The Stata Journal (*yyyy*) *vv*, Number *ii*, pp. 1–12

A Workflow for Managing Projects

Preliminary Draft – Comments Welcome

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

2017-08-19

Abstract. An effective way to manage multiple projects is to create a different working directory for each project. This article shows you how to easily create commands to quickly change among working directories. Optionally, these commands create globals to customize the environment for each project and to automatically run other commands when the project is loaded.

Note To install the commands described below, `search workflow` and follow the links to install the commands. As of 2017-08-19, the commands you download are named `cdsave1`, `cdlist1`, `cdinfo1`, and `cddelete1`.

Keywords: st0001, `cdsave`, `cdlist`, `cdinfo`, `cddelete`, `cd`, working directory, managing projects

Unless you work on only one project, having separate working directories for each project is critical for an effective workflow. For many people, changing the working directory is the first thing done after entering Stata. Sadly, Stata does not provide an efficient way to move among working directories. You either type a possibly lengthy path specification or tediously click through your directory structure to select your working directory. With Stata for Mac, Stata opens in the working directory you were using when you last exited Stata. In Windows and Unix, the directory selected at installation is used with no simple way to change the default. The `workflow` package lets you create commands to quickly changing your working directory and lets you set up other characteristics associated with each working directory.

The `workflow` package is organized around the idea of a project. A project is a distinct activity such as a writing a paper, taking or teaching a class, writing a book, or developing a Stata command. Most simply, the `workflow` package creates commands that let you quickly change working directories as your move from one project to another. It also lets you easily change the default working directory. For many users, this will be the only features of the commands of interest. For those doing more complex work, however, you can create an environment for each project where globals are defined that specify a path where datasets are located, associate an URL with your project, and automatically run commands each time a project is started.

The `workflow` package includes four commands:

`cdsave` creates commands that change the working directory, create globals with information about each project, and optionally run additional commands when projects are switch. These commands are named `cd`*project*. To make these commands available regardless of your current working directory, they are saved in your PERSONAL directory.

`cdlists` lists your `cd`*project* commands along a note describing the project or the name of the working directory that is set. You can click on a command name to change working directories.

`cdinfo` lists the globals created by the last `cd`*project* command that was run.

`cddrop` drops `cd`*project* commands.

Non-programmers can use these commands without knowing anything about Stata programming. Programmers can easily customize the commands to add other features.

The `workflow` package was inspired by the `fastcd` package by Winters (2002), Baum's ideas about creating globals for setting up the environment for a research project (Baum 2016, 78-79), and Long's ado files for changing the working directories (Long 2009, 111-112,117-118).

1 Using the workflow commands

I begin by illustrating how the `workflow` package lets you change working directories. Next, I show you how to modify your `profile.do` to control the working directory in which Stata opens. Next, I explain how to create robust and portable do-files that use globals to specify the location of datasets. Finally, I explain how commands can be automatically executed when you change projects.

1.1 Quickly changing the working directory

A simple example explains how to create `cd`*project* commands to move among directories. Suppose that project A uses working directory `d:/projecta/work`, while project B uses `d:/projectb/work`.[1] To create the command `cdprojecta` to change the working directory to `d:/projecta/work`, I begin by changing my working directory to that location,

```
. cd d:/projecta/work
d:/projecta/work
```

Next, `cdsave` creates a command named `cdprojecta`:

```
. cdsave projecta
command cdprojecta saved in your personal directory
```

1. I use forward slashes rather than a backslashes since these works in all operating systems.

The command is saved in my personal directory. If you want to know where that is, run `sysdir`.

Next, I change to the working directory I want for project B and run `cdsave` again:

```
. cd d:/projectb/work
d:/projectb/work

. cdsave projectb
command cdprojectb saved in your personal directory
```

Now I can use these commands to move between directories. To change to project A,

```
. cdprojecta
d:/projecta/work
```

where the command echoes the working directory that has been set. To change to project B:

```
. cdprojectb
d:/projectb/work
```

I can create as many the `cd`*project* commands as I want. To list my commands,

```
. cdlist
      cdcda - d:/active/cda2017/work/
     cddesk - c:/users/jslong/desktop/
 cdprojecta - d:/projecta/work/
 cdprojectb - d:/projectb/work/
    cdstart - d:/statastart/
  cdwficpsr - d:/active/wficpsr2017/work/
 cdworkflow - d:/active/workflowwd/work/
```

The command names on the left are shown in blue, and if I click on them the command is executed. (This feature is not available when running Stata in console mode.) I can also run the command from the Command window or include it in a do-file.

