The Stata Journal (*yyyy*) *vv*, Number *ii*, pp. 1–5

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 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.

Appendix

Appendix 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´
        }
}
```

These commands should ensure that all necessary packages must be installed into a project specific directory. In rare cases, replication packages may already include `ado` files, which should then be used. Once installed, the `ado` *directory* should also be committed to the git repository tracking the reproducibility check.[1] The resulting updated (private) repository thus contains authors' Stata code, and the replicator's installed `ado` files.

We thus obtained from our internal records the last known state of 665 repositories as of July 2021. Of these, 546 contained at least one Stata do-file. The average replication package had 25 do-files, and after verification had 32 ado-files. For each replication package, we ran (an earlier version of) the `packagesearch` command, and matched against the list of ado-files found. We counted as a `hit` a package that was confirmed to have the package installed, and counted the hits. We then ranked the list by hits. The resulting list is naturally somewhat different than the `whatshot` list, and in particular is pruned of spurious package names. Table 1 displays the top ten by the `domain(econ)` ranking, Table 2 displays the top ten packages per `whatshot` and their

1. Compliance with these instructions is not perfect.

equivalent `domain(econ)` ranking.

Table 1: Top ten `econ` packages.

Packagename	`econ`	`whatshot`
estout	1	2
reghdfe	2	5
outreg2	3	1
coefplot	4	13
ftools	5	6
ivreg2	6	8
ranktest	7	16
binscatter	8	24
spmap	9	25
unique	10	20

Generated on 2022-01-30.

Table 2: Top ten `whatshot` packages.

Packagename	`econ`	`whatshot`
outreg2	3	1
estout	1	2
asdoc		3
winsor2	51	4
reghdfe	2	5
ftools	5	6
logout	119	7
ivreg2	6	8
ivreg29		9
ivreg210		10

Generated on 2022-01-30.