The Stata Journal (2018/08/08 Draft) *vv*, Number *ii*

doit, dtait, editit, and more. Utilities to speed up working from the Command window.

Draft – Comments Welcome

Scott Long
Indiana University
Bloomington, IN / USA
jslong@iu.edu

2018-08-08

Abstract. Commands are introduced that make it faster to work from the Command window. These command lists files in the Results window. Clicking on the name of a file will execute a command that operates on that file. `doit` lists do-files directory than can be run by clicking; `editit` opens a do-file in the Do-file Editor. `dtait` lists datasets that are loaded by clicking, while `browseit` opens the dataset in browse mode in the Data Editor. The second part of this note explains how you can write similar commands.

Acknowledgements xxx.

Keywords: st0001, `workingdir` package, `adoit`, `browseit`, `doit`, `dtait`, `editit`, working directory, Command window

Note To install the commands in this article, `search workingdir` and follow the link.

I usually work on multiple do-files at the same time. I might be developing a series of programs to create a new dataset or several do-files for the analysis in a paper. When teaching, I have dozens of examples that I might use to illustrate some point or to respond to a student's question. When consulting, I ask people to deposit all of their do-files and datasets in a shared folder in the cloud.

I can execute a do-file in several ways. I can `do` *filename* from the Command window, which requires me to remember and type the file name. I can use the menu (alt-File, Do...) to submit the file, but unfortunately the menu does not open in the current working directory. Or, I can load the file into Stata's Do-file Editor and run it from there. After a do-file has been run, I use the Review

window to re-run the program or PageUp to insert prior commands to run the program into the Command window. Unfortunately, the history in the Review window disappears when I exist Stata.

To speed up my workflow, I wrote the `doit` command that lists the do-files in the working directory. If I click on a file name, the program run. For example, entering `doit` displays:

```
. doit
russia1-controls.do
russia2-scalebin.do
russia3-scaleord.do
russia4-analysisvars.do
```

where the names of the do-files are shown in blue. If I click on a name, say `doit-russia2-binary.do`, Stata executes the command:

```
do russia2-binary.do
```

I can limited the files that are listed. For example,

```
. doit *scale*
russia2-scalebin.do
russia3-scaleord.do
```

When teaching this makes it easy to find the example I want. For example, `doit *blm*` lists only examples for the binary logit model.

I found `doit` to be so handy, I extended the idea with several other `*it` commands.

`browseit` lists datasets in the working directory. Clicking on a dataset name opens the dataset for browsing in the the Data Editor.

`dtait` lists datasets in the working directory. Clicking on a dataset names submits the command `use` *filename*`, clear`.

`editit` lists do-files and ado files in the working directory. Clicking on a name, submits the command `doedit` *filename*. This makes it easy to get back to work when I return to a project.

When writing new commands (i.e., ado files), I use:

`adoit` clears programs from memory and lists ado files. Clicking on a name clear memory of programs and runs the program.

Like `dtait`, these commands let you to use a search string that controls the files that are listed. For example, `editit logit*.do` only lists the do-files with names that start with `logit`.

I find these simple utilities are great for sophisticated users (e.g., me), but are also great for beginners. In a teaching lab, each person's working directory contains the of do-files used in exercises. First time users have no trouble entering `editit` and clicking on the file to load it to the editor.

To install these commands, run `search workingdir` and clicking to install.[1] The commands are so simple that they do not have help files.

Writing point and click commands

Programmers might want to learn how these commands work so they can customize them or create other commands. Here is the code for `doit`:

```
 1:  program define doit
 2:      version 13.0

 3:      syntax [ anything ]

 4:      local filespec : subinstr local anything ".do" "", all
 5:      if "‘filespec’"==""  local filespec "*"

 6:      local filelist : dir "‘c(pwd)’" files "‘filespec’.do"

 7:      foreach file in ‘filelist’ {
 8:          display ‘"{stata do ‘file’:‘file’}"’
 9:      }

10:  end
```

Line 1 starts the program named `doit` that is ended on line 10. Line 2 indicates that the program was written to run in Stata 13 or later. Line 3 lets you can type anything after the command name. What you type is assigned to the local `anything`. For example, `doit *scale*` assigns `*scale*` to the local `anything`. Line 4 removes `.do` from the local `anything` to create the local

1. Other commands for changing the working directory are also installed.

`filespec` with my file specification. I remove `.do` so that you can enter a file specification with or without the suffix `.do`. For example, I list the do-files with `scale` in the name with either `doit *scale*` or `doit *scale*.do`. Line 5 says that if you do not provide a file specification, use the specification is `*` so that all files are listed. Line 6 is an extended macro function that creates the local `filelist` with names of the files in the working directory (whose name is in the system local `c(pwd)`) that satisfy the file specification.

Lines 7 through 9 loop through the list of file names. Line 7 creates the local `file` with the name of the current file. Line 8 allows you to click on the file name to execute the do-file. Fortunately, this command is easier to use than to explain! (`help smcl` for a lot more information). The `display` command sends to the screen the results of the SMCL (Stata Markup Control Language) directive contained within the braces `{` and `}`. The `stata` directive has two parts. Part 1, between `stata` and `:`, is a Stata command. Part 2, between `:` and `}`, is displayed in blue in the Results window. If you click on this displayed text, the command in part 1 is executed. In this case, clicking on the blue `russia2-scalebin.do` executes `do russia2-scalebin.do`. If line 8 is replaced by `display ‘"{stata doedit ‘file’:‘file’}"’`, then clicking on `russia2-scalebin.do` open that file in the Do-file Editor. While the `stata` directive only executes a single command, this does not limit what you can do when you click on a name. For example, the `browseit` command uses `display ‘"stata usebrowse ‘file’:‘file’"’` to run the command `usebrowse` (defined within `usebrowse.ado`) that first loads a dataset and then opens the Data Editor in browse mode.

Finally, if you want to create your own commands, or modify the versions I wrote, the easiest thing is to put the files in your PERSONAL directory. Run `adopath` to find out where that directory is located.

About the authors

Scott Long started writing these commands while teaching a course on reproducible results at the ICPSR Summer Program.