As the number of `cd`*project* commands increase, scanning the directories becomes awkward. A more efficient approach is to use the `note()` option when creating a command:

```
. cdsave projecta, note(Project A with Fred Mason) replace
command cdprojecta saved in your personal directory
```

After creating several commands,

```
. cdlist
      cdcda - CDA class
     cddesk - Desktop
 cdprojecta - Project A  with Fred Mason
 cdprojectb - d:/projectb/work/
    cdstart - Default starting directory
  cdwficpsr - ICPSR WF class
 cdworkflow - Workflow cd commands
```

For `cdprojectb` the working directory is listed since no note was added when the command was created. Even when notes are set for projects, I can list the directories set by each command by running `cdlist, dir`.

1.2 Controlling the starting working directory

There is no easy way to change the working directory in which Stata starts. One solution is to run `cdlist` as soon as Stata starts and click on the project where you want to start. Alternatively, you can modify the `profile.do` file which is automatically run each time Stata is started. To find where `profile.do` is located is to run:

```
findfile profile.do, path(STATA;BASE;SITE;PERSONAL;PLUS)
```

If it is not found, you need to create the file. To determine the best location for the file, check *Stata Installation Guide* or *Getting Stata with Stata* for your operating system. You can find these guides by running `help getting started`.

You can add a `cd` command to `profile.do` to set the starting working directory. For example,

```
cd "d:/statastart/"
```

If you want to change the starting directory, you must edit `profile.do` again. Since I like Stata to open to the project I am currently working on and since these project change frequently, I find editing `profile.do` to be cumbersome. An easier solution is to use `cdsave` to create the command `cdstart` to change to the working directory you want to start in. I add `cdstart` to `profile.do` so that `cdstart` will run each time Stata starts. If I want to change the startup directory, I replace `cdstart` with a version that opens up the directory I want. For example,

```
cd d:/projecta/work
cdsave start, replace
```

Even if Stata opens the the working directory where I am focusing my work, I often want to work on other projects, if only for a short time. To make it easy to change directories immediately after I start Stata, I add `cdlist` to `profile.do`:

```
cdstart
cdlist
```

The `cdstart` command opens with my preferred working directory and then `cdlist` shows me other locations I can chose. To prevent an error if the `workflow` commands have not been installed on the computer I am using, I use the more robust commands:

```
capture noisily cdstart
capture noisily cdlist
```

1.3 Robust specification of data paths

The `cd`*project* commands can also create a global indicating where datasets are stored for the project. The `use` command loads a dataset from the working directory unless a path has been specified. This path can by a directory on your computer, a location on a LAN, or an URL. While I prefer to keep the datasets for a project in that project's working directory, in some cases this is not possible. For example, I might need to access data over the web or in a collaboration datasets might be located on a shared directory on a LAN. Or, I might simply prefer to have my datasets stored someplace other than the working directory.

For your do-files to run on other computers, you should not hard code a directory location in your do-files. That is, you should not include a command like this:

```
use d:/data/binlfp4, clear
```

If I do, the do-file will not run on a computer that does not have exactly the same directory structure, including drive letters or names. Globals can be used so that your do-files load datasets from locations other than your working directory in a way that the do-files are portable across computers. To do this, create a global with the path where data is saved. For example,

```
global S_cddata "d:/datasets/"
```

This global is *not* created in my do-file since I want the file to run without changes on other computers. My do-file loads data with the command

```
use "${S_cddata}binlfp4", clear
```

which is interpreted as

```
use "d:/datasets/binlfp4", clear
```

If the global is not defined, the command is interpreted as

```
use "binlfp4", clear
```

If I am working on a computer with a different file structure, my do-file will still run if I first create the global `S_cddata`. For example,

```
global S_cddata "c:/users/jslong/documents/datasets/"
```

If the ending slash was missing, I would need to load my dataset with this command:

```
use "${S_cddata}/binlfp4", clear
```

I could even remove the parentheses:

```
use "$S_cddata/binlfp4", clear
```

While this is simpler, an error occurs if the global has not been defined since the command is interpreted as

```
use "/binlfp4", clear
```

which causes an error.

`cdsave` lets your `cd`*project* command create three globals defining data locations. The `dataset(`*path*`)` option creates the global `S_cddata` that contains the path where datasets are located. For example,

```
cdsave sdsc, data(k:/user/jslong/sdsc/datasets)
```

The global `S_cdurl` can hold a web address by using the `url(`*address*`)` option. For example,

```
cdsave spost, url(http://www.indiana.edu/~jslsoc/stata/spex_data/)
```

Since you might want more than two data locations, perhaps one for source datasets and another for datasets you create, the `user(`*string*`)` option creates the global `S_cduser` that can hold another path.

You can add both a data path, an URL, and a user path at the same time. For example,

```
cdsave paths, note(Example of data paths) ///
    data(d:/datasets/source) user(d:/datasets/derived/) ///
    url(http://www.indiana.edu/~jslsoc/stata/spex_data) replace
```

`cdsave` automatically adds an ending `/` to the data path and URL, but you must add it yourself with the `user()` option since this option could hold information where you do not want an ending slash. If you do not want an ending slash the data locations, add the `noslash` option to `cdsave`.

After running a `cd`*project* that includes data locations, a do-file could use and save data from multiple locations:

```
use "${S_cddata}binlfp4", clear
  (output omitted)
save "${S_cduser}binlfp5", replace
  (output omitted)
use ${S_cdurl}nomocc4, clear
  (output omitted)
use "couart4", clear // load data from working directory
  (output omitted)
```

The quotes are included in case a path includes a space.

There are two ways to check the globals created by a command. First, run `cdinfo`:

```
. cdinfo
  S_cdcmd:    cdpaths
  S_cdauto:
  S_cddata:   d:/data/source
  S_cdnote:   Example of data paths
  S_cdurl:    http://www.indiana.edu/~jslsoc/stata/spex_data/
```

```
  S_cduser:   d:/data/derived/
  S_cdwd:     d:/dropbox/active/workflow/work/
```

The data locations are saved using `/` which works in Win, Mac, and Unix. If you enter a path using `\` it is converted to `/`. The list of globals can also be obtained by adding `details` or simply `d` when running a project command:

```
. cdpaths d
d:\workflow\work
  S_cdcmd:    cdpaths
  S_cdauto:
  S_cddata:   d:/data/source/
  S_cdnote:   Example of data paths
  S_cdurl:    http://www.indiana.edu/~jslsoc/stata/spex_data/
  S_cduser:   d:/data/derived/
  S_cdwd:     d:/workflow/work/
```

Since the names of the globals begins with `S_`, they are considered to be system globals which are not dropped when `macro drop all` is run. The names begin with `S_cd` to make them less likely to conflict with official Stata system globals.

1.4 Further customization of your research environment

The `autorun(`*command*`)` option specifies a command that is run by `cd`*project* command after the working directory is changed. The command can be an internal Stata command. For example,

```
cdsave autodir, auto(dir *.dta)
```

When I run `cdautodir`, datasets in the working directory are listed:

```
. cdautodir
d:/datasets
  51.1k   9/08/16  8:48  binlfp4.dta
  30.6k   4/24/14  6:02  couart4.dta
  69.4k  11/20/13 15:31  gssclass4.dta
  75.9k   3/02/14 11:19  gsskidvalue4.dta
  39.0k   3/12/14 13:26  partyid4.dta
  32.0k   3/02/14 11:03  science4.dta
 935.4k   3/04/14 13:16  svyhrs4.dta
  12.3k   4/01/14  7:14  travel4.dta
 153.6k   8/11/14 15:08  wlsrank4.dta
```

Or, you could automatically load a dataset:

```
cdsave autouse, auto(use binlfp4, clear)
```

Then,

```
. cdautouse
d:\workflow\work
. use binlfp4, clear
(binlfp4.dta | Mroz data on labor force participation of women | 2014-10-20)
```

where `cdautouse` automatically ran `use binlfp4, clear`. If you want to automatically run more than one command, you can include them in a do file located in the working directory for the project:

```
cdsave projectc, autorun(do projectc-setup)
```

2 Programming features used by the workflow package

The `workflow` package depends on several features of Stata.

Dynamically creating ado files Non-programmers are unlikely to learn how to program simply to make it easier to change working directories. Programmers are unlikely to take the time to write such mundane commands. The `cdsave` command does this automatically by using `file write` to create `cd`*project*`.ado` with the commands for changing the work directory, creating globals, and automatically running commands.

The PERSONAL directory The `cd`*project*`.ado` files are automatically saved in the PERSONAL directory so that they will run regardless of your current working directory. If PERSONAL is not defined or the directory specified in PERSONAL does not exist, `cdsave` exists with an error. Run `sysdir` to determine where your PERSONAL is located. If no directory is listed or it points to a directory that does not exist, you can set the directory with `sysdir set`; enter `help sysdir` for details. For work on a network, see section 7.

Use of system macros It is generally better if your do-files do not depend on globals created outside of the do-file. If they do, the do-file might not run correctly if those globals do not exist, which makes it more difficult to reproduce your analyses. Accordingly, I typically include `macro drop _all` at the start of each do-file. Since the globals created by `cd`*project* commands need to persist as long as you are working on that project, system globals are used. System globals are globals whose names with `S_` which can only be deleted by `macro drop S_`*name* or `macro drop S_*`. The following globals are created by each `cd`*project* command:

`S_cdauto`	User specified command executed by `cd`*project*.
`S_cdcmd`	Name of the current `cd`*project* command.
`S_cddata`	Path for using datasets.
`S_cdnote`	Note associated with `cd`*project*.
`S_cdurl`	Web address for using datasets.
`S_cduser`	Any string.
`S_cdwd`	Working directory set by `cd`*project*.

3 cdsave: creating cd commands

The `cdsave` command creates a `cd`*project*`.ado` file that is saved in the PERSONAL directory. When no options are specified, the only thing that the `cd`*project* command does is change the working directory to the working directory that was active when `cdsave` was run. The syntax is:

`cdsave cd`*project* [`, note(`*string*`) datapath(`*data-path*`) url(`*url*`)`
`autorun(`*command*`) details user(`*string*`) noslash replace`]

where *project* is the mnemonic for your project. Commands created by `cdsave` must begin with the letters `cd`, but you can enter the name without the `cd` prefix. For example, both `cdsave myproject` and `cdsave cdmyproject` can be used.

Options

`note(`*string*`)` adds a note describing the project. If no note is specified, the working directory is used as the note. Notes are shown by `cdlist` to help you remember what each command does.

`datapath(`*data-path*`)` is a path where datasets are located. The path is stored in the global `S_cddata`. An ending `/` is added to the path unless the `noslash` option is used. You can load a dataset from this path with `use ${S_cddata}`*filename* or save a dataset with `save ${S_cddata}`*filename*.

`url(`*web-address*`)` is a web address which is stored in the global `S_cdurl`. An ending `/` is added to the path unless the `noslash` option is used. You can load datasets with `use ${S_cdurl}`*filename*.

`user(`*string*`)` can by any string, such as another data directory or URL. The string is stored in the global `S_cduser`.

`autorun(`*command*`)` is a command that is run by `cd`*project*. This can be a Stata command, an ado file you wrote, or a do-file such as `auto(do projectstartup)`.

`details` lists the globals that will be created when `cd`*project* is run.

`noslash` prevents an ending `/` from being added to the data path or URL.

`replace` overwrite `cd`*project*`.ado` if it exists.

Globals created by `cd`*project* commands are not removed by `macro drop _all`. To remove them you must run `macro drop` *name*, such as `macro drop S_cddata` or `macro drop S_cd*`.

