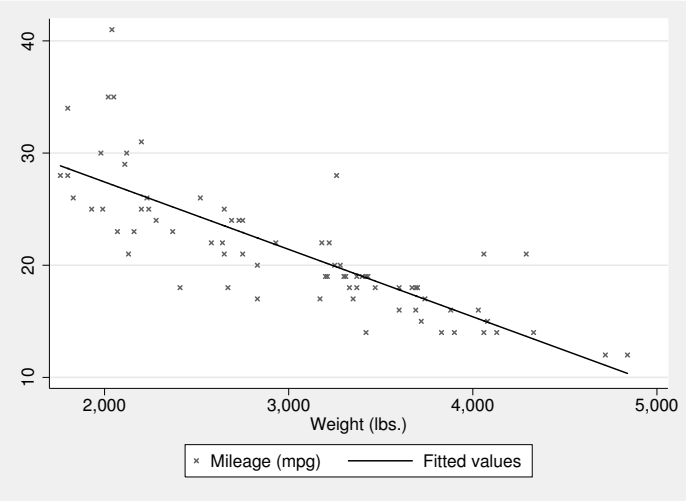

Figure 1: Scatterplot with simple linear regression line

EPS is the preferred format for graphs and line art. Figures should be included in the main text rather than at the end of the document and should be called out in the text prior to appearance. If your article is written in Word, you should submit your figures as separate EPS files. Rasterized-based files of at least 300 dpi (dots per inch) are acceptable. Avoid using bitmaps for figures and graphs, because even if images are outputted at 300 dpi, bitmaps can increase the size of the resulting file for printing. (However, bitmaps will be allowed for photographs, which are used in, for example, the *Stata Journal* Editors' prize announcement.) Images should be submitted in black and white (grayscale). We recommend that graphs created in Stata use the `sj` scheme.

2.6 Stored results

The `stresults` environment provides a table to describe the stored results of a Stata command. It consists of four columns: the first and third column are for Stata result identifiers (for example, `r(N)`, `e(cmd)`), and the second and fourth columns are for a brief description of the respective identifier. Each group of results is generated using the `\stresultsgroup` macro. The following is an example containing a brief description of the results that `regress` stored to `e()`:

Scalars

`e(N)`	number of observations	`e(F)`	F statistic
`e(mss)`	model sum of squares	`e(rmse)`	root mean squared error
`e(df_m)`	model degrees of freedom	`e(ll_r)`	log likelihood
`e(rss)`	residual sum of squares	`e(ll_r0)`	log likelihood, constant-only model
`e(df_r)`	residual degrees of freedom		
`e(r2)`	R-squared	`e(N_clust)`	number of clusters

Macros

`e(cmd)`	`regress`	`e(wexp)`	weight expression
`e(depvar)`	name of dependent variable	`e(clustvar)`	name of cluster variable
`e(model)`	`ols` or `iv`	`e(vcetype)`	title used to label Std. Err.
`e(wtype)`	weight type	`e(predict)`	program used to implement `predict`

Matrices

`e(b)`	coefficient vector	`e(V)`	variance–covariance matrix of the estimators

Functions

`e(sample)`	marks estimation sample

Alternatively, you can use the `stresults2` environment to create a two column table. This format works better if your descriptions are long.

2.7 Examples and notes

The following are environments for examples and notes similar to those given in the Stata reference manuals. They are generated using the `stexample` and `sttech` environments, respectively.

▷ Example

This is the default alignment for a Stata example.

◁

▷ Example

For this example, `\stexamplehskip` was set to `0.0pt` before beginning. This sentence is supposed to spill over to the next line, thus revealing that the first sentence was indented.

This sentence is supposed to show that new paragraphs are automatically indented (provided that `\parindent` is nonzero).

◁

❑ **Technical note**

For this note, `\sttechhskip` was set to `-13.90755pt` (the default) before beginning. This sentence is supposed to spill over to the next line, thus revealing that the first sentence was indented.