4 The cdlist command

`cdlist` lists the names of ado-files in PERSONAL that begin with `cd` along with the note

for each project. If you click on the name of a command, which is shown in blue, the command is executed. The syntax is:

`cdlist` [`, directories details`]

If the working directory for a command is not found, a warning is given.

Options

`details` displays the `S_cd` globals created by each command.

`directories` displays the directory being set even if a note is defined.

5 The cdinfo commands

`cdinfo` lists the `S_cd*` macros that are in memory.

`cdinfo`

There are no options.

6 The cddrop command

`cddrop` drops, deletes, or restores `cd`*project*`.ado` files in PERSONAL.

`cddrop cd`*project* | `_all` [`, delete restore`]

`cd`*project* is the name of the command whose ado file in PERSONAL is modified. By default, `cddrop cd`*project* renames `cd`*project*`.ado` to `cd`*project*`.dropped`. After the command is renamed, it will no longer run. With the `delete` option, the ado file is deleted. A file that has been renamed can be restored with the `restore` option. For example, `cddrop cdexample, restore` renames `cdexample.dropped` to `cdexample.ado`. The abbreviation `_all` will drop or restore all `cd` commands.

Options

`delete` erase the ado file for the command.

`restore` renames `cd`*project*`.dropped` to `cd`*project*`.ado`.

7 Workflow commands with Stata on a network

For user written commands to work with networked version of Stata, Stata must know where to find the ado files and those files must persist from one session to the next. The

instructions that follow should work with most networked versions of Stata. If not, you need to ask you system administrator for help.

On most networks, you will not have permission to write files to Stata's system directories because these are protected to avoid a user from changing something that is essential for Stata to work properly. Accordingly, you need a directory location where you have read/write permissions and which cannot be changed by others. In the instructions that follow, I assume that `u:` is a drive or directory that you control. This could be a flash drive, a directory on a LAN, a location in the Cloud, or a directory on the computer you are using to log into the network. You can replace `u:` with whatever location you want to use. Here are the steps we need to be taken:

1. Create `u:/profile.do` that sets up your environment in Stata, including directory locations where user written commands will be installed.
2. When Stata is not installed on a network, `profile.do` is usually located where Stata can find and run the file each time Stata is started. Depending on how Stata is installed on your network, this might not be possible. Accordingly, each time you start Stata, run `do u:/profile.do` from the command line.
3. After running `profile.do`, install user written packages. These will be saved to the location set in `profile.do`. To install the `workflow` package, run `search workflow` and follow the links that are provided. Do the same thing for other packages you want. You only need to install each package once, although periodically you should run `adoupdate,update` to update the commands.
4. Stata will find the user written commands, including the `cd`*project* commands created by `cdsave`.

Assuming that `u:/stataado` is where you want to save user written files, `profile.do` would contain these commands:

```
//  profile.do for Stata on a network - 2017-08-18
sysdir set PERSONAL "u:/stataado"
net set ado PERSONAL
capture noisily cdstart
capture noisily cdlist
exit
```

You can add other commands that you want to run each time Stata starts, such as:

```
set logtype text
set matsize 11000
set scrollbufsize 2000000
set linesize 80
```

If different computers on the network have different drive letters assigned, you might need to create different do-files for different locations of your files. For example, if you sometimes log on to a network where your flash drive is assigned letter `v:`, you could create `profilev.do` containing:

```
//  profilev.do for Stata on a network - 2017-08-18
sysdir set PERSONAL "v:/stataado"
net set ado PERSONAL
capture noisily cdstart
capture noisily cdlist
exit
```

Remember that each time you start Stata on the network, you must run `profile.do`.

8 Conclusions

I hope these commands are useful for developing an efficient workflow that supports reproducible results.

9 References

Baum, C. F. 2016. *An introduction to Stata programming*, vol. 2. 2nd ed. Stata Press College Station.

Long, J. S. 2009. *The Workflow of Data Analysis Using Stata*. College Station, TX: Stata Press.

Winters, N. 2002. fastcd: module to automate changing directories. http://www.nd.edu/~rwilliam/stata/gologit2.pdf..