This sentence is supposed to show that new paragraphs are automatically indented (provided that `\parindent` is nonzero).

❑

2.8 Special characters

Table 4 contains macros that generate some useful characters in the typewriter (fixed width) font. The exceptions are `\stcaret` and `\sttilde`, which use the currently specified font; the strictly fixed-width versions are `\caret` and `\tytilde`, respectively.

Table 4: Special characters

Macro	Result	Macro	Result
`\stbackslash`	\	`\sttilde`	~
`\stforslash`	/	`\tytilde`	~
`\stcaret`	ˆ	`\lbr`	{
`\caret`	^	`\rbr`	}

2.9 Equations and formulas

In (1), $\overline{x}$ was generated using `\stbar{x}`. Here `\stbar` is equivalent to the TEX macro `\overline`.

$$E(\overline{x}) = \mu \tag{1}$$

In (2), $\widehat{\beta}$ was generated using `\sthat{\beta}`. Here `\sthat` is equivalent to the TEX macro `\widehat`.

$$V(\widehat{\beta}) = V\{(X'X)^{-1}X'y\} = (X'X)^{-1}X'V(y)X(X'X)^{-1} \tag{2}$$

Formulas should be defined and follow a concise style. Different disciplines adhere to different notation styles; however, if the notation cannot be clearly interpreted, you may be asked to make changes. The bolding and font selection guidelines are the following:

- Matrices are capitalized and bolded; for instance, $\mathbf{\Pi} + \mathbf{\Theta} + \mathbf{\Phi} - \mathbf{B}$.
- Vectors are lowercased and bolded; for instance, $\boldsymbol{\pi} + \boldsymbol{\theta} + \boldsymbol{\phi} - \mathbf{b}$.
- Scalars are lowercased and nonbolded; for instance, $r_2 + c_1 - c_2$.

Sentence punctuation should not be used in formulas set off from the text.

Formulas in line with the text should use the solidus (/) instead of a horizontal line for fractional terms.

Nesting of grouping is square brackets, curly braces, and then parentheses, or [{()}].

Only those equations explicitly referred to in the text should be assigned an equation number.

3 References

Cox, N. J. 2010. A Conversation with Kit Baum. *The Stata Journal* 10(1): 3–8. http://journals.sagepub.com/doi/10.1177/1536867X1001000102.

Vilhuber, L., J. Turitto, and K. Welch. 2020. Report by the AEA Data Editor. *AEA Papers and Proceedings* 110: 764–75.

About the authors

Lydia Reiner was an undergraduate at Cornell University studying economics when developing this package. She is now... (UPDATE).

Lars Vilhuber is an economist at Cornell University, and currently the Data Editor for the American Economic Association, responsible for verifying the computational reproducibility of manuscripts submitted to the AEA's various journals.

Author Contributions LR came up with the idea, did the research, implemented and tested the code, and wrote the manuscript.

LV assisted with the code, tested the code, and wrote the manuscript.

1 Computing domain-specific frequencies

Instead of using the output of `ssc whatshot` to classify strings into likely package names, we have also allowed for the use of domain-specific frequencies. Such frequencies need to come from a large corpus of verified code. In this appendix, we describe how we computed these frequencies for the economics domain (invoked through option `domain(econ)`).

As part of the AEA Data Editors work, students download replication packages and attempt to run the code in combination with the available data. They do so in a (nearly) clean environment, by invoking the following commands at the start of the command execution:

```
/* install any packages locally */
capture mkdir "$rootdir/ado"
sysdir set PERSONAL "$rootdir/ado/personal"
sysdir set PLUS     "$rootdir/ado/plus"
sysdir set SITE     "$rootdir/ado/site"
local ssc_packages "pkg1 pkg2"
if !missing("`ssc_packages´") {
        foreach pkg in `ssc_packages´ {
                capture which `pkg´
                if _rc == 111 {
                        dis "Installing `pkg´"
                        ssc install `pkg´, replace
                }
                which `pkg´
        }
}